

2014



Proceedings, Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Edited by
Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter

Palo Alto, California
AAAI Press

Copyright © 2014,
Association for the Advancement of Artificial Intelligence
AAAI Press
2275 East Bayshore Road, Suite 160
Palo Alto, California 94025

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

ISBN 978-1-57735-657-8 Printed Proceedings (softcover).
Online ISSN 2334-1033
Print ISSN 2334-1025

Printed on acid free paper.

Book and electronic proceedings manufactured in the United States of America

The Foureenth International Conference on Principles of Knowledge Representation
and Reasoning was held in Vienna, Austria, July 20–24, 2014

Contents

Conference Committee / xi

Area Chairs and Program Committee / xii

Preface / xv
Chitta Baral, Giuseppe De Giacomo, Thomas Eiter

Vienna Summer of Logic 2014 / xvii
Matthias Baaz, Thomas Eiter, Helmut Veith

Invited Talks / xix
Franz Baader, Tony Cohn, Georg Gottlob, Sheila McIlraith

Tutorials / xxii
Alessio Lomuscio, Lawrence S. Moss, Ekaterina Ovchinnikova, Riccardo Rosati

Sponsors / xxiv

Full Papers

Answer Set Programming and Logic Programming

Logic Programs with Ordered Disjunction: First-Order Semantics and Expressiveness / 2
Vernon Asuncion, Yan Zhang, Heng Zhang

Constructive Negation in Extensional Higher-Order Logic Programming / 12
Angelos Charalambidis, Panos Rondogiannis

The Well-Founded Semantics Is the Principle of Inductive Definition, Revisited / 22
Marc Denecker, Joost Vennekens

The Semantics of Gringo and Infinitary Propositional Formulas / 32
Amelia Harrison, Vladimir Lifschitz, Fangkai Yang

Argumentation

An SCC Recursive Meta-Algorithm for Computing Preferred Labellings in Abstract Argumentation / 42
Federico Cerutti, Massimiliano Giacomin, Mauro Vallati, Marina Zanella

On the Revision of Argumentation Systems: Minimal Change of Arguments Statuses / 52
Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, Pierre Marquis

A Dynamic Logic Framework for Abstract Argumentation / 62
Sylvie Doutre, Andreas Herzig, Laurent Perrussel

Characteristics of Multiple Viewpoints in Abstract Argumentation / 72
Paul E. Dunne, Wolfgang Dvorák, Thomas Linsbichler, Stefan Woltran

Automated Reasoning and Computation

The Parameterized Complexity of Reasoning Problems Beyond NP / 82
Ronald de Haan, Stefan Szeider

On OBDDs for CNFs of Bounded Treewidth / 92
Igor Razgon

Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory / 101
Hannes Strass, Johannes Peter Wallner

Skolemization for Weighted First-Order Model Counting / 111
Guy Van den Broeck, Wannes Meert, Adnan Darwiche

Belief Revision and Nonmonotonicity

On Egalitarian Belief Merging / 121
Patricia Everaere, Sébastien Konieczny, Pierre Marquis

Justified Beliefs by Justified Arguments / 131
Davide Grossi, Wiebe Van Der Hoek

Belief Change Operations: A Short History of Nearly Everything, Told in Dynamic Logic of Propositional Assignments / 141
Andreas Herzig

Belief Change and Base Dependence / 151
Mehrdad Oveisi, James P. Delgrande, Francis Jeffry Pelletier, Fred Popowich

Belief Change and Semiorders / 160
Pavlos Peppas, Mary-Anne Williams

David Poole's Specificity Revised / 168
Claus-Peter Wirth, Frieder Stolzenburg

Causality and Rationality

Axiomatizing Rationality / 178
Adam Bjorndahl, Joseph Y. Halpern, Rafael Pass

Dynamic Causal Calculus / 188
Alexander Bochman

Appropriate Causal Models and Stability of Causation / 198
Joseph Y. Halpern

∃GuaranteeNash for Boolean Games Is NEXP-Hard / 208
Egor Ianovski, Luke Ong

Description Logics

Nested Regular Path Queries in Description Logics / 218
Meghyn Bienvenu, Diego Calvanese, Magdalena Ortiz, Mantas Simkus

Decidable Gödel Description Logics without the Finitely-Valued Model Property / 228
Stefan Borgwardt, Felix Distel, Rafael Peñaloza

Query Inseparability for Description Logic Knowledge Bases / 238
Elena Botoeva, Roman Kontchakov, Vladislav Ryzhikov, Frank Wolter, Michael Zakharyaschev

Answering Instance Queries Relaxed by Concept Similarity / 248
Andreas Ecke, Rafael Peñaloza, Anni-Yasmin Turhan

Stable Model Semantics for Guarded Existential Rules and Description Logics / 258
Georg Gottlob, André Hernich, Clemens Kupke, Thomas Lukasiewicz

Polynomial Combined Rewritings for Existential Rules / 268
Georg Gottlob, Marco Manna, Andreas Pieris

Lightweight Description Logics and Branching Time: A Troublesome Marriage / 278
Víctor Gutiérrez-Basulto, Jean Christoph Jung, Thomas Schneider

Finite Model Reasoning in Horn Description Logics / 288
Yazmín Ibañez-García, Carsten Lutz, Thomas Schneider

Exact Learning of Lightweight Description Logic Ontologies / 298
Boris Konev, Carsten Lutz, Ana Ozaki, Frank Wolter

Nominal Schemas in Description Logics: Complexities Clarified / 308
Markus Krötzsch, Sebastian Rudolph

Practical Uniform Interpolation and Forgetting for $\mathcal{ALC}$ TBoxes with Applications to Logical Difference / 318
Michel Ludwig, Boris Konev

Knowledge Representation and Reasoning

Certain Answers as Objects and Knowledge / 328
Leonid Libkin

A Formalization of Programs in First-Order Logic with a Discrete Linear Order / 338
Fangzhen Lin

Simultaneous Learning and Prediction / 348
Loizos Michael

Tackling Winograd Schemas by Formalizing Relevance Theory in Knowledge Graphs / 358
Peter Schüller

Generalized Multi-Context Systems / 368
Shahab Tasharrofi, Eugenia Ternovska

Qualitative Spatial Representation and Reasoning in Angry Birds: The Extended Rectangle Algebra / 378
Peng Zhang, Jochen Renz

Planning, Strategies, and Diagnosis

Diagnostic Problem Solving via Planning with Ontic and Epistemic Goals / 388
Jorge A. Baier, Brent Mombourquette, Sheila A. McIlraith

Satisfiability of Alternating-Time Temporal Epistemic Logic through Tableaux / 398
Francesco Belardinelli

Reasoning about Equilibria in Game-Like Concurrent Systems / 408
Julian Gutierrez, Paul Harrenstein, Michael Wooldridge

A Temporal Logic of Strategic Knowledge / 418
Xiaowei Huang, Ron van Der Meyden

An Abstraction Technique for the Verification of
Multi-Agent Systems against ATL Specifications / 428
Alessio Lomuscio, Jakub Michaliszyn

Reasoning about Actions and Processes

How to Progress Beliefs in Continuous Domains / 438
Vaishak Belle, Hector J. Levesque

Transforming Situation Calculus Action Theories for Optimised Reasoning / 448
Christopher Ewin, Adrian R. Pearce, Stavros Vassos

State-Boundedness in Data-Aware Dynamic Systems / 458
Babak Bagheri Hariri, Diego Calvanese, Marco Montali, Alin Deutsch

Decidable Reasoning in a Fragment of the Epistemic Situation Calculus / 468
Gerhard Lakemeyer, Hector J. Levesque

A First-Order Semantics for Golog and ConGolog under a
Second-Order Induction Axiom for Situations / 478
Fangzhen Lin

Model Checking Unbounded Artifact-Centric Systems / 488
Alessio Lomuscio, Jakub Michaliszyn

Forgetting in Action / 498
David Rajaratnam, Hector J. Levesque, Maurice Pagnucco, Michael Thielscher

Reports from the Field

Computing Narratives of Cognitive User Experience for Building Design Analysis: KR for Industry Scale
Computer-Aided Architecture Design / 508
Mehul Bhatt, Carl Schultz, Madhura Thosar

SmartPM: An Adaptive Process Management System through
Situation Calculus, IndiGolog, and Classical Planning / 518
Andrea Marrella, Massimo Mecella, Sebastian Sardina

Tweety: A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and
Knowledge Representation / 528
Matthias Thimm

Uncertainty

Reasoning with Uncertain Inputs in Possibilistic Networks / 538
Salem Benferhat, Karim Tabia

Relational Logistic Regression / 548
Seyed Mehran Kazemi, David Buchman, Kristian Kersting, Sriraam Natarajan, David Poole

Probabilistic Sentential Decision Diagrams / 558
Doga Kisa, Guy Van den Broeck, Arthur Choi, Adnan Darwiche

Linear Programs for Measuring Inconsistency in Probabilistic Logics / 568
Nico Potyka

Short Papers

Axioms .2 and .4 as Interaction Axioms / 579
Guillaume Aucher

Stable Models of Multi-Valued Formulas: Partial versus Total Functions / 583
Michael Bartholomew, Joohyung Lee

Rough Set Semantics for Identity on the Web / 587
Wouter Beek, Stefan Schlobach, Frank van Harmelen

On the Progression of Knowledge in Multiagent Systems / 590
Vaishak Belle, Gerhard Lakemeyer

Interval Methods for Judgment Aggregation in Argumentation / 594
Richard Booth, Edmond Awad, Iyad Rahwan

Strong Equivalence of Non-Monotonic Temporal Theories / 598
Pedro Cabalar, Martín Diéguez

Using Answer Set Programming for Solving Boolean Games / 602
Sofie De Clercq, Martine De Cock, Kim Bauters, Steven Schockaert, Ann Nowé

Toward a Knowledge Level Analysis of Forgetting / 606
James P. Delgrande

A Psychology-Inspired Approach to Automated Narrative Text Comprehension / 610
Irene-Anna Diakidoy, Antonis Kakas, Loizos Michael, Rob Miller

Concept Dissimilarity with Triangle Inequality / 614
Felix Distel, Jamal Atif, Isabelle Bloch

On Redundant Topological Constraints (Extended Abstract) / 618
Matt Duckham, Sanjiang Li, Weiming Liu, Zhiguo Long

Belief Revision in the Propositional Closure of a Qualitative Algebra / 622
Valmi Dufour-Lussier, Alice Hermann, Florence Le Ber, Jean Lieber

How to Argue for Anything: Enforcing Arbitrary Sets of Labellings Using AFs / 626
Sjur Kristoffer Dyrkolbotn

Knowledge Maps of Web Graphs / 630
Valeria Fionda, Claudio Gutierrez, Giuseppe Pirró

ASP Encodings of Acyclicity Properties / 634
Martin Gebser, Tomi Janhunen, Jussi Rintanen

Tracking Beliefs and Intentions in the Werewolf Game / 638
Codruta Girlea, Eyal Amir, Roxana Girju

Representing and Reasoning about Time Travel Narratives: Foundational Concepts / 642
Leora Morgenstern

Aggregative Deontic Detachment for Normative Reasoning / 646
Xavier Parent, Leon van der Torre

Action Theories over Generalized Databases
with Equality Constraints (Extended Abstract) / 650
Fabio Patrizi, Stavros Vassos

An Abductive Reasoning Approach to the Belief-Bias Effect / 653
Luís Moniz Pereira, Emmanuelle-Anna Dietz, Steffen Hölldobler

Minimal Change in AGM Revision for Non-Classical Logics / 657
Marcio M. Ribeiro, Renata Wassermann

Predicting Performance of OWL Reasoners: Locally or Globally? / 661
Viachaslau Sazonau, Uli Sattler, Gavin Brown

Canonical Logic Programs Are Succinctly Incomparable with Propositional Formulas / 665
Yuping Shen, Xishun Zhao

Heuristic Guided Optimization for Propositional Planning / 669
Andreas Sideris, Yannis Dimopoulos

First-Order Default Logic Revisited / 673
Yi Zhou

Invited Talks

Ontology-Based Monitoring of Dynamic Systems / 678
Franz Baader

Datalog+/–: Questions and Answers / 682
Georg Gottlob, Thomas Lukasiewicz, Andreas Pieris

Index / 687

Conference Committee

General Chair

Thomas Eiter (Vienna University of Technology Austria)

Program Chairs

Chitta Baral (Arizona State University, USA)
Giuseppe De Giacomo (Sapienza University of Rome, Italy)

Local Organization

Michael Fink (Vienna University of Technology, Austria)
Stefan Woltran (Vienna University of Technology, Austria)

Doctoral Consortium Chair

Birte Glimm (University of Ulm, Germany)
Adrian Pearce (University of Melbourne, Australia)

Sponsorship

Marco Maratea (University of Genova, Italy)

Area Chairs and Program Committee

Area Chairs

Franz Baader
Vinay Chaudhri
Adnan Darwiche
Jim Delgrande
Luc De Raedt
Didier Dubois
Phan Minh Dung
Hector Geffner
Ian Horrocks
Sarit Kraus
Jerome Lang
Maurizio Lenzerini
Yves Lesperance
Vladimir Lifschitz
Fangzhen Lin
Alessio Lomuscio
Sheila McIlraith
Daniele Nardi
Jochen Renz
Torsten Schaub
Michael Thielscher
Francesca Toni
Frank Wolter
Yan Zhang

Program Commtitee

Thomas Ågotnes
Natasha Alechina
Eyal Amir
Jorge Baier
Marcello Balduccini
Mohua Banerjee
Pietro Baroni
Roberto Basili
Francesco Belardinelli
Brandon Bennett
Mehul Bhatt
Meghyn Bienvenu
Blai Bonet
Johan Bos
Alex Borgida
Stefano Borgo
Gerhard Brewka
Krysia Broda
Pedro Cabalar
Diego Calvanese
Martin Caminada
Xiaoping Chen
Yin Chen
Arthur Choi
Jens Classen
Robert Craven
James Cussens
Marc Denecker
Yannis Dimopoulos
Patrick Doherty
Paul Dunne
Stefan Edelkamp
Ulle Endriss
Esra Erdem
Jérôme Euzenat
Wolfgang Faber
Xiuyi Fan
Michael Fink
Alfredo Gabaldon
Yang Gao
Martin Gebser
Michael Gelfond
Michael Genesereth
Alfonso E. Gerevini
Malik Ghallab
Massimiliano Giacomin
Birte Glimm
Lluis Godo
Georg Gottlob
John Grant
Bernardo Cuenca Grau
Davide Grossi
Michael Gruninger
Claudio Gutierrez
Patrik Haslum
Nick Hawes
Malte Helmert
Andreas Herzig
Stijn Heymans
Pascal Hitzler
Jinbo Huang
Anthony Hunter
Giovambattista Ianni
Katsumi Inoue
Luca Iocchi
Wojtek Jamroga
Tomi Janhunen
Jianmin Ji
Antonis Kakas
Yevgeny Kazakov
Gabriele Kern-Isberner
Kristian Kersting
Roni Khardon
Matthew Klenk
Pavel Klinov
Sebastien Konieczny
Roman Kontchakov
Markus Krötzsch
Gerhard Lakemeyer
Joohyung Lee
Daniel Lehmann
Joao Leite
Nicola Leone
Yuliya Lierler
Jason Jingshi Li
Sanjiang Li
Weiru Liu
Yongmei Liu
Jorge Lobo
Emiliano Lorini
Thomas Lukasiewicz
Carsten Lutz
Jiefei Ma
Pierre Marquis
John-Jules Meyer
Thomas Meyer
Sanjay Modgil

Dan Moldovan
Ralf Möller
Angelo Montanari
Serafin Moral
Maxime Morge
Leora Morgenstern
Yoram Moses
Boris Motik
Bernhard Nebel
Nir Oren
Magdalena Ortiz
Mauricio Osorio
Maurice Pagnucco
Odile Papini
Peter Patel-Schneider
Fabio Patrizi
Adrian Pearce
Rafael Peñaloza
Pavlos Peppas
Patrice Perny
Ron Petrick
Gabriella Pigozzi
David Poole
Henry Prakken
Jussi Rintanen
Riccardo Rosati
Ji Ruan
Sebastian Rudolph
Alessandra Russo
Alessandro Saffiotti
Chiaki Sakama
Paulo E. Santos
Sebastian Sardina
Vítor Santos Costa
Ken Satoh
Uli Sattler
Stephan Schiffel
Steven Schockaert
Luciano Serafini
Yi-Dong Shen
Laurent Simon
Tran Cao Son
John Stell
Hannes Straß
Andrzej Szałas
Moritz Tenorth
Eugenia Ternovska
Sylvie Thiebaux
David Toman
Hans Tompits
Mirek Truszczynski
Guy Van Den Broeck
Wiebe Van Der Hoek
Hans van Ditmarsch
Barbara Vantaggi
Stavros Vassos
Kewen Wang
Yisong Wang
Renata Wassermann
Michael Wessel
Mary-Anne Williams
Stefan Woelfl
Diedrich Wolter
Stefan Woltran
Michael Wooldridge
Maonian Wu
Jia-Huai You
Michael Zakharyaschev
Bruno Zanuttini
Dongmo Zhang
Yuanlin Zhang
Yi Zhou

Additional Reviewers

Mario Alviano
Nahla Ben Amor
Mikkel Birkegaard Andersen
Ofer Arieli
Arnon Avron
Marco Baioletti
Ringo Baumann
Bart Bogaerts
Stefan Borgwardt
Francesco Calimeri
David Carral
Suming Chen
Roberto Confalonieri
Fabio Gagliardi Cozman
Jose-Luis Craballido
Graham Deane
Broes De Cat
Bertrand Decoster
Jun Deng
Christian Eichhorn
Patricia Everaere
Christopher Ewin
Liangda Fang
Paolo Felli
Johannes Klaus Fichte
Valeria Fionda
Oliver Fernandez Gil
Codruta Girlea
Sara Girotto
Umberto Grandi
Joachim Jansen
Guifei Jiang
Roland Kaminski
Eric Kao
Tabia Karim
Szymon Klarman
Sophia Knight
Ilianna Kollia
Seifeddine Kramdi
Emily Leblanc
Churn-Jung Liau
Guohua Liu
Theofilos Mailis
Marco Manna
Maria Vanina Martinez
Loizos Michael
Karthika Mohan
Abhijeet Mohapatra
Kody Moodley
Kedian Mu
Christian Muise
Raghava Mutharaju
Juan Antonio Navarro Perez
Rolf Niedermeier
Juan Carlos Nieves
Nadeschda Nikitina
Vilem Novak
Philipp Obermeier
Johannes Oetsch
Oezguer Luetfue Oezcep
Simona Perri
Andreas Pieris
Denis Ponomaryov
Enrico Pontelli
Daniele Porello

Wen Pu
Jörg Pührer
Guilin Qi
Odinaldo Rodrigues
Paulo Rodrigues
Javier Romero
Fariba Sadri
Alessandro Saetti
Roberto Baginski Batista Santos
Claudia Schulz
Nicolas Schwind
Kunal Sengupta
Marek Sergot
Gerardo Simari
Mantas Simkus
John Slaney
Andreas Steigmiller
Jakob Suchan
Michael Thomazo
Kazuko Takahashi
Giorgio Terracina
Son To
David Toman
Gianluca Torta
Calin-Rares Turliuc
Pieter Van Hertum
Ivan Varzinczak
Hanne Vlaeminck
Johannes P. Wallner
Cong Wang
Shenling Wang
Yanjing Wang
Yi N. Wang
Zhe Wang
Florian Widmann
Maonian Wu
Guohui Xiao
Jiongkun Xie
Marina Zanella
Yuting Zhao
Dmitriy Zheleznyakov
Zhiqiang Zhuang

Preface

The KR conference series is a leading forum for timely in-depth presentation of progress in the theory and principles underlying the representation and computational management of knowledge.

The first Knowledge Representation & Reasoning (KR) conference was held 25 years ago in 1989. On this 25th year of the first KR conference, we have embarked on the 14th KR conference. While thoughts and discussions on KR go back to the early philosophers, the importance of KR in building intelligent systems was realized from the very beginning of the field of artificial intelligence (AI). This recognition and subsequent research on KR as an important aspect of AI was apt, as the two key aspects of "intelligence," modeling the available knowledge on the world and reasoning on the world by means of this model center around KR. Since then there has been significant progress in KR and now we have many well developed KR formalisms with supporting theories, building-block results, reasoning systems, and applications. There are many books that present these developments. We have reached a stage where various disciplines in AI and computer science are now using KR. Among them is the use of KR in agents, semantic web applications, video understanding, natural language understanding, and many others.

The 2014 edition of KR is held as part of the Vienna Summer of Logic together with the Federated Conference on Logic, Logic Colloquium, and other related events, making it a very special edition.

We received 220 submissions to the conference of which 58 (26.4 percent) were accepted as regular papers. In addition to regular papers, the conference hosts 25 short technical papers, following a tradition introduced at KR 2010. Accepted papers can be broadly divided into the following areas: description logics (11), reasoning about actions and processes (7), belief revision and nonmonotonicity (6), general knowledge representation and reasoning (6), planning, strategies, and diagnosis (5), answer set programming and logic programming (4), argumentation (4), automated reasoning and computation (4), causality and rationality (4), uncertainty (4), and reports from the field (3).

The conference program has been enriched with several tutorials and invited talks on traditional KR topics as well as on emerging application areas.

With respect to applications, Tony Cohn contributes an invited talk, Knowledge Representation Meets Computer Vision: From Pixels to Symbolic Activity Descriptions, which discusses the role of KR in video understanding; and Ekaterina Ovchinnikova holds a tutorial on Natural Language Understanding with World Knowledge and Inference, which discusses the role of KR in natural language understanding (NLU), as well as various available knowledge bases that can be and are being used in developing NLU systems.

Research in dynamic systems and multi-agent systems has been strongly influenced by advances in KR; this is underlined by Franz Baader in his invited VSL/KR Keynote talk, supported by ECCAI, Ontology-Based Monitoring of Dynamic Systems. In addition, Alessio Lomuscio presents the tutorial Verification of Multi-Agent Systems against Epistemic Specifications, and Lawrence S. Moss presents another tutorial Dynamic Epistemic Logic and its Interaction with Knowledge Representation.

The close connection between databases, ontologies and KR is emphasized by Georg Gottlob, who gives an invited talk Datalog+/–: Questions and An-

swers and by Riccardo Rosati in his tutorial Query Answering and Rewriting in Ontology-Based Data Access.

The invited talks are rounded out by the traditional Great Moments in KR talk, which Sheila McIlraith delivers on Situation Calculus: The Last 15 Years. This talk will honor Ray Reiter and address the impact of his ground-breaking work on situation calculus.

We would like to thank several people who helped in the many aspects that are needed to make KR 2014 a successful event. The Senior Program Committee and Program Committee members were extremely helpful and supportive in their thorough and timely review of the submissions. Birte Glimm and Adrian Pearce took over the important charge of organizing the doctoral consortium and Marco Maratea was chasing sponsorships and liaisoning with organizations full of verve. We gratefully acknowledge the support of our sponsors. A special thanks goes to Michael Fink and Stefan Woltran and their team for the local organization, including Eva Nedoma, Markus Pichlmair, Johannes Wallner, Thomas Linsbichler, Sylwia Polberg, Friedrich Slivovsky, and all volunteers. We are also very grateful to the VSL Organization Chair Vesna Sabljakovic-Fritz. And if you have been attracted by the amazing KR 2014 poster or webpage design: contact Paolo Felli!

In closing, we thank our KR colleagues for their contributions which appear in the following pages and welcome our readers to this record of KR 2014.

Chitta Baral and Giuseppe De Giacomo
Program Chairs

Thomas Eiter
General Chair
Vienna Austria
Summer of Logic
July, 2013

Vienna Summer of Logic



In the summer of 2014, Vienna hosted the largest scientific conference in the history of logic. The Vienna Summer of Logic consisted of twelve large conferences and 82 workshops, attracting more than 2,000 researchers from all over the world. This unique event was organized by the Kurt Gödel Society at Vienna University of Technology and held from July 9 to 24, 2014, under the auspices of the Federal President of the Republic of Austria, Dr. Heinz Fischer.

The conferences and workshops dealt with the main theme — logic — from three important angles: logic in computer science, mathematical logic, and logic in artificial intelligence. They naturally gave rise to respective streams gathering the following meetings:

Logic in Computer Science / Federated Logic Conference (FLoC)

- 26th International Conference on Computer Aided Verification (CAV)
- 27th IEEE Computer Security Foundations Symposium (CSF)
- 30th International Conference on Logic Programming (ICLP)
- 7th International Joint Conference on Automated Reasoning (IJCAR)
- 5th Conference on Interactive Theorem Proving (ITP)
- Joint meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL) and the 29th ACM/IEEE Symposium on Logic in Computer Science (LICS)
- 25th International Conference on Rewriting Techniques and Applications (RTA) joint with the 12th International Conference on Typed Lambda Calculi and Applications (TLCA)
- 17th International Conference on Theory and Applications of Satisfiability Testing (SAT)
- 76 FLoC Workshops
- FLoC Olympic Games (System Competitions)

Mathematical Logic

- Logic Colloquium 2014 (LC)
- Logic, Algebra and Truth Degrees 2014 (LATD)
- Compositional Meaning in Logic (GeTFun 2.0)
- The Infinity Workshop (INFINITY)
- Workshop on Logic and Games (LG)
- Kurt Gödel Fellowship Competition

Logic in Artificial Intelligence

- 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)
- 27th International Workshop on Description Logics (DL)
- 15th International Workshop on Non-Monotonic Reasoning (NMR)
- 6th International Workshop on Knowledge Representation for Health Care 2014 (KR4HC)

The VSL keynote talks, which were directed to all participants, were given by Franz Baader (Technische Universität Dresden), Edmund Clarke (Carnegie Mellon University), Christos Papadimitriou (University of California, Berkeley) and Alex Wilkie (University of Manchester); Dana Scott (Carnegie Mellon University) spoke in the opening session. Since the Vienna Summer of Logic contained more than a hundred invited talks, it is infeasible to list them here.

The program of the Vienna Summer of Logic was very rich, including not only scientific talks, poster sessions and panels, but also two distinctive events. One was the award ceremony of the Kurt Gödel Research Prize Fellowship Competition, in which the Kurt Gödel Society awarded three research fellowship prizes endowed with 100.000 Euro each to the winners. This was the third edition of the competition, themed Logical Mind: Connecting Foundations and Technology this year.

The other distinctive event were the 1st FLoC Olympic Games hosted by the Federated Logic Con-

ference (FLoC) 2014. Intended as a new FLoC element, the games brought together 12 established logic solver competitions by different research communities. In addition to the competitions, the Olympic Games facilitated the exchange of expertise between communities, and increased the visibility and impact of state-of-the-art solver technology. The winners in the competition categories were honored with Kurt Gödel medals at the FLoC Olympic Games award ceremonies.

Organizing an event like the Vienna Summer of Logic has been a challenge. We are indebted to numerous people whose enormous efforts were essential in making this vision become reality. With so many colleagues and friends working with us, we are unable to list them individually here. Nevertheless, as representatives of the three streams of VSL, we would like to particularly express our gratitude to all people who have helped to make this event a success: the sponsors and the honorary committee; the organization committee and the local organizers; the conference and workshop chairs and program committee members; the reviewers and authors; and of course all speakers and participants of the many conferences, workshops and competitions.

The Vienna Summer of Logic continues a great legacy of scientific thought that started in Ancient Greece and flourished in the city of Gödel, Wittgenstein and the Vienna Circle. The heroes of our intellectual past shaped the scientific world-view and changed our understanding of science. Owing to their achievements, logic has permeated a wide range of disciplines, including computer science, mathematics, artificial intelligence, philosophy, linguistics, and many more. Logic is everywhere – or in the language of Aristotle, πάντα πλήρη λογικῆς τέχνης.

– *Matthias Baaz, Thomas Eiter, Helmut Veith*
Vienna, July 2014

Invited Talks

VSL/KR Keynote Talk, Supported by ECCAI

Ontology-Based Monitoring of Dynamic Systems

Franz Baader, TU Dresden

Our understanding of the notion "dynamic system" is a rather broad one: such a system has states, which can change over time. Ontologies are used to describe the states of the system, possibly in an incomplete way. Monitoring is then concerned with deciding whether some run of the system or all of its runs satisfy a certain property, which can be expressed by a formula of an appropriate temporal logic. I consider different instances of this broad framework, which can roughly be classified into two cases. In one instance, the system is assumed to be a black box, whose inner working is not known, but whose states can be (partially) observed during a run of the system. In the second instance, one has (partial) knowledge about the inner working of the system, which provides information on which runs of the system are possible.

In this talk, I will review some of our recent research that investigates different instances of this general framework of ontology-based monitoring of dynamic systems. I will also sketch possible extensions towards probabilistic reasoning and the integration of mathematical modeling of dynamical systems.

Franz Baader has been a full professor of theoretical computer science at TU Dresden since 2002, and dean of the Faculty of Computer Science of TU Dresden since 2012. He obtained his PhD in computer science at the University of Erlangen in 1989. He was a senior researcher at the German Research Centre for Artificial Intelligence (DFKI) in Kaiserslautern and Saarbrücken for four years, and associate professor of theoretical computer science at RWTH Aachen for eight years. His main research area is logic in computer science and artificial intelligence, in particular knowledge representation (description logics, modal logics, ontologies) and automated deduction (term rewriting, unification theory, combination of decision procedures). He was program chair of the KI'01, CADE'03, LPAR'04, and RTA'07 conferences, is an editorial board member of several journals in AI and logic in computer Sscience, and has published more than 200 refereed articles in major journals and conferences. He is a member of the Academia Europaea and an ECCAI fellow, and was for twelve years president of CADE Inc., the organization that runs the International Conference on Automated Deduction.

Knowledge Representation Meets Computer Vision: From Pixels to Symbolic Activity Descriptions

Tony Cohn, University of Leeds

While the fields of KR and computer vision have diverged since the early days of AI, there have been recent moves to re-integrate these fields. I will talk about some of this work, focusing on research from Leeds on building models of video activity from video input. I will present techniques, both supervised and unsupervised, for learning the spatio-temporal structure of tasks and events from video or other sensor data, particularly in the case of scenarios with concurrent activities. The representations exploit qualitative spatio-temporal relations which have been an active area of research in KR for a number of years. I will also talk about the problem of robustly computing symbolic descriptions from noisy video data. Finally, I will show how objects can be functionally categorised according to their spatiotemporal behaviours.

Tony Cohn holds a personal chair at the University of Leeds, where he is professor of automated reasoning and director of the Institute for Artificial Intelligence and Biological Systems. His work on knowledge representation and reasoning has a par-

ticular focus on qualitative spatial/spatio-temporal reasoning, the best known being the well cited region connection calculus (RCC). His current research interests range from theoretical work on spatial calculi and spatial ontologies, to cognitive vision, modeling spatial information in the hippocampus, and detecting buried underground assets (e.g. utilities and archaeological residues) using a variety of geolocated sensors. He has been chairman or president of SSAISB, ECCAI, KR inc, the IJCAI Board of Trustees and is presently editor-in-chief of the AAAI Press, *Spatial Cognition and Computation,* and the *Artificial Intelligence* journal. He was elected a founding Fellow of ECCAI, and is also a Fellow of AAAI, AISB, the BCS, and the IET. Work from the Cogvis project won the British Computer Society Machine Intelligence prize in 2004, and the VAULT system from his Mapping the Underworld project won a 2012 IET Innovation Award.

Datalog+/–: Questions and Answers

Georg Gottlob, University of Oxford

Datalog$^{+/?}$ is a family of languages for knowledge representation and reasoning. These languages extend Datalog with features such as existential quantifiers, equalities, and the falsum in rule heads, and, at the same time, applies restrictions to achieve decidability and tractability. After a general overview of the Datalog$^{+/?}$ family, this talk will focus on more recent issues. Among other things, I will report on the combination of the two main decidability paradigms guardedness and stickiness, yielding the Tame Fragment, and on incorporating non-monotonic negation and disjunction into Datalog+/–:. I will also report about a special version of Datalog$^{+/?}$ suitable for reasoning with reverse-engineered UML class diagrams, and about the TriQ language that expresses SPARQL with entailment regimes.

Georg Gottlob is a professor of informatics at Oxford University, a Fellow of St John's College, Oxford, and an adjunct professor at TU Wien. His interests include knowledge representation and reasoning including ontological reasoning, logic and complexity, database theory, graph decomposition techniques, and web data extraction. Gottlob has received the Wittgenstein Award from the Austrian National Science Fund, is an ACM Fellow, an ECCAI Fellow, a Fellow of the Royal Society, and a member of the Austrian Academy of Sciences, the German National Academy of Sciences, and the Academia Europaea. He chaired the Program Committees of IJCAI 2003 and ACM PODS 2000. He is currently a member of the editorial boards of *CACM* and *JCSS.* He was the main founder of Lixto (www.lixto.com), a company that provides tools and services for web data extraction which was recently acquired by McKinsey & Company. Gottlob was awarded an ERC Advanced Investigator's Grant for the project DIADEM: Domain-centric Intelligent Automated Data Extraction Methodology project. More information on Georg Gottlob can be found on his Web page (www.cs.ox.ac.uk/people/georg.gottlob).

Great Moments in KR Talk

Situation Calculus: The Last 15 Years

Sheila McIlraith, University of Toronto

In 2001 Ray Reiter published his seminal book, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The book represented the culmination of 10 years of work by Reiter and his many collaborators investigating, formalizing, and extending the situation calculus, first introduced by John McCarthy in 1963 as a way of logically specifying dynamical systems. While researchers continue to extend the situation calculus, it has also seen significant scientific deployment to aid in the specification and implementation of a diversity of automated reasoning endeavors including diagnosis, web services composition and customization, and nonclassical automated planning. In this talk I will examine the important role the situation calculus has more recently been playing in explicating nuances in the logical specification and realization of some of these diverse automated reasoning tasks.

Sheila McIlraith is a professor in the Department of Computer Science, University of Toronto. Prior to joining the University of Toronto, she spent six years as a research scientist at Stanford University, and one year at Xerox PARC. McIlraith's research is in the area of knowledge representation and automated reasoning, with specific interest in reasoning

about dynamical systems and decision making in its many guises. She has 10 years of industrial research and development experience developing artificial intelligence applications. McIlraith is a fellow of AAAI and a past associate editor of the journal *Artificial Intelligence.* She is past program cochair of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR2012), and the International Semantic Web Conference (ISWC2004). In 2011 she and her coauthors were honored with the SWSA 10-year Award, recognizing the highest impact paper from the International Semantic Web Conference, 10 years prior. Her research has also made practical contributions to the development of next-generation space systems and to emerging web standards.

Tutorials

Verification of Multi-Agent Systems against Epistemic Specifications

Alessio Lomuscio, Imperial College London

Autonomous multiagent systems are typically specified by using expressive properties representing, among others, the knowledge of the agents and what they can bring about collectively or on their own in a system. This tutorial will survey some of the current techniques addressing the model checking of multi-agent systems against agent-based specifications. Specifically, the tutorial will cover syntax and semantics for the temporal-epistemic logic CTLK and ATLK, the epistemic variant of alternating time temporal logic. It will also cover ordered-binary decision diagrams and symbolic labeling, parameter synthesis and strategy synthesis in ATLK specifications, verification of systems with an unbounded number of agents, and MCMAS: a model checker for multiagent systems implementing these techniques, as well as the application of the techniques to a range of scenarios including in service-oriented computing.

Alessio Lomuscio is a professor in logic for multiagent systems in the Department of Computing, Imperial College London, where he leads the Verification of Autonomous Systems research group. He currently holds an EPSRC leadership fellowship. He received a PhD in computer science from the University of Birmingham in 1999 and a Laurea in electronic engineering from Politecnico di Milano in 1995. Before joining Imperial he was a lecturer at King's College London and a senior lecturer at University College London. His research interests concern the specification and verification of multiagent systems by means of techniques based on computational logic. In particular he has made theoretical contributions in the area of logic for multiagent systems (including studying the completeness, decidability and complexity of several temporal-epistemic formalisms) and has put forward several symbolic model checking techniques for the verification of agent-based systems. He has applied these techniques to the verification of a range of applications including autonomous underwater vehicles, webservices and security protocols including e-voting.

Dynamic Epistemic Logic and Its Interaction with Knowledge Representation

Lawrence S. Moss, Indiana University

Dynamic epistemic logic (DEL) takes the classical topic epistemic logic and adds features from propositional dynamic logic (PDL). DEL comes to life in the multiagent setting. It allows one to represent epistemic actions in addition to epistemic situations. DEL has been around for about 20 years, and so it has seen a lot of progress. By now, the original topics have been applied, generalized, and adapted in various ways. My primary goal in this tutorial is to present an overview of the area to people at KR 2014. The talk will discuss recent work in KR that uses DEL. After this, I would like to ask why it is that tools from DEL are not more used in the KR community. Turning things around, I also ask what influence issues from KR and other fields will have on the future of DEL and related frameworks.

Lawrence S. Moss is director of Indiana University's program in pure and applied logic. He is a professor of mathematics, and an adjunct professor of computer science, informatics, linguistics, and philosophy. For a number of years he was the steering committee chair of the North American Summer School in Logic, Language, and Information (NASSLLI).

Natural Language Understanding with World Knowledge and Inference

Ekaterina Ovchinnikova, University of Southern California

In order to understand natural language expressions it is usually not enough to know the literal (dictionary) meaning of words used in these expressions and compositional rules of the corresponding language. Much more knowledge is involved here; knowledge, which may have nothing to do with the linguistic competence but is rather related to our general conception of the world. In this tutorial, I will present an overview of the approaches to modeling knowledge-intensive natural language understanding (NLU) in a computational framework, with the main focus on inference-based NLU. I will discuss the types of knowledge required for NLU and the techniques to obtain this knowledge in a machine-readable form. The tutorial will cover lexical-semantic knowledge bases, ontologies (general and domain-specific), and corpora-based resources. Then I will focus on reasoning procedures applicable to NLU and compare two main forms of logical inference: deduction and abduction. In the last part of the tutorial, I will present applications of some of the recent large-scale inference-based NLU systems to such knowledge-intensive tasks as recognizing textual entailment, metaphor interpretation, and narrativization of videos.

*Ekaterina Ovchinnikova i*s a computer scientist (faculty) at the Institute of Information Sciences, University of Southern California. Her research interests include computational natural language understanding, human and automated reasoning, knowledge representation and extraction, scene recognition, and ontologies. Applications she is currently working on include metaphor interpretation, narrativization of videos, textual entailment recognition, and other knowledge intensive natural language understanding tasks. In 2011, Ekaterina earned her PhD degree in cognitive science from the Institute of Cognitive Science, University of Osnabrück (Germany). In her dissertation project, Ekaterina was focusing on inference-based natural language understanding. The main results of this work were published in *Integration of World Knowledge for Natural Language Understanding.*

Query Answering and Rewriting in Ontology-Based Data Access

Riccardo Rosati, Sapienza University of Rome

Ontology-based data access (OBDA) has recently become a hot research topic in knowledge representation and data management. The basic idea underlying OBDA is to superimpose an ontology (conceptual layer) to a set of data sources (data layer), and use the ontology as a virtual schema for querying the data sources. The ontology and the data sources are connected through declarative mappings that provide the semantic relationship between the two layers. This tutorial will introduce ontology-based data access and will deal with the problem of query answering in OBDA, focusing in particular on the query rewriting approach to query answering. After a brief history of ontology-based data access, the tutorial will overview the most important results obtained in this area. Then, the main query rewriting techniques for OBDA will be presented and compared. The relationship between OBDA and other research topics will also be addressed. The main focus of the tutorial is on ontologies expressed using Description Logics: however, other ontology specification languages, like Datalog+/–:, will be considered.

Riccardo Rosati is an associate professor in the Department of Computer, Control, and Management Engineering, Sapienza University of Rome, Italy. His main research interests are in the fields of databases and artificial intelligence, and include knowledge representation, description logics, ontologies, semantic web, databases, and nonmonotonic reasoning. He is the author of more than 150 publications in international journals and conferences. He is a member of the steering committees of the Description Logics workshop, the Nonmonotonic Reasoning workshop, and the International Conference on Web Reasoning and Rule Systems.

Sponsors

The KR conference has been partially supported by

Artificial Intelligence Journal

European Coordinating Committee for Artificial Intelligence (ECCAI)

Principles of Knowledge Representation and Reasoning, Incorporated (KR Inc.)

Siemens AG Österreich

Association for Logic Programming (ALP)

IOS Press

Kurt Gödel Society



Long Papers

Logic Programs with Ordered Disjunction: First-Order Semantics and Expressiveness

Vernon Asuncion[1,2], **Yan Zhang**[1], **Heng Zhang**[1]
[1] School of Computing, Engineering and Mathematics, University of Western Sydney, Australia
[2] Department of Computer Science and Engineering, Qatar University, Qatar

Abstract

Logic programs with ordered disjunction (LPODs) (Brewka 2002) generalize normal logic programs by combining alternative and ranked options in the heads of rules. It has been showed that LPODs are useful in a number of areas including game theory, policy languages, planning and argumentations. In this paper, we extend propositional LPODs to the first-order case, where a classical second-order formula is defined to capture the stable model semantics of the underlying first-order LPODs. We then develop a progression semantics that is equivalent to the stable model semantics but naturally represents the reasoning procedure of LPODs. We show that on finite structures, every LPOD can be translated to a first-order sentence, which provides a basis for computing stable models of LPODs. We further study the complexity and expressiveness of LPODs and prove that almost positive LPODs precisely capture first-order normal logic programs, which indicates that ordered disjunction itself and constraints are sufficient to represent negation as failure.

Introduction

Logic programs with ordered disjunction (LPODs) generalize normal logic programs by combining features of qualitative choice logic so that alternative and ranked options may be explicitly expressed in the heads of rules (Brewka 2002; Brewka, Benferhat, and Berre 2002). A LPOD contains a finite set of rules of the form

$$\alpha_1 \times \cdots \times \alpha_k \leftarrow \beta_1, \cdots, \beta_l, \mathsf{not}\,\gamma_1, \cdots, \mathsf{not}\,\gamma_m, \quad (1)$$

where $\alpha_i, \beta_j, \gamma_h$ ($1 \leq i \leq k, 1 \leq j \leq l, 1 \leq h \leq m$) are propositional atoms. Intuitively, rule (1) says that when the body is satisfied, then whenever it is possible, derive α_1, otherwise if possible, derive α_2, and so on. The semantics of an LPOD is defined in terms of the stable models of so-called *split programs* of the underlying LPOD.

Let us consider a simple program Π_1 from (Brewka 2002):

$$A \times B \leftarrow \mathsf{not}\, C,$$
$$B \times C \leftarrow \mathsf{not}\, D.$$

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Π_1 has the following four split programs:

$$A \leftarrow \mathsf{not}\, C \qquad A \leftarrow \mathsf{not}\, C$$
$$B \leftarrow \mathsf{not}\, D \qquad C \leftarrow \mathsf{not}\, D, \mathsf{not}\, B$$

$$B \leftarrow \mathsf{not}\, C, \mathsf{not}\, A \qquad B \leftarrow \mathsf{not}\, C, \mathsf{not}\, A$$
$$B \leftarrow \mathsf{not}\, D \qquad C \leftarrow \mathsf{not}\, D, \mathsf{not}\, B.$$

Then the class of stable models of Π consists of all stable models of these four split programs, which is $\{\{A, B\}, \{B\}, \{C\}\}$. Then by integrating proper preference relation among these stable models, the preferred stable models can be obtained for an LPOD.

There have been several extensions of LPODs in recent years: Karger *et al* (2008) extended LPODs by allowing both ordered and unordered disjunction in the heads of rules; Confalonieri *et al* (2010) recently defined a possibilistic semantics for LPODs in order to handle uncertainty; and Cabalar (2011) also proposed a direct translation from LPODs to normal logic programs via the logic of Here-and-There. It has been argued that LPODs provide a natural way to deal with preference in reasoning that are useful in various applications such as game theory, policy languages, planning and argumentations (Brewka 2002; Cabalar 2011; Confalonieri et al. 2010).

On the other hand, in recent years, Answer Set Programming (ASP) has been generalized to arbitrary first-order sentences (Ferraris, Lee, and Lifschitz 2011). One challenging research along this direction is to establish proper logical and computational foundations for promoting useful functionalities in existing ASP paradigm to the first-order level. A number of topics in this aspect have been investigated and relevant properties revealed, e.g., (Asuncion et al. 2012; Asuncion, Zhang, and Zhou 2013; Lee and Meng 2011; Babb and Lee 2012). One major advantage of first-order ASP is that it provides a succinct declarative language, in which the underlying problem constraints (rules) may be completely separated from concrete problem instances, and hence more flexible for problem representation and modeling (Lin and Zhou 2011).

In this paper, we study the semantics and expressiveness of LPODs on the first-order level. We make the following main contributions towards this topic:

1. Following the style of general stable model semantics

(Ferraris, Lee, and Lifschitz 2011), we define the stable model semantics for first-order LPODs via a classical second-order sentence, and show that this semantics is a correct lifting of Brewka's original LPODs semantics to the first-order level. We then develop a progression semantics that is equivalent to the second-order based semantics but naturally represents the reasoning procedure of LPODs.

2. We propose a translation from LPODs to first-order sentences on finite structures. We argue that this translation actually may be used as a computational basis for developing an LPOD solver.
3. We address the complexity and expressiveness issues of LPODs. We show that LPODs data complexity is NP-complete, that remains true even for positive LPODs. We also prove that almost positive LPODs precisely capture the full class of first-order normal logic programs, which indicates that ordered disjunction itself and constraints are sufficient to represent negation as failure.
4. We further consider preferred LPODs, and provide a logic characterization of such preference semantics, which again, generalizes the corresponding preferred propositional LPODs.

The rest of this paper is organized as follows. In section 2 we focus on the syntax and semantics of first-order LPODs. In section 3 we then present a progression semantics for first-order LPODs, which we believe is more intuitive to reflect the underlying derivation process embedded in LPODs. Considering to compute stable models of an LPOD, in section 4 we propose a translation from LPODs to first-order sentences on finite structures, so that the problem of computing stable models of an LPOD can be viewed as a SAT solving problem. We study the complexity and expressiveness issues in section 5. In section 6 we define the preferred stable models for LPODs and provide a logic formulation for that. Finally, we conclude the paper with some remarks in section 7.

First-order LPODs: Syntax and Semantics

We consider a second-order logic but without function symbols (i.e., functions of arities not greater than 0). A *signature* τ is a set of symbols of the form $\{c_1, \ldots, c_m, P_1, \ldots, P_n\}$ such that c_i (for $1 \leq i \leq m$) are *constant* symbols and P_j (for $1 \leq j \leq n$) are *predicate* symbols. A *structure* $\mathcal{M}$ of signature τ (or a τ-structure) is a tuple of the form

$$(M, c_1^{\mathcal{M}}, \ldots, c_m^{\mathcal{M}}, P_1^{\mathcal{M}}, \ldots, P_n^{\mathcal{M}}), \tag{2}$$

where M is the *domain* of $\mathcal{M}$, which we will mostly refer to as $Dom(\mathcal{M})$; $c_i^{\mathcal{M}}$ and $P_j^{\mathcal{M}}$ are the respective interpretations of the constant and predicate symbols.

Let $\mathcal{M}$ be a τ-structure and $\mathcal{M}'$ a τ'-structure such that $\tau' \subseteq \tau$ and where (1) $Dom(\mathcal{M}') = Dom(\mathcal{M})$; (2) $c^{\mathcal{M}'} = c^{\mathcal{M}}$ for each constant symbol c of τ'; and (3) $P^{\mathcal{M}'} = P^{\mathcal{M}}$ for each predicate symbol P of τ'. We refer $\mathcal{M}'$ as the *restriction* of $\mathcal{M}$ to the signature of τ', which we denote by $\mathcal{M}\restriction_{\tau'}$, i.e., $\mathcal{M}' = \mathcal{M}\restriction_{\tau'}$. Symmetrically, $\mathcal{M}$ would be called an *expansion* of $\mathcal{M}'$ to the signature τ.

The Syntax

An *ordered disjunction rule* is a construct of the form:

$$\alpha_1 \times \ldots \times \alpha_k \leftarrow \beta_1, \ldots, \beta_l, \mathsf{not}\, \gamma_1, \ldots, \mathsf{not}\, \gamma_m, \tag{3}$$

where:

- Each α_i (for $1 \leq i \leq k$) is a standard atom $P(\mathbf{x})$ for some predicate P and tuple of terms (variables or constants) $\mathbf{x}$. If $k = 0$, then (3) is called a *constraint*.
- Each β_i ($1 \leq i \leq l$) and γ_i ($1 \leq i \leq m$) are either a standard or an equality atom.
- By $Head(r)$, we denote the ordered expression $\alpha_1 \times \ldots \times \alpha_k$, that corresponds to the head of r. We also denote $Head(r)_i = \alpha_i$.
- As usual, by $Body(r)$, we denote the set of literals $\{\beta_1, \ldots, \beta_l, \mathsf{not}\, \gamma_1, \ldots, \mathsf{not}\, \gamma_m\}$.
- Similarly, by $Pos(r)$ and $Neg(r)$, we denote the sets of atoms $\{\beta_1, \ldots, \beta_l\}$ and $\{\gamma_1, \ldots, \gamma_m\}$, respectively.

A *first-order logic program with ordered disjunction*, or simply called (first-order) LPOD, Π, is a finite set of ordered disjunction rules of the form (3). If for all rules $r \in \Pi$ we have that $k \leq 1$ in $Head(r) = \alpha_1 \times \ldots \times \alpha_k$, then Π becomes a normal logic program. It is obvious that this first-order LPOD syntax naturally extends Brewka's original propositional logic programs with ordered disjunction (Brewka 2002) to the first-order case.

For a given LPOD Π, a predicate is called *intensional* if it occurs at least once in the head of some rule in Π. All other predicates are called *extensional*. For convenience, we denote by $\tau(\Pi)$, $\mathcal{P}_{ext}(\Pi)$ and $\mathcal{P}_{int}(\Pi)$, the signature of Π, the set of extensional predicates and the set of intensional predicates, respectively. Furthermore, we also use $\tau_{ext}(\Pi)$ and $\tau_{int}(\Pi)$ to denote the sub-signatures of $\tau(\Pi)$ which only contain extensional and intensional predicates respectively, together with all constant symbols of $\tau(\Pi)$. When there is no confusion in the context, we often omit the parameter Π in these notions.

The Stable Model Semantics

For a rule of the form (3), we denote by $\widehat{Body(r)}$, the formula $\beta_1 \wedge \cdots \wedge \beta_l \wedge \neg\gamma_1 \wedge \cdots \wedge \neg\gamma_m$, and $Head(r)^{FO}$, the following formula

$$\alpha_1 \vee (\neg\alpha_1 \wedge \alpha_2) \vee (\neg\alpha_1 \wedge \neg\alpha_2 \wedge \alpha_3) \vee \ldots \vee (\neg\alpha_1 \wedge \neg\alpha_2 \wedge \ldots \wedge \neg\alpha_{k-1} \wedge \alpha_k).$$

Note that Cabalar (2011) has showed the strong equivalence of $\alpha_1 \times \cdots \times \alpha_k$ and $Head(r)^{FO}$ for the propositional case, that can be easily extended to our first-order case. Then for an LPOD Π, by $\widehat{\Pi}$, we denote the conjunction

$$\bigwedge_{r \in \Pi} (\widehat{Body(r)} \rightarrow Head(r)^{FO}). \tag{4}$$

Definition 1 *Let Π be an LPOD with tuple of distinct (intensional) predicates $\mathbf{P} = P_1 P_2 \ldots P_n$ and $\mathcal{M}$ a $\tau(\Pi)$-structure. Then we say that $\mathcal{M}$ is a* stable model *of Π iff it is a model of the following second-order sentence:*

$$\mathrm{SM}_{\mathbf{P}}(\Pi) = \widehat{\Pi} \wedge \neg\exists \mathbf{U}(\mathbf{U} < \mathbf{P} \wedge \widehat{\Pi}(\mathbf{U})^*), \tag{5}$$

where:

- $\mathbf{U} = U_1 U_2 \ldots U_n$ *is a fresh tuple of distinct predicates matching* $\mathbf{P}$*;*
- $\mathbf{U} < \mathbf{P}$ *denotes the formula*

$$\bigwedge_{1 \le i \le n} \forall \mathbf{x}(U_i(\mathbf{x}) \rightarrow P_i(\mathbf{x})) \wedge \neg \bigwedge_{1 \le i \le n} \forall \mathbf{x}(P_i(\mathbf{x}) \rightarrow U_i(\mathbf{x}));$$

- $\widehat{\Pi}(\mathbf{U})^*$ *denotes the formula*

$$\bigwedge_{r \in \Pi, \atop r \text{ of form the (3)}} \forall \mathbf{x}_r(\, \beta_1^* \wedge \ldots \wedge \beta_l^* \wedge \neg\gamma_1 \wedge \ldots \wedge \neg\gamma_m \rightarrow (\alpha_1 \times \ldots \times \alpha_k)^*\,),$$

where $(\alpha_1 \times \ldots \times \alpha_k)^*$ *denotes the formula:*

$$\alpha_1^* \vee (\neg\alpha_1 \wedge \alpha_2^*) \vee (\neg\alpha_1 \wedge \neg\alpha_2 \wedge \alpha_3^*) \vee \ldots \vee (\neg\alpha_1 \wedge \neg\alpha_2 \wedge \ldots \wedge \neg\alpha_{k-1} \wedge \alpha_k^*), \quad (6)$$

and such that $P_i(\mathbf{x})^* = U_i(\mathbf{x})$ *for any atomic formula* $P_i(\mathbf{x})$ *with* $1 \le i \le n$*, and* $Q(\mathbf{y})^* = Q(\mathbf{y})$ *otherwise.*

It is not difficult to see that Definition 1 is a counterpart of the second-order definition for the general stable model semantics (Ferraris, Lee, and Lifschitz 2011), with the additional operation (6). In the following, we will show that Definition 1 is also a generalization of the semantics for the original propositional LPODs, and hence, our first-order LPOD stable model semantics is a proper uplifting from the propositional level.

Let τ be a signature of Π with the sets $\mathcal{P}_{ext}$ and $\mathcal{P}_{int}$ of extensional and intensional predicates respectively, and let $\mathcal{M}$ be a τ-structure. Then we define $I_{\mathcal{M}}$ as follows:

$$I_{\mathcal{M}} := \{P(\mathbf{a}) \mid P \in (\mathcal{P}_{ext} \cup \mathcal{P}_{int}) \text{ and } \mathbf{a} \in P^{\mathcal{M}}\},$$

i.e., the set of propositional atoms corresponding to the extents of the predicates of τ under $\mathcal{M}$. In addition, by $Ext_{\mathcal{M}}$, define the set of rules (which are facts) as follows:

$$Ext_{\mathcal{M}} := \{Q(\mathbf{a}) \;\leftarrow \mid Q \in \mathcal{P}_{ext} \text{ and } \mathbf{a} \in Q^{\mathcal{M}}\},$$

i.e., the set of "facts" corresponding to the extensional database under $\mathcal{M}$.

In addition to the two previous notions, we now introduce the notion of a grounded program. Let Π be an LPOD and $\mathcal{M}$ a $\tau(\Pi)$-structure with domain denoted by $Dom(\mathcal{M})$. Then by $\Pi\,|_{Dom(\mathcal{M})}$, let us denote the propositional program obtained from Π by (1) simultaneously replacing the variables by domain elements from $Dom(\mathcal{M})$ in all possible ways, and the constant symbols by their corresponding interpretation in $Dom(\mathcal{M})$ under $\mathcal{M}$; and (2) removing all rules that contain false instantiations of equality predicates and deleting those true instantiations of equalities from the remaining rules.

Before we present the following result, we also need a formal definition of split programs for a given propositional LPOD. Let Π be a propositional LPOD consisting of rules of the form of (3), but all those α's, β's and γ's are propositional atoms. Then a *split program* of Π is a normal logic program Π' obtained from Π by replacing each rule of (3) in Π, for some $i \le k$, by

$$\alpha_i \leftarrow \beta_1, \ldots, \beta_l, \mathsf{not}\ \gamma_1, \ldots, \mathsf{not}\ \gamma_m, \mathsf{not}\ \alpha_1, \cdots, \mathsf{not}\ \alpha_{i-1}. \quad (7)$$

Theorem 1 *Let* Π *be an LPOD and* $\mathcal{M}$ *a* $\tau(\Pi)$*-structure. Then* $\mathcal{M}$ *is a stable model of* Π *iff* $I_{\mathcal{M}}$ *is a stable model of some split program of* $\Pi|_{Dom(\mathcal{M})} \cup Ext_{\mathcal{M}}$*.*

Progression Semantics: An Alternative

According to Definition 1, the semantics of first-order LPODs is defined via a second-order sentence, which hardly reveals the rule based reasoning feature of the underlying LPOD. In this section, we provide an alternative semantics for FO LPODs - the progression semantics, which is an extension of Zhang and Zhou's progression semantics for first-order normal logic program (Zhang and Zhou 2010). As we will see, the important feature of such semantics is that it naturally represents the reasoning procedure through the progression stages.

Definition 2 *Let* Π *be a FO LPOD and* $\mathcal{M}$ *a* $\tau(\Pi)$*-structure*

$$(Dom(\mathcal{M}), c_1^{\mathcal{M}}, \ldots, c_r^{\mathcal{M}}, Q_1^{\mathcal{M}}, \ldots, Q_s^{\mathcal{M}}, P_1^{\mathcal{M}}, \ldots, P_n^{\mathcal{M}})$$

such that c_i *(for* $1 \le i \le r$*),* Q_i *(for* $1 \le i \le s$*), and* P_i *(for* $1 \le i \le n$*) are its constant symbols, extensional and intensional predicate symbols, respectively. We define the* $\tau(\Pi)$*-structure* $\mathcal{M}^{t+1}(\Pi)$ *inductively as follows:*

$$\mathcal{M}^0(\Pi) = (Dom(\mathcal{M}), c_1^{\mathcal{M}^0(\Pi)}, \ldots, c_r^{\mathcal{M}^0(\Pi)}, Q_1^{\mathcal{M}^0(\Pi)}, \ldots, Q_s^{\mathcal{M}^0(\Pi)}, P_1^{\mathcal{M}^0(\Pi)}, \ldots, P_n^{\mathcal{M}^0(\Pi)}), \quad (8)$$

where $c_i^{\mathcal{M}^0(\Pi)} = c_1^{\mathcal{M}}$ *(for* $1 \le i \le r$*),* $Q_i^{\mathcal{M}^0(\Pi)} = Q_i^{\mathcal{M}}$ *(for* $1 \le i \le s$*), and* $P_i^{\mathcal{M}^0(\Pi)} = \emptyset$ *(for* $1 \le i \le n$*);*

$$\mathcal{M}^{t+1}(\Pi) = \mathcal{M}^t(\Pi) \cup \{\alpha_i\eta \mid \exists r \in \Pi \textit{ with } Head(r) = \alpha_1 \times \ldots \times \alpha_i \times \ldots \times \alpha_k \textit{ and } \eta \textit{ an assignment such that:}$$

1. $Pos(r)\eta \subseteq \mathcal{M}^t(\Pi)$ *and* $Neg(r)\eta \cap \mathcal{M} = \emptyset$*;*

2. i *(for* $1 \le i \le k$*) is the largest* i *such that* $\{\alpha_1\eta, \ldots, \alpha_{i-1}\eta\} \cap \mathcal{M} = \emptyset\}$. (9)

Then let $\mathcal{M}^{\infty}(\Pi) = \bigcup_{0 \le t \le \infty} \mathcal{M}^t(\Pi)$.

Since $\mathcal{M}^t(\Pi) \subseteq \mathcal{M}^{t+1}(\Pi)$, we know that the sequence $\mathcal{M}^0, \mathcal{M}^1, \cdots$ is monotonic and $\mathcal{M}^t(\Pi)$ always converges to its fixpoint. Also note that when $Head(r) = \alpha_1$ (i.e., r is a normal rule), then we will have in Item 2 in (9) that $i = 1$ will automatically be the largest i such that $\{\alpha_1\eta, \ldots, \alpha_{i-1}\eta\} \cap \mathcal{M} = \emptyset$. From this fact, the following proposition shows that the progression characterization captures the stable models when Π is a normal logic program.

Proposition 1 *Let* Π *be a normal logic program and* $\mathcal{M}$ *a* $\tau(\Pi)$*-structure. Then* $\mathcal{M}$ *is a stable model of* Π *iff* $\mathcal{M}^{\infty}(\Pi) = \mathcal{M}$*.*

Proof: If Π is a normal program, then we will have by default in Item 2 in (9) of Definition 2 that $i = 1$ will be the largest i such that $\{\alpha_1\eta, \ldots, \alpha_{i-1}\eta\} \cap \mathcal{M} = \emptyset$, since $\{\alpha_1, \ldots, \alpha_{i-1}\} = \emptyset$ in this case. Therefore, we can omit Item 2 in (9) so that $\mathcal{M}^{t+1}(\Pi)$ will be as originally defined in (Zhang and Zhou 2010) for normal programs. □

Theorem 2 shows that the progression semantics coincides with the second-order stable model semantics of LPODs as defined in Definition 1. We first present the following lemma which is needed in the proof of Theorem 2.

Lemma 1 *If $\mathcal{M}^\infty(\Pi) = \mathcal{M}$ then $\mathcal{M} \models \widehat{\Pi}$.*

Theorem 2 *Let Π be an LPOD and $\mathcal{M}$ a $\tau(\Pi)$-structure. Then $\mathcal{M}$ is a stable model of Π iff $\mathcal{M}^\infty(\Pi) = \mathcal{M}$.*

Proof: ($\Rightarrow$) First we show that $\mathcal{M}^\infty \subseteq \mathcal{M}$ by showing $\mathcal{M}^t \subseteq \mathcal{M}$ holds for all $t \geq 0$ by induction. Clearly, $\mathcal{M}^0(\Pi) \subseteq \mathcal{M}$ holds by the definition of $\mathcal{M}^0(\Pi)$ since all the intensional relations are set to empty. Now assume that $\mathcal{M}^{t'}(\Pi) \subseteq \mathcal{M}$ holds for all $0 \leq t' \leq t$ and let $P(\mathbf{a}) \in \mathcal{M}^{t+1}(\Pi)$ such that $P(\mathbf{a}) \notin \mathcal{M}^t(\Pi)$. Then by the definition of $\mathcal{M}^{t+1}(\Pi)$, we have that there exists a rule $r \in \Pi$ with $Head(r) = \alpha_1 \times \ldots \times \alpha_i \times \ldots \times \alpha_k$ and assignment η such that:

1. $P(\mathbf{a}) = \alpha_i\eta$;
2. $Pos(r)\eta \subseteq \mathcal{M}^t(\Pi)$ and $Neg(r) \cap \mathcal{M} = \emptyset$;
3. i (for $1 \leq i \leq k$) is the largest i such that $\{\alpha_1\eta, \ldots, \alpha_{i-1}\eta\} \cap \mathcal{M} = \emptyset$.

Then since $Pos(r)\eta \subseteq \mathcal{M}^t(\Pi)$ and $Neg(r) \cap \mathcal{M} = \emptyset$, and where $\mathcal{M}^t(\Pi) \subseteq \mathcal{M}$, it follows that $\mathcal{M} \models \widehat{Body(r)}\eta$. Thus, since $\mathcal{M} \models \widehat{\Pi}$ (because $\mathcal{M} \models \mathrm{SM}_{\mathbf{P}}[\Pi]$), then it follows that $\mathcal{M} \models Head(r)^{FO}\eta \equiv (\neg\alpha_1 \wedge \ldots \wedge \neg\alpha_{i-1} \wedge \alpha_i)\eta$, which implies that $\alpha_i\eta = P(\mathbf{a}) \in \mathcal{M}$. Therefore, we had shown that $\mathcal{M}^\infty(\Pi) \subseteq \mathcal{M}$. Now let us assume on the contrary that it is also the case that $\mathcal{M}^\infty(\Pi) \subset \mathcal{M}$ and we show we derive a contradiction. Then define a $\tau(\Pi) \cup \{U_1, \ldots, U_n\}$-structure $\mathcal{U}$ as follows:

- $c^{\mathcal{U}} = c^{\mathcal{M}}$ for every constant c of $\tau(\Pi)$;
- $Q^{\mathcal{U}} = Q^{\mathcal{M}}$ for every extensional predicate of $\tau(\Pi)$;
- $P_i^{\mathcal{U}} = P_i^{\mathcal{M}}$ for every intensional predicate P_i of $\tau(\Pi)$, such that $1 \leq i \leq n$.
- $U_i^{\mathcal{U}} = P_i^{\mathcal{M}^\infty(\Pi)}$ for every predicate U_i in $\{U_1, \ldots, U_n\}$, with $1 \leq i \leq n$.

Now we will show that $\mathcal{U} \models \mathbf{U} < \mathbf{P}$ and $\mathcal{U} \models \widehat{\Pi}(\mathbf{U})^*$. The first part $\mathcal{U} \models \mathbf{U} < \mathbf{P}$ follows from the fact that $\mathcal{M}^\infty(\Pi) \subset \mathcal{M}$ and by the way the $\tau(\Pi) \cup \{U_n, \ldots, U_n\}$-structure $\mathcal{U}$ is constructed from $\mathcal{M}$ and $\mathcal{M}^\infty(\Pi)$ above. Now we show the second part. So assume for some rule r in Π of the form (3) and some assignment η that $\mathcal{U} \models \widehat{Body(r)}^*\eta$. Then since $\mathcal{U} \models \mathbf{U} < \mathbf{P}$ and by the definition of $\mathcal{U}$, it follows that $Pos(r)\eta \subseteq \mathcal{M}^\infty(\Pi)$, which implies that $Pos(r)\eta \subseteq \mathcal{M}^t(\Pi) \subseteq \mathcal{M}^\infty(\Pi) \subset \mathcal{M}$ for some $t \geq 0$. Then since $Pos(r)\eta \subset \mathcal{M}$, then $\mathcal{M} \models \widehat{Body(r)}\eta$ as well. Then since $\mathcal{M} \models \widehat{\Pi}$, we have that $\mathcal{M} \models Head(r)^{FO}\eta$, which further implies that $\mathcal{M} \models \neg\alpha_1 \wedge \ldots \wedge \neg\alpha_{i-1} \wedge \alpha_i$ for some $1 \leq i \leq k$. Then by Item 2 of (9), this implies that $\alpha_i\eta \in \mathcal{M}^{t+1}(\Pi) \subseteq \mathcal{M}^\infty(\Pi)$ since i will be the largest i such that $\{\alpha_1\eta, \ldots, \alpha_{i-1}\eta\} \cap \mathcal{M} = \emptyset$, which further implies that $\mathcal{U} \models Head(r)^*\eta$ by the way $\mathcal{U}$ was constructed from $\mathcal{M}^\infty(\Pi)$ and since $\mathcal{M}^{t+1}(\Pi) \subseteq \mathcal{M}^\infty(\Pi)$. Therefore, because we have shown that $\mathcal{U} \models \widehat{\Pi}(\mathbf{U})^*$, then we now have a contradiction since we initially assumed that $\mathcal{M} \models \mathrm{SM}_{\mathbf{P}}(\Pi)$, and where this implies that $\mathcal{M} \models \neg\exists\mathbf{U}(\mathbf{U} < \mathbf{P} \wedge \widehat{\Pi}(\mathbf{U})^*)$ (which is contradicted by the existence of the structure $\mathcal{U}$).

($\Leftarrow$) Now let us assume $\mathcal{M}^\infty(\Pi) = \mathcal{M}$ but, on the contrary that $\mathcal{M} \not\models \mathrm{SM}_{\mathbf{P}}[\Pi]$. Since by Lemma 1 we have that $\mathcal{M} \models \widehat{\Pi}$, then this implies that $\mathcal{M} \not\models \neg\exists\mathbf{U}(\mathbf{U} < \mathbf{P} \wedge \widehat{\Pi}(\mathbf{U})^*)$. Thus, assume that $\mathcal{U}$ is a $\tau(\Pi) \cup \{U_n, \ldots, U_n\}$-structure such that that $\mathcal{U} \models \mathbf{U} < \mathbf{P} \wedge \widehat{\Pi}(\mathbf{U})^*$. Then we will show by induction on t that $P_i^{\mathcal{M}^t(\Pi)} \subseteq U_i^{\mathcal{U}}$, for $1 \leq i \leq n$. Therefore, since $\mathcal{U} \models \mathbf{U} < \mathbf{P}$ (i.e., $\mathbf{U}$ is "strictly smaller" than $\mathbf{P}$), then this contradicts the assumption that $\mathcal{M}^\infty(\Pi) = \mathcal{M}$ (because $\mathcal{M}^\infty(\Pi)$ would be "strictly less" than $\mathcal{M}$ in this case). The base case for $\mathcal{M}^0(\Pi)$ clearly holds since $P_i^{\mathcal{M}^0(\Pi)} = \emptyset$, for $1 \leq i \leq n$. So let us assume that $P_i^{\mathcal{M}^t(\Pi)} \subseteq U_i^{\mathcal{U}}$, for $1 \leq i \leq n$, and consider $\mathcal{M}^{t+1}(\Pi)$. Indeed, let $P(\mathbf{a}) \in \mathcal{M}^{t+1}(\Pi) \setminus \mathcal{M}^t(\Pi)$. Then by the definition of $\mathcal{M}^{t+1}(\Pi)$ as we find in (9) of Definition (2), there exists a rule $r \in \Pi$ with $Head(r) = \alpha_1 \times \ldots \times \alpha_i \times \ldots \times \alpha_k$ such that:

1. $Pos(r)\eta \subseteq \mathcal{M}^t(\Pi)$ and $Neg(r)\eta \cap \mathcal{M} = \emptyset$;
2. i (for $1 \leq i \leq k$) is the largest i such that $\{\alpha_1\eta, \ldots, \alpha_{i-1}\eta\} \cap \mathcal{M} = \emptyset$,

and where $P_i(\mathbf{a}) = \alpha_i\eta$. Then it is not difficult to see that $\mathcal{U} \models \widehat{Body(r)}^*\eta$ as well since $P_i^{\mathcal{M}^t(\Pi)} \subseteq U_i^{\mathcal{U}}$, for $1 \leq i \leq n$. Then since $\mathcal{U} \models \widehat{\Pi}(\mathbf{U})^*$, we have that $\mathcal{U} \models Head(r)^*$, and where in particular, $\mathcal{U} \models (\neg\alpha_1 \wedge \ldots \wedge \neg\alpha_{i-1} \wedge \alpha_i^*)\eta$, so that $\alpha_i^*\eta = P(\mathbf{a}) \in \mathcal{U}$. This completes the proof of Theorem 2. □

Example 1 *Let Π be the following LPOD with the two rules:*

$$r_1 : P(x) \times Q(x) \times R(x) \leftarrow S(x), \tag{10}$$
$$r_2 : T(x) \times U(x) \leftarrow Q(x), \tag{11}$$

such that S is the only extensional predicate. Now let $\mathcal{M}$ be a $\tau(\Pi)$-structure such that:

$$\mathcal{M} = (\{a, b\}, S^{\mathcal{M}} = \{a, b\}, P^{\mathcal{M}} = \{a\}, Q^{\mathcal{M}} = \{b\}, R^{\mathcal{M}} = \emptyset, T^{\mathcal{M}} = \emptyset, U^{\mathcal{M}} = \{b\}).$$

Now based on (8) of Definition 2, we have that

$$\mathcal{M}^0(\Pi) = (\{a, b\}, S^{\mathcal{M}} = \{a, b\}, P^{\mathcal{M}} = \emptyset, Q^{\mathcal{M}} = \emptyset, R^{\mathcal{M}} = \emptyset, T^{\mathcal{M}} = \emptyset, U^{\mathcal{M}} = \emptyset),$$

i.e., all the intensional relations are initially set to empty. Now let us compute $\mathcal{M}^1(\Pi)$. From (9) of Definition 2, we have that $P(a) \in \mathcal{M}^1(\Pi)$ since $\alpha_{i=1}\eta = P(a)$ (with $\eta : x \longrightarrow a$) is the largest i such that $\{\alpha_1\eta, \ldots, \alpha_{i-1}\eta\} = \emptyset \cap \mathcal{M} = \emptyset$. Similarly, we also have that $Q(b) \in \mathcal{M}^1(\Pi)$ since in this case, with $\eta : x \longrightarrow b$, we have that $\alpha_{i=2}\eta = Q(b)$ is the largest i such that $\{\alpha_1\eta, \ldots, \alpha_{i-1}\eta\} = \{P(b)\} \cap \mathcal{M} = \emptyset$ (i.e., since $P(b) \notin \mathcal{M}$), so that we now have

$$\mathcal{M}^1(\Pi) = (\{a, b\}, S^{\mathcal{M}} = \{a, b\}, P^{\mathcal{M}} = \{a\}, Q^{\mathcal{M}} = \{b\}, R^{\mathcal{M}} = \emptyset, T^{\mathcal{M}} = \emptyset, U^{\mathcal{M}} = \emptyset).$$

Similarly, we will have $\mathcal{M}^2(\Pi)$ as follows:

$$(\{a, b\}, S^{\mathcal{M}} = \{a, b\}, P^{\mathcal{M}} = \{a\}, Q^{\mathcal{M}} = \{b\}, R^{\mathcal{M}} = \emptyset, T^{\mathcal{M}} = \emptyset, U^{\mathcal{M}} = \{b\}),$$

and further, we can obtain $\mathcal{M}^2(\Pi) = \mathcal{M}^t(\Pi)$ *for all* $t \geq 3$*, that is* $\mathcal{M}^\infty(\Pi) = \mathcal{M}^2(\Pi)$*. Therefore, since* $\mathcal{M}^\infty(\Pi) = \mathcal{M}$*, we have by Theorem 2 that* $\mathcal{M}$ *is a stable model of* Π*.* □

From LPODs to First-order Formulas

While Definition 2 provides an alternative semantics for FO LPODs, which represents the program reasoning feature through progression process, it, however, still does not reveal much information about how a stable model of a given LPOD may be computed. In this section, we show how a variant of the ordered completion (Asuncion et al. 2012) for normal logic programs can capture the stable models of first-order LPODs, which, like demonstrated in (Asuncion et al. 2012), may be viewed as a computational basis for an FO LPOD solver development.

For this purpose, for a given pair of predicates (P, Q) (where P and Q can be the same), by $\leq_{PQ}$, we denote a new predicate such that its arity is the sum of the arities of P and Q. We refer to such predicates as the *comparison predicates*. The intuitive meaning of atom $\leq_{PQ}(\mathbf{x}, \mathbf{y})$ is that: $Q(\mathbf{y})$ is true only if $P(\mathbf{x})$ is true.

Definition 3 *Let* Π *be an LPOD with tuple of distinct (intensional) predicates* $\mathbf{P}$*. Then by* $\text{MCOMP}_{\mathbf{P}}(\Pi)$*, we denote the following first-order sentence:*

$$\bigwedge_{r \in \Pi} \forall \mathbf{x}_r (\, \widehat{Body(r)} \rightarrow Head(r)^{FO}\,) \tag{12}$$

$$\wedge \bigwedge_{P \in \mathbf{P}} \forall \mathbf{x} (\, P(\mathbf{x}) \rightarrow \bigvee_{\substack{r \in \Pi, \\ Head(r)_i = P(\mathbf{y}), \\ 1 \leq i \leq k}} \exists \mathbf{x}_r (\mathbf{x} = \mathbf{y} \wedge \widehat{Body(r)} \wedge \widehat{Pos(r) < P}(\mathbf{x}) \wedge \bigwedge_{1 \leq j \leq i-1} \neg Head(r)_j)\,), \tag{13}$$

where:

- *For a rule* $r \in \Pi$*,* $\mathbf{x}_r$ *in (12) and (13) is the tuple of distinct variables of* r*;*
- *We assume that* $\mathbf{x}$ *in (13) is a tuple of fresh distinct variables not mentioning those from* $\mathbf{x}_r$*;*
- *For two tuples* $\mathbf{x} = x_1 x_2 \ldots x_s$ *and* $\mathbf{y} = y_1 y_2 \ldots y_s$*,* $\mathbf{x} = \mathbf{y}$ *in (13) denotes the formula* $(x_1 = y_1) \wedge (x_2 = y_2) \wedge \ldots \wedge (x_s = y_s)$*;*
- $\widehat{Pos(r) < P}(\mathbf{x})$ *in (13) denotes the formula*

$$\bigwedge_{\substack{Q(\mathbf{z}) \in Pos(r), \\ Q \in \mathbf{P}}} (\leq_{QP}(\mathbf{z}, \mathbf{x}) \wedge \neg \leq_{PQ}(\mathbf{x}, \mathbf{z})),$$

where the "$\leq_{QP}$*" and "*$\leq_{PQ}$*" here are comparison predicates.*

Then finally, we define

$$\text{OC}_{\mathbf{P}}(\Pi) = \text{MCOMP}_{\mathbf{P}}(\Pi) \wedge \text{TRANS}_{\mathbf{P}}(\Pi),$$

where $\text{TRANS}_{\mathbf{P}}(\Pi)$ *denotes the following formula:*

$$\bigwedge_{P,Q,R \in \mathbf{P}} \forall \mathbf{xyz} (\leq_{PQ}(\mathbf{x}, \mathbf{y}) \wedge \leq_{QR}(\mathbf{y}, \mathbf{z}) \rightarrow \leq_{PR}(\mathbf{x}, \mathbf{z})), \tag{14}$$

and each of $\mathbf{x}$*,* $\mathbf{y}$*, and* $\mathbf{z}$ *are tuples of distinct variables.*

Now we present the main result of this section as follows.

Theorem 3 *Let* Π *be an LPOD with tuple of distinct intensional predicates* $\mathbf{P}$ *and* $\mathcal{M}$ *a finite* $\tau(\Pi)$*-structure. Then* $\mathcal{M}$ *is a stable model of* Π *iff it can be expanded to a model of* $\text{OC}_{\mathbf{P}}(\Pi)$*.*

Proof: ($\Rightarrow$) Then $\mathcal{M} \models \widehat{\Pi}$, so that $\mathcal{M}$ is also a model of (12). Therefore, it is now sufficient to show that $\mathcal{M}$ can be expanded to a model of (13) and (14). Indeed, by Theorem 2, we have that $\mathcal{M}^\infty(\Pi) = \mathcal{M}$ as well since $\mathcal{M}$ is a stable model of Π (by assumption). Then based on the progression stages $\mathcal{M}^1(\Pi)$, $\mathcal{M}^2(\Pi)$, …, $\mathcal{M}^k(\Pi)$, $\mathcal{M}^{k+1}(\Pi)$, …, we now construct an expansion $\mathcal{M}'$ of $\mathcal{M}$ to the signature $\tau(\Pi) \cup \{\leq_{PQ} |\ P, Q \in \mathbf{P}\}$. Thus, for $t \geq 0$, set $\Delta^t(\Pi)$ as the $\tau(\Pi)$-structure defined inductively as follows: $\Delta^0(\Pi) = \mathcal{M}^0(\Pi)$ and where for $t \geq 1$, $\Delta^t(\Pi) = \mathcal{M}^t(\Pi) \setminus \mathcal{M}^{t-1}(\Pi)$. Then intuitively speaking, for $t \geq 1$, $\Delta^t(\Pi)$ represents the difference of the intensional relations between the stages t and $t-1$. Then we can prove the following Claim 1.

Claim 1. *If* $P(\mathbf{a}) \in \Delta^t(\Pi)$ *then for all* $t' \neq t$*, we have that* $P(\mathbf{a}) \notin \Delta^{t'}(\Pi)$*.*

Now the expansion $\mathcal{M}'$ of $\mathcal{M}$ is constructed by setting:

$$\leq_{PQ}^{\mathcal{M}'} = \{\mathbf{ab} \mid P(\mathbf{a}) \in \Delta^{t_1}(\Pi), Q(\mathbf{b}) \in \Delta^{t_2}(\Pi), \text{ and } t_1 < t_2\},$$

for each pair of predicates P and Q (can be the same) of $\mathbf{P}$. Now we show that $\mathcal{M}'$ satisfies both (13) and (14). Indeed, let $\mathbf{ab} \in \leq_{PQ}^{\mathcal{M}'}$ and $\mathbf{bc} \in \leq_{QR}^{\mathcal{M}'}$ for some predicates P, Q, and R of $\mathbf{P}$. Then by the definitions of the interpretations $\leq_{PQ}^{\mathcal{M}'}$ and $\leq_{QR}^{\mathcal{M}'}$, we have that: $P(\mathbf{a}) \in \Delta^{t_1}(\Pi)$ and $Q(\mathbf{b}) \in \Delta^{t_2}(\Pi)$ with $t_1 < t_2$; and $Q(\mathbf{b}) \in \Delta^{t_1'}(\Pi)$ and $R(\mathbf{c}) \in \Delta^{t_2'}(\Pi)$ with $t_1' < t_2'$. Then since by Claim 1, $Q(\mathbf{b})$ can only be in one particular t such that it is in $\Delta^t(\Pi)$, then we have that $t_2 = t_1'$, so that by transitivity, we have $t_1 < t_2'$. Then by the definition of the interpretation $\leq_{PR}^{\mathcal{M}'}$, we have that $\mathbf{ac} \in \leq_{PR}^{\mathcal{M}'}$, so that $\mathcal{M}'$ satisfies (14). Now to show $\mathcal{M}'$ satisfies (13), let $P(\mathbf{a}) \in \mathcal{M}'$ such that $P \in \mathbf{P}$. Then $P(\mathbf{a}) \in \mathcal{M}^\infty(\Pi)$ since $\mathcal{M}$ is a stable model of Π and $\mathcal{M}^\infty(\Pi) = \mathcal{M}$ by Theorem 2. Then for some t, we have that $P(\mathbf{a}) \in \mathcal{M}^t(\Pi)$. Let us assume without loss of generality that $P(\mathbf{a}) \notin \mathcal{M}^{t'}(\Pi)$ for all $0 \leq t' < t$, i.e., t is the first stage that derives $P(\mathbf{a})$. Then by the definition of $\mathcal{M}^t(\Pi)$, there exists some rule $r \in \Pi$ with $Head(r) = \alpha_1 \times \ldots \times \alpha_i \times \ldots \times \alpha_k$ and assignment η such that:

1. $Pos(r)\eta \subseteq \mathcal{M}^{t-1}(\Pi) \subseteq \mathcal{M}$ (and where $\mathcal{M}^{t-1}(\Pi) \subseteq \mathcal{M}$ since $\mathcal{M}^{t-1}(\Pi) \subseteq \mathcal{M}^\infty(\Pi) = \mathcal{M}$) and $Neg(r)\eta \cap \mathcal{M} = \emptyset$;
2. i (for $1 \leq i \leq k$) is the largest i such that $\{\alpha_1\eta, \ldots, \alpha_{i-1}\eta\} \cap \mathcal{M} = \emptyset$.

Then since $\mathcal{M}'$ is simply the expansion of $\mathcal{M}$ to include the interpretation of the comparison symbols in $\{\leq_{PQ} |\ P, Q \in \mathbf{P}\}$, we have that

$$\mathcal{M}' \models (\mathbf{a} = \mathbf{y} \wedge \widehat{Body(r)} \wedge \bigwedge_{1 \leq j \leq i-1} \neg \alpha_j)\eta.$$

Therefore, it is only left to show that $\mathcal{M}' \models (\widehat{Pos(r) < P}(\mathbf{a}))\eta$ so that

$$\mathcal{M}' \models (\mathbf{a} = \mathbf{y} \wedge \widehat{Body(r)} \wedge Pos(\widehat{r) < P}(\mathbf{a}) \wedge \bigwedge_{1 \leq j \leq i-1} \neg\alpha_j)\eta \tag{15}$$

as well. Indeed, since $\mathcal{M}^t(\Pi)$ is the first stage that derives $P(\mathbf{a})$, then by the definition of $\Delta^t(\Pi)$, we have that $P(\mathbf{a}) \in \Delta^t(\Pi)$. Now let $Q(\mathbf{b}) \in Pos(r)\eta$ such that $Q \in \mathbf{P}$. Then since $Q(\mathbf{b}) \in \mathcal{M}^{t-1}(\Pi)$, there exists the least stage $t' \leq t-1$ such that $Q(\mathbf{b}) \in \mathcal{M}^{t'}(\Pi)$. Then this implies that $Q(\mathbf{b}) \in \Delta^{t'}(\Pi)$, and since $t' \leq t-1 < t$, we further have by the definition of the interpretation $\leq_{QP}^{\mathcal{M}'}$ that $\mathbf{ba} \in \leq_{QP}^{\mathcal{M}'}$. Now we show that $\mathbf{ab} \notin \leq_{PQ}^{\mathcal{M}'}$. Otherwise, assume that $\mathbf{ab} \in \leq_{PQ}^{\mathcal{M}'}$. Then by the definition of $\leq_{PQ}^{\mathcal{M}'}$, there exists t_1 and t_2, where $t_1 < t_2$, such that $P(\mathbf{a}) \in \Delta^{t_1}(\Pi)$ and $Q(\mathbf{b}) \in \Delta^{t_2}(\Pi)$. Then by Claim 1, we have that $t_2 = t'$ since $Q(\mathbf{b})$ is both in $\Delta^{t_2}(\Pi)$ and $\Delta^{t'}(\Pi)$. Then because we already have that $t' < t$ and where $t_1 < t_2 = t' < t$, then we have that $t_1 < t$. Thus, since $P(\mathbf{a})$ is both in $\Delta^{t_1}(\Pi)$ and $\Delta^t(\Pi)$ where $t_1 \neq t$ (since $t_1 < t$), this is a contradiction by Claim 1. Therefore, it follows that $\mathcal{M}' \models \widehat{Pos(r) < P}(\mathbf{a})$ so that (15) holds as well.

($\Leftarrow$) Assume that $\mathcal{M}'$ is the expansion of $\mathcal{M}$ such that $\mathcal{M}' \models \mathrm{OC}_{\mathbf{P}}(\Pi)$. Then there exists a *strict-partial* order $\mathcal{P} = (Dom(\mathcal{P}), <^{\mathcal{P}})$, where

$$Dom(\mathcal{P}) = \{P(\mathbf{a}) \mid \mathbf{a} \in P^{\mathcal{M}} \text{ and } P \in \mathbf{P}\},$$

such that $\mathcal{P}$ is induced by the interpretations of the comparison predicates $\leq_{PQ}$ (for $P, Q \in \mathbf{P}$). Then by the *Order-extension Theorem* (Kaye and Macpherson 1994), there exists a *strict-total* order $\mathcal{T} = (Dom(\mathcal{T}), <^{\mathcal{T}})$, where $Dom(\mathcal{T}) = Dom(\mathcal{P})$, such that for all distinct elements $P(\mathbf{a})$ and $Q(\mathbf{b})$ in $Dom(\mathcal{T})$, either $P(\mathbf{a}) <^{\mathcal{T}} Q(\mathbf{b})$ or $Q(\mathbf{b}) <^{\mathcal{T}} P(\mathbf{a})$ holds. Now, since $\mathcal{M}$ is finite, then $Dom(\mathcal{T})$ will be finite as well, so that a bottom element, denoted $bot(\mathcal{T})$, exists. Then for a $P(\mathbf{a}) \in Dom(\mathcal{T})$, define $\mathcal{T}^{P(\mathbf{a})}$ inductively by:

1. $\mathcal{T}^{bot(\mathcal{T})} = \{bot(\mathcal{T})\}$;
2. $\mathcal{T}^{succ(P(\mathbf{a}))} = \mathcal{T}^{P(\mathbf{a})} \cup \{succ(P(\mathbf{a}))\}$,

where $succ(P(\mathbf{a}))$ denotes the *successor* of $P(\mathbf{a})$ under the ordering $<^{\mathcal{T}}$. Then $\mathcal{T}^{top(\mathcal{T})}$, where $top(\mathcal{T})$ is the greatest element under $<^{\mathcal{T}}$, is simply the collection of all the elements of $Dom(\mathcal{T})$, since $\mathcal{T}$ is a strict-total ordering of $Dom(\mathcal{T})$. We will now show by induction on $P(\mathbf{a})$ that $\mathcal{T}^{P(\mathbf{a})} \subseteq \mathcal{M}^\infty(\Pi)$ for all $P(\mathbf{a}) \in Dom(\mathcal{T})$. For the base case, assume $bot(\mathcal{T}) = P(\mathbf{a})$. Then since $\mathcal{M}'$ satisfies (13), there exists a rule $r \in \Pi$ with $Head(r) = \alpha_1 \times \ldots \times \alpha_k$ such that for some $1 \leq i \leq k$ and some assignment η, we have that

$$\mathcal{M}' \models (\mathbf{a} = \mathbf{y} \wedge \widehat{Body(r)} \wedge Pos(\widehat{r) < P}(\mathbf{a}) \wedge \bigwedge_{1 \leq j \leq i-1} \neg\alpha_j)\eta,$$

and where $\alpha_i = P(\mathbf{y})$. Moreover, since $P(\mathbf{a})$ is the bottom element under the ordering $<^{\mathcal{T}}$, then $Pos(r)$ does not mention atoms of intensional predicates (since the strict-total order $\mathcal{T}$ is induced by the interpretation of the comparison atoms). Therefore, since $P(\mathbf{a}) = \alpha_i\eta \in \mathcal{M}$ (because $Dom(\mathcal{T}) = \{P(\mathbf{a}) \mid \mathbf{a} \in P^{\mathcal{M}} \text{ and } P \in \mathbf{P}\}$), then it is not difficult to verify by Item 2 in (9) that $P(\mathbf{a}) \in \mathcal{M}^1(\Pi) \subseteq \mathcal{M}^\infty(\Pi)$. Now assume $\mathcal{T}^{Q(\mathbf{b})} \subseteq \mathcal{M}^\infty(\Pi)$ and we will show that $\mathcal{T}^{succ(Q(\mathbf{b}))} \subseteq \mathcal{M}^\infty(\Pi)$ as well. Thus, for convenience, assume that $succ(Q(\mathbf{b})) = P(\mathbf{a})$. Now, since $\mathcal{M}'$ satisfies (13), then there exists some rule $r \in \Pi$ with $Head(r) = \alpha_1 \times \ldots \times \alpha_k$ and assignment η such that

$$\mathcal{M}' \models (\mathbf{a} = \mathbf{y} \wedge \widehat{Body(r)} \wedge Pos(\widehat{r) < P}(\mathbf{a}) \wedge \bigwedge_{1 \leq j \leq i-1} \neg\alpha_j)\eta,$$

where $\alpha_i = P(\mathbf{y})$ (the "ordered support" for $P(\mathbf{a})$). Then since $\mathcal{M}' \models (\widehat{Pos(r) < P}(\mathbf{a}))\eta$ and $\mathcal{T}$ is the total-order extension of the strict-partial order induced by the interpretation of the comparison atoms, we have that $Pos(r)\eta \subseteq \mathcal{T}^{Q(\mathbf{b})}$ since $Q(\mathbf{b}) <^{\mathcal{T}} P(\mathbf{a})$. Moreover, since $\mathcal{M}' \models \widehat{Body(r)}\eta$, then $Neg(r)\eta \cap \mathcal{M} = \emptyset$ as well. Therefore, since $\mathcal{T}^{Q(\mathbf{b})} \subseteq \mathcal{M}^t(\Pi)$ for some t (since $\mathcal{T}^{Q(\mathbf{b})} \subseteq \mathcal{M}^\infty(\Pi)$ by the inductive hypothesis), then we have that both $Pos(r)\eta \subseteq \mathcal{M}^t(\Pi)$ and $Neg(r)\eta \cap \mathcal{M} = \emptyset$ holds. In addition, since we also have from (13) that $\mathcal{M} \models (\neg\alpha_1 \times \ldots \times \neg\alpha_{i-1})\eta$ and where $\alpha_i\eta = P(\mathbf{a}) \in \mathcal{M}$, then it follows from (9) that $P(\mathbf{a}) \in \mathcal{M}^{t+1}(\Pi)$. Hence, since we have shown that $\mathcal{T}^{top(\mathcal{T})} \subseteq \mathcal{M}^\infty(\Pi)$, then since $\mathcal{M}^0(\Pi) \subseteq \mathcal{M}^\infty(\Pi)$, it follows that $(\mathcal{T}^{top(\mathcal{T})} \cup \mathcal{M}^0(\Pi)) = \mathcal{M} \subseteq \mathcal{M}^\infty(\Pi)$. Therefore, to show that $\mathcal{M} = \mathcal{M}^\infty(\Pi)$, it is left for us to show that $\mathcal{M}^\infty(\Pi) \subseteq \mathcal{M}$. We now show this fact by induction on t. For the base case, we clearly have that $\mathcal{M}^0(\Pi) \subseteq \mathcal{M}$ since for $\mathcal{M}^0(\Pi)$, all the interpretation of the comparison predicates are set to empty. Now let us assume that for $0 \leq t' \leq t$, we have that $\mathcal{M}^{t'}(\Pi) \subseteq \mathcal{M}$ holds and we will now show that $\mathcal{M}^{t+1}(\Pi) \subseteq \mathcal{M}$ holds as well. Indeed, let $P(\mathbf{a}) \in \mathcal{M}^{t+1}(\Pi) \setminus \mathcal{M}^t(\Pi)$. Then by the definition of $\mathcal{M}^{t+1}(\Pi) \setminus \mathcal{M}^t(\Pi)$, there exists a rules $r \in \Pi$ with $Head(r) = \alpha_1 \times \ldots \times \alpha_i \times \ldots \times \alpha_k$ and assignment η such that

1. $P(\mathbf{a}) = \alpha_i\eta$;
2. $Pos(r)\eta \subseteq \mathcal{M}^t(\Pi)$ and $Neg(r)\eta \cap \mathcal{M} = \emptyset$;
3. i (for $1 \leq i \leq k$) is the largest i such that $\{\alpha_1\eta, \ldots, \alpha_{i-1}\eta\} \cap \mathcal{M} = \emptyset$.

Then since $\mathcal{M}^t(\Pi) \subseteq \mathcal{M}$ and $\mathcal{M} \models \widehat{\Pi}$ (since $\mathcal{M}$ satisfies (12)), then it follows that $P(\mathbf{a}) \in \mathcal{M}$. Therefore, since we had shown that $\mathcal{M}^\infty(\Pi) = \mathcal{M}$, then we have by Theorem 2 that $\mathcal{M}$ is a stable model of Π. □

Theorem 3 has an important practical value towards an LPOD solver development. Basically, for a given LPOD, to compute its stable models, we can firstly translate this program into its corresponding ordered completion $\mathrm{OC}_{\mathbf{P}}(\Pi)$, then by taking extensional databases as input to ground $\mathrm{OC}_{\mathbf{P}}(\Pi)$ to a propositional formula, and finally compute its classical models by calling an SAT solver (e.g., SMT). Some advantages of this approach over other ASP solvers for normal logic programs have been demonstrated in (Asuncion et al. 2012).

Proposition 2 *Let Π be an LPOD. Then $\mathrm{OC}_{\mathbf{P}}(\Pi)$ can be computed in time $O(|\Pi| \cdot N \cdot H \cdot |\mathbf{P}| + |\mathbf{P}|^3)$, where N and H are the maximum length of the rules and the maximum length of the ordered disjunctive heads in the rules of Π, respectively.*

Complexity and Expressiveness

Now we investigate the complexity and expressiveness issues of LPODs. For the propositional case, we can prove that the stable model existence problem for LPODs is NP-complete[1]. In the following, we focus on the first-order case. Let Π_1 and Π_2 be two programs and $\tau(\Pi_1) \subseteq \tau(\Pi_2)$. We say that Π_1 and Π_2 are *equivalent under* $\tau(\Pi_1)$ if Π_1 and Π_2 have exactly the same stable models by restricting each Π_2's stable model $\mathcal{M}$ to $\mathcal{M}\restriction_{\tau(\Pi_1)}$.

We say that a program is *positive* if negation only occurs on atoms of extensional predicates. A normal logic program is *local variable free* if all the variables in the bodies of rules are also mentioned in their corresponding heads.

Proposition 3 *Under finite structures, every local variable free normal logic program Π can be translated to a positive LPOD with auxiliary predicates such that these two programs are equivalent under $\tau(\Pi)$.*

Proof: Let Π^{NORM} be a local variable free FO normal program with intensional predicate symbols $\mathcal{P}_{int}(\Pi^{\mathrm{NORM}})$. Then for each predicate $P \in \mathcal{P}_{int}(\Pi^{\mathrm{NORM}})$, let us introduce a new predicate $\overline{P}$ of the same arity as P. Roughly speaking, $\overline{P}$ will encode the negative extents of P. Now we are ready to define the positive LPOD Π^{LPOD}.

Denote by $(\Pi^{\mathrm{NORM}})^{pos}$ the following set of rules:

$$\{\alpha \leftarrow \beta_1, \ldots, \beta_l, \overline{\gamma_1}, \ldots, \overline{\gamma_m} \mid \alpha \leftarrow \beta_1, \ldots, \beta_l, \mathsf{not}\ \gamma_1, \ldots, \mathsf{not}\ \gamma_m \in \Pi^{\mathrm{NORM}}, \text{ for } 1 \leq i \leq m, \text{if } \gamma_i = P(\mathbf{t}) \text{ and } P \in \mathcal{P}_{int}(\Pi^{\mathrm{NORM}}) \text{ then } \overline{\gamma_i} = \overline{P}(\mathbf{t}), \text{ otherwise } \overline{\gamma_i} = \gamma_i\}. \quad (16)$$

Then clearly, $(\Pi^{\mathrm{NORM}})^{pos}$ is a positive LPOD of signature $\tau(\Pi) \cup \{\overline{P} \mid P \in \mathcal{P}_{int}(\Pi^{\mathrm{NORM}})\}$.

Let S be a set of atoms occurring in Π, we denote by $\mathcal{A}_{ext}(S)$ and $\mathcal{A}_{int}(S)$ the sets of extensional and intensional atoms in S, respectively. Now let us define another program Π' as follows:

$$\{Bd_{r_1}(\mathbf{x}) \times \ldots \times Bd_{r_k}(\mathbf{x}) \leftarrow P(\mathbf{x}) \mid P \in \mathcal{P}_{int}(\Pi^{\mathrm{NORM}}), P \text{ has defining rules } r_1, \ldots, r_k \in \Pi^{\mathrm{NORM}}\} \quad (17)$$

$$\cup \{\beta \leftarrow Bd_r(\mathbf{x}), \bot \leftarrow \mathsf{not}\ \beta', Bd_r(\mathbf{x}), \overline{\gamma} \leftarrow Bd_r(\mathbf{x}), \bot \leftarrow \gamma', Bd_r(\mathbf{x}) \mid r \in \Pi^{\mathrm{NORM}}, \beta \in \mathcal{A}_{int}(Pos(r)), \beta' \in \mathcal{A}_{ext}(Pos(r)), \gamma \in \mathcal{A}_{int}(Neg(r)), \gamma' \in \mathcal{A}_{ext}(Neg(r))\} \quad (18)$$

$$\cup \{<_{QP}(\mathbf{z}, \mathbf{x}) \leftarrow Bd_r(\mathbf{x}) \mid r \in \Pi^{\mathrm{NORM}}, Head(r) = P(\mathbf{x}), Q(\mathbf{z}) \in \mathcal{A}_{int}(Pos(r))\} \quad (19)$$

$$\cup \{<_{PR}(\mathbf{x}, \mathbf{z}) \leftarrow <_{PQ}(\mathbf{x}, \mathbf{y}), <_{QR}(\mathbf{y}, \mathbf{z}) \mid P, Q, R \in \mathcal{P}_{int}(\Pi^{\mathrm{NORM}})\} \quad (20)$$

$$\cup \{\bot \leftarrow <_{PQ}(\mathbf{x}, \mathbf{y}), <_{QP}(\mathbf{y}, \mathbf{x}) \mid P, Q \in \mathcal{P}_{int}(\Pi^{\mathrm{NORM}})\}. \quad (21)$$

Lastly, further define one more program Π'' as follows:

$$\{\bot \leftarrow P(\mathbf{x}), \overline{P}(\mathbf{x}), \quad (22)$$

$$P(\mathbf{x}) \times \overline{P}(\mathbf{x}) \leftarrow \mid P \in \mathcal{P}_{int}(\Pi^{\mathrm{NORM}})\}. \quad (23)$$

Then we can now set Π^{LPOD} as the union $(\Pi^{\mathrm{NORM}})^{pos} \cup \Pi' \cup \Pi''$. Clearly, Π^{LPOD} is a positive LPOD.

Intuitively, we have that (16)-(21) encode the ordered completion of Π^{NORM} within the LPOD Π^{LPOD} itself. In particular, we note (18), which encodes the "supporting" atoms for the "body atoms" $Bd_r(\mathbf{x})$ (which corresponds to the body of the rule r), but in such a way that we avoid defining the extensional predicates in a head of a rule. On the other hand, (22) enforces that $(P(\mathbf{x}) \rightarrow \neg\overline{P}(\mathbf{x})) \wedge (\overline{P}(\mathbf{x}) \rightarrow \neg P(\mathbf{x}))$ while (23) enforces that $(\neg P(\mathbf{x}) \rightarrow \overline{P}(\mathbf{x})) \wedge (\neg\overline{P}(\mathbf{x}) \rightarrow P(\mathbf{x}))$, i.e., the "necessary" counterpart of (22) with respect to the extents of $\overline{P}$ being the symmetric negation of those in P. □

Theorem 4 *Let Π be an LPOD and $\mathcal{M}$ a finite $\tau_{ext}(\Pi)$-structure, i.e., an extensional or input database structure. Then the problem of determining if $\mathcal{M}$ can be expanded to a stable model of Π is NP-complete. This result remains true even for positive LPODs.*

Proof: (*Membership*) Given our reduction of Π to the FO formula $\mathrm{OC}_{\mathbf{P}}[\Pi]$ via Theorem 3 (i.e., the reduction of Π to its ordered completion), we have that determining if $\mathcal{M}$ can be expanded to a model of $\mathrm{OC}_{\mathbf{P}}[\Pi]$ is the model expansion problem, which is in NP (model expansion is in fact NP-complete).

(*Hardness*) Consider the 3-color program Π_{3color} as fol-

[1] In our full version, we provide a polynomial translation from normal logic programs to LPODs. This result can be also obtained using a different translation proposed in (Nieuwenborgh and Vermeir 2003).

lows:

$$C_1(x) \leftarrow \mathsf{not}\, C_2(x), \mathsf{not}\, C_3(x), \quad (24)$$
$$C_2(x) \leftarrow \mathsf{not}\, C_1(x), \mathsf{not}\, C_3(x), \quad (25)$$
$$C_3(x) \leftarrow \mathsf{not}\, C_1(x), \mathsf{not}\, C_2(x), \quad (26)$$
$$\leftarrow E(x,y), C_1(x), C_1(y), \quad (27)$$
$$\leftarrow E(x,y), C_2(x), C_2(y), \quad (28)$$
$$\leftarrow E(x,y), C_3(x), C_3(y). \quad (29)$$
$$(30)$$

Given a graph structure $\mathcal{G} = (Dom(\mathcal{G}), V^{\mathcal{G}}, E^{\mathcal{G}})$ such that $V^{\mathcal{G}} = Dom(\mathcal{G})$ (i.e., the vertices of $\mathcal{G}$), $\mathcal{G}$ has a corresponding 3-coloring iff Π_{3color} has a stable model. It is well known that the problem of 3-coloring is NP-complete. On the other hand, we have by Proposition 3 that Π_{3color} can be reduced to a positive LPOD since Π_{3color} is a local variable free normal program. □

A LPOD is called *almost positive* if each negated intensional atoms in the program only occurs in the bodies of some constraints. The following theorem states that almost positive LPODs precisely capture the full class of normal logic programs. That is, ordered disjunction and constraints are sufficient enough to represent negation as failure.

Theorem 5 *Every normal logic program* Π *can be translated to an almost positive LPODs with auxiliary predicates such that these two programs are equivalent under* $\tau(\Pi)$.

Proof: Let Π^{NORM} be an arbitrary normal logic program. We define a reduction from Π^{NORM} to an almost positive LPOD Π^{LPOD} by defining Π^{LPOD} as the following set of rules:

$$\{\alpha \leftarrow \beta_1, \ldots, \beta_l, \overline{\gamma_1}, \ldots, \overline{\gamma_m} \mid \alpha \leftarrow \beta_1, \ldots, \beta_l, \mathsf{not}\, \gamma_1, \ldots, \mathsf{not}\, \gamma_m \in \Pi^{\text{NORM}},$$
$$\text{for } 1 \le i \le m, \text{if } \gamma_i = P(\mathbf{t}) \text{ and } P \in \mathcal{P}_{int}(\Pi^{\text{NORM}}) \text{ then } \overline{\gamma_i} = \overline{P}(\mathbf{t}), \text{ otherwise } \overline{\gamma_i} = \gamma_i\} \quad (31)$$
$$\cup \{\bot \leftarrow P(\mathbf{x}), \overline{P}(\mathbf{x}), \quad (32)$$
$$\bot \leftarrow \mathsf{not}\, P(\mathbf{x}), \mathsf{not}\, \overline{P}(\mathbf{x}), \quad (33)$$
$$\bot \leftarrow P'(\mathbf{x}), \overline{P}(\mathbf{x}), \quad (34)$$
$$P'(\mathbf{x}) \times \overline{P}(\mathbf{x}) \leftarrow \mid P \in \mathcal{P}_{int}(\Pi^{\text{NORM}})\}, \quad (35)$$

where for each $P \in \mathcal{P}_{int}(\Pi)$, we introduce two new predicate symbols $\overline{P}$ and P'. Then Π^{LPOD} is clearly an almost positive LPOD since negation only occurs in the constraints in (33).

Intuitively, the key here is the combination of (34) and (35), which enforces $(\neg P'(\mathbf{x}) \rightarrow \overline{P}(\mathbf{x})) \wedge (\neg \overline{P}(\mathbf{x}) \rightarrow P'(\mathbf{x}))$. This has the effect of "fixing" $\overline{P}(\mathbf{x})^*$ since the "minimization" of $\overline{P}^*$ implies the "expansion" of $(P')^*$ (and vice versa). This simulates the condition that $\overline{P}(\mathbf{x})$ (which encodes "$\mathsf{not}\, P(\mathbf{x})$") will be fixed in the minimization of $\overline{P}(\mathbf{x})^*$ since we also have $(\neg P'(\mathbf{x})^* \rightarrow \overline{P}(\mathbf{x})^*) \wedge (\neg \overline{P}(\mathbf{x})^* \rightarrow P'(\mathbf{x})^*)$. On the other hand, (32) and (33) again enforces $(P(\mathbf{x}) \rightarrow \neg \overline{P}(\mathbf{x})) \wedge (\overline{P}(\mathbf{x}) \rightarrow \neg P(\mathbf{x}))$ and $(\neg P(\mathbf{x}) \rightarrow \overline{P}(\mathbf{x})) \wedge (\neg \overline{P}(\mathbf{x}) \rightarrow P(\mathbf{x}))$, respectively. □

Preferred Stable Model Semantics

As indicated by Brewka (2002; 2006), the stable models based on split programs are usually not sufficient to capture the preference semantics for a given LPOD, while a certain preference relation based on the degree of satisfaction has to be imposed on the stable models. Now we show how such preferred stable model semantics can be defined for our FO LPODs.

Definition 4 *Let* r *be a rule in some LPOD* Π, $\mathcal{M}$ *a* $\tau(\Pi)$*-structure, and* η *an assignment from the distinct tuple of variables* $\mathbf{x}$ *of* r *to* $Dom(\mathcal{M})$. *The* satisfaction degree *of* r *under the structure* $\mathcal{M}$ *and assignment* η, *denoted by* $Deg_{(\mathcal{M},\eta)}(r)$, *is defined as follows:*

$$Deg_{(\mathcal{M},\eta)}(r) := \begin{cases} 1 & \text{if } \mathcal{M} \not\models \widehat{Body(r)}\eta, \text{ otherwise} \\ i & \text{if } \mathcal{M} \models (\widehat{Body(r)} \wedge \neg\alpha_1 \wedge \neg\alpha_2 \wedge \\ & \ldots \wedge \neg\alpha_{i-1} \wedge \alpha_i)\eta. \end{cases}$$

Similarly to the propositional case, if $(\mathcal{M}, \eta)$ does not satisfy the rule r's body, then the default satisfaction degree is 1. Otherwise, the satisfaction degree is the "minimal" (i.e., most preferred) of the atoms in $\alpha_1 \times \ldots \times \alpha_k$ that $(\mathcal{M}, \mathbf{a})$ satisfies. Again as in the propositional case, this encodes that we are paying for a penalty for the "least preferred" atoms in $\alpha_1 \times \ldots \times \alpha_k$ being satisfied in the sense that, the smaller the satisfaction degree, then the better it is.

Definition 5 *Let* Π *be an LPOD and* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *be two stable models of* Π *such that* $\mathcal{M}_1 \restriction_{\tau_{ext}(\Pi)} = \mathcal{M}_2 \restriction_{\tau_{ext}(\Pi)}$. *Then we say that* $\mathcal{M}_1$ *is* Pareto-preferred *to* $\mathcal{M}_2$, *denoted by* $\mathcal{M}_1 > \mathcal{M}_2$, *if the following two conditions hold:*

1. *For each rule* $r \in \Pi$ *and assignment* η, *we have that* $Deg_{(\mathcal{M}_1,\eta)}(r) \le Deg_{(\mathcal{M}_2,\eta)}(r)$;
2. *There is a rule* $r' \in \Pi$ *and an assignment* η' *such that* $Deg_{(\mathcal{M}_1,\eta')}(r') < Deg_{(\mathcal{M}_2,\eta')}(r')$.

Example 2 *Assume* Π *to be the following LPOD with the single rule* r*:*

$$r: \; P(x) \times Q(x) \times R(x) \leftarrow S(x) \quad (36)$$

such that S *is the only extensional predicate. Now let* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *be two* $\tau(\Pi)$*-structures such that:*

$$\mathcal{M}_1 = (\{a,b\}, S^{\mathcal{M}_1} = \{a,b\}, P^{\mathcal{M}_1} = \{a\}, Q^{\mathcal{M}_1} = \{b\}, R^{\mathcal{M}_1} = \emptyset);$$
$$\mathcal{M}_2 = (\{a,b\}, S^{\mathcal{M}_2} = \{a,b\}, P^{\mathcal{M}_2} = \emptyset, Q^{\mathcal{M}_2} = \{b\}, R^{\mathcal{M}_2} = \{a\}).$$

Then we have that $\mathcal{M}_1 \restriction_{\tau_{ext}(\Pi)} = \mathcal{M}_2 \restriction_{\tau_{ext}(\Pi)}$ *and where the grounding of* Π *under the domain* $\{a,b\}$ *will be the propositional program*

$$P(a) \times Q(a) \times R(a) \leftarrow S(a) \quad (37)$$
$$P(b) \times Q(b) \times R(b) \leftarrow S(b). \quad (38)$$

Then based on the grounded Π, *it can be seen that both* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *are stable models of* Π. *In addition, since for all the assignments* $\eta_1 : x \longrightarrow a$ *and* $\eta_2 : x \longrightarrow b$ *of the variable* x *to* $\{a,b\}$, *we have that* $Deg_{(\mathcal{M}_1,\eta_1)}(r) < Deg_{(\mathcal{M}_2,\eta_1)}(r)$ *and* $Deg_{(\mathcal{M}_1,\eta_2)}(r) = Deg_{(\mathcal{M}_2,\eta_2)}(r)$, *then* $\mathcal{M}_1 > \mathcal{M}_2$, *i.e.,* $\mathcal{M}_1$ *is Pareto-preferred to* $\mathcal{M}_2$. □

Based on the notion of a Pareto-preferred stable models, we can now define the notion of a preferred stable model of an LPOD.

Definition 6 *Let Π be an LPOD and $\mathcal{M}$ its stable model. $\mathcal{M}$ is called a* preferred stable model *of Π if there does not exists another structure $\mathcal{M}'$ for which $\mathcal{M}' > \mathcal{M}$.*

Logic Formalization of Preferred Stable Models

Now we provide a logical characterization of the preferred stable models. For this purpose, let us assume the tuple of distinct intensional predicates in our signature to be $\mathbf{P} = P_1 \dots P_n$. We use $\mathbf{P'}$ to denote the tuple of distinct predicates $P'_1 \dots P'_s$ such that each P'_i $(1 \leq i \leq s)$ is a new predicate symbol.

Consider a rule r of an LPOD with the form (3) and whose atoms are predicates from $\mathbf{P}$. We use $Deg(r)^{\mathbf{P'}} \leq Deg(r)^{\mathbf{P}}$ to denote the following formula:

$$\bigwedge_{1 \leq i \leq k} \forall \mathbf{x}_r (\, (\widehat{Body(r)} \wedge \bigwedge_{1 \leq j \leq i} \neg\alpha_j)[\mathbf{P}/\mathbf{P'}] \rightarrow (\widehat{Body(r)} \wedge \bigwedge_{1 \leq j \leq i} \neg\alpha_j)\,), \quad (39)$$

where $(\widehat{Body(r)} \wedge \bigwedge_{1 \leq j \leq i} \neg\alpha_j)[\mathbf{P}/\mathbf{P'}]$ denotes the formula obtained from $(\widehat{Body(r)} \wedge \bigwedge_{1 \leq j \leq i} \neg\alpha_j)$ by replacing the occurrences of predicates from $\mathbf{P}$ by those corresponding ones in $\mathbf{P'}$.

Roughly speaking, $Deg(r)^{\mathbf{P'}} \leq Deg(r)^{\mathbf{P}}$ is a formula that encodes all instances of r under the interpretation of the predicates from $\mathbf{P'}$ is at an equal or lower satisfaction degree than those ones from $\mathbf{P}$. Indeed, (39) encodes that the degree of r under $\mathbf{P'}$ is less than or equal than those under $\mathbf{P}$. As we will see later on, the purpose of the predicates $\mathbf{P'}$ that matches $\mathbf{P}$ is it will be used as second-order variables.

We now extend this notion to a whole LPOD Π so that by $Deg(\Pi)^{\mathbf{P'}} \leq Deg(\Pi)^{\mathbf{P}}$, we denote the conjunctions $\bigwedge_{r \in \Pi}(Deg(r)^{\mathbf{P'}} \leq Deg(r)^{\mathbf{P}})$, and that we further denote by $Deg(\Pi)^{\mathbf{P'}} < Deg(\Pi)^{\mathbf{P}}$ as the conjunction

$$(Deg(\Pi)^{\mathbf{P'}} \leq Deg(\Pi)^{\mathbf{P}}) \wedge \neg(Deg(\Pi)^{\mathbf{P}} \leq Deg(\Pi)^{\mathbf{P'}}). \quad (40)$$

Intuitively, $Deg(\Pi)^{\mathbf{P'}} < Deg(\Pi)^{\mathbf{P}}$ is a formula which encodes that the interpretation $\mathbf{P'}$ satisfies Π in a more optimal manner (i.e., in the sense of Definition 5) than that of Π under $\mathbf{P}$.

The following theorem now provides a classical logic characterization of preferred stable models of LPODs via a second-order sentence.

Theorem 6 *Let Π be an LPOD with tuple of intensional predicates $\mathbf{P} = P_1 \dots P_n$ and $\mathbf{P'} = P'_1 \dots P'_n$ the fresh tuple of predicates matching $\mathbf{P}$, and $\mathcal{M}$ a $\tau(\Pi)$-structure. Then $\mathcal{M}$ is a preferred stable model of Π if and only if it satisfies the following second-order sentence:*

$$\mathrm{SM}_{\mathbf{P}}(\Pi) \wedge \neg\exists \mathbf{P'}(\, \mathrm{SM}_{\mathbf{P}}(\Pi)[\mathbf{P}/\mathbf{P'}] \wedge Deg(\Pi)^{\mathbf{P'}} < Deg(\Pi)^{\mathbf{P}}\,), \quad (41)$$

where $\mathrm{SM}_{\mathbf{P}}(\Pi)[\mathbf{P}/\mathbf{P'}]$ denotes the formula obtained from $\mathrm{SM}_{\mathbf{P}}(\Pi)$ by simultaneously replacing all predicates from $\mathbf{P}$ by those corresponding ones from $\mathbf{P'}$.

It can be showed that (41) also correctly represents Brewka's preferred stable model semantics when we restrict it to the case of propositional LPODs.

Corollary 1 *On finite structures, $\mathcal{M}$ is a preferred stable model of Π if and only if it satisfies the following second-order sentence:*

$$\mathrm{OC}_{\mathbf{P}}(\Pi) \wedge \neg\exists \mathbf{P'}\mathbf{T'}(\, \mathrm{OC}_{\mathbf{P}}(\Pi)[\mathbf{PT}/\mathbf{P'}\mathbf{T'}] \wedge Deg(\Pi)^{\mathbf{P'}\mathbf{T'}} < Deg(\Pi)^{\mathbf{QT}}\,),$$

where $\mathrm{OC}_{\mathbf{P}}(\Pi)$ is the ordered completion of Π (a first-order sentence) as defined in Definition 3, and we assume that $\mathbf{T}$ denotes the tuple of distinct comparison predicates and such that $\mathbf{T'}$ are the new predicate symbols that matches $\mathbf{T}$.

Theorem 7 *Let Π be an LPOD and $\mathcal{M}$ a $\tau(\Pi)$-structure. Then determining if $\mathcal{M}$ is a preferred stable model of Π is co-NP-complete.*

Proof: (*Sketch*) Due to a space limit, here we only present a sketch proof for the hardness. A full proof is given in our full paper. For a given normal logic program Π^{NORM} and an extensional structure $\mathcal{M}_{ext}$ of it, we reduce the problem of determining if $\mathcal{M}_{ext}$ cannot be expanded to a stable model of Π^{NORM} to the problem of determining if a stable model $\mathcal{M}^*$ of an LPOD Π^* is a preferred stable model. Let Π^{NORM} be an arbitrary normal logic program. We specify a program $\Pi^{\text{NORM}}_{\{P(a)\}}$ as follows:

$$\Pi^{\text{NORM}}_{\{P(a)\}} := \{\, \alpha \leftarrow P(a), \beta_1, \dots, \beta_l, \mathsf{not}\ \gamma_1, \dots, \mathsf{not}\ \gamma_m \mid \alpha \leftarrow \beta_1, \dots, \beta_l, \mathsf{not}\ \gamma_1, \dots, \mathsf{not}\ \gamma_m \in \Pi^{\text{NORM}} \,\},$$

where we assume that $P(a)$ is an atom with P a new predicate symbol and a a new constant symbol. Then $\Pi^{\text{NORM}}_{\{P(a)\}}$ is simply the program obtained from Π^{NORM} by adding the atom $P(a)$ (i.e., propositional atom) into each of the positive bodies of the rules of Π^{NORM}. Now by Π^{LPOD}, define the following set of rules such that:

$$\begin{aligned}
\Pi^{\text{LPOD}} := \{ \quad P(a) &\leftarrow \mathsf{not}\, Q(a), && (42)\\
Q(a) &\leftarrow \mathsf{not}\, P(a), && (43)\\
R_1(a) \times R_2(a) &\leftarrow Q(a), && (44)\\
&\leftarrow R_1(a) \quad \}, && (45)
\end{aligned}$$

where we assume here that Q, R_1, and R_2 are new predicate symbols as well. Intuitively, rules (42) and (43) are the defining (and the only defining) rules for $P(a)$ and $Q(a)$, respectively, such that at least one, and not both, must be in any stable model of Π^{LPOD}. In addition, the rule (44) is an LPOD rule with $R_1(a)$ taking preference over $R_2(a)$. Finally, the constraint (45) simply enforces to choose $R_2(a)$ (which is less preferred over $R_1(a)$) over $R_1(a)$ so that the satisfaction degree of (44) will only be as low as 1 only if $Q(a)$ is false (since $Q(a)$ false makes the body of (44) to be false as well). Thus, if we want to get the most preferred stable model of Π^{LPOD}, then it must imply $Q(a)$ to be false. Finally, we can prove the following claim, which leads to the completion of this proof.

Claim: *There exists a stable model $\mathcal{M}^*$ of program $\Pi^{\text{NORM}}_{\{P(a)\}} \cup \Pi^{\text{LPOD}}$, such that $\mathcal{M}^*$ is a preferred stable model of $\Pi^{\text{NORM}}_{\{P(a)\}} \cup \Pi^{\text{LPOD}}$ iff $\mathcal{M}_{ext}$ cannot be expanded to a stable model of Π^{NORM}.*

□

Proposition 4 *Let Π be an LPOD and $\mathcal{M}$ a corresponding extensional structure. The problem of determining if $\mathcal{M}$ can be expanded to a preferred stable model of Π is NP-complete.*

Conclusions

Preference plays an important role in commonsense reasoning, while developing an effective yet expressive mechanism of handling preference in Answer Set Programming is technically challenging, e.g., (Brewka, Truszczynski, and Woltran 2010; Delgrande, Schaub, and Tompits 2004). In this paper we have developed a formulation of fist-order LPODs which may be viewed as a natural generalization of Brewka's propositional LPODs. The relevant semantics and expressiveness results also provide important insights for us to understand this type of first-order LPODs.

The proposed both the second-order based stable model semantics and the progression semantics capture two important aspects of LPODs: their relationship to classical logic as well as the underlying reasoning feature involving ordered disjunction on the first-order level.

The translation from LPODs to first-order sentences extends the previous work of Asuncion *et al*' ordered completion and we argue that this translation will be useful in developing an effective solver for first-order LPODs, as demonstrated in (Asuncion et al. 2012) for normal logic programs. The complexity and expressiveness results confirm that LPODs remain in NP and the hardness holds even for positive LPODs, while almost positive LPODs capture the full class of normal logic programs. Our logic characterization of preferred stable model semantics reveals that we can eventually use a classical second-order sentence to precisely represent the preference relation among stable models so that such first-order preference semantics may be formalized in a unified way as LPOD first-order stable model semantics.

For future work, we are considering to develop a first-order LPOD solver based on the translation proposed in section 4. Another interesting work is to extend our semantics for first-order LPODs by allowing both ordered and unordered disjunction in the heads of rules, as discussed in (Cabalar 2011), and explore their logical properties and possible applications in practical domains.

Acknowledgement

This publication was made possible by the support of an NPRP grant (NPRP 09-079-1-013) from the Qatar National Research Fund (QNRF). The statements made herein are solely the responsibility of the authors.

References

Asuncion, V.; Lin, F.; Zhang, Y.; and Zhou, Y. 2012. Ordered completion for first-order logic programs on finite structures. *Artificial Intelligence* 177-179:1–24.

Asuncion, V.; Zhang, Y.; and Zhou, Y. 2013. Preferred first-order answer set programs. *ACM Transactions on Computational Logic* to appear.

Babb, J., and Lee, J. 2012. Module theorem for the general theory of stable models. *Theory and Practice of Llogic Programming* 12(4-5):719–735.

Brewka, G.; Benferhat, S.; and Berre, D. L. 2002. Qualitive choice logic. In *Proceedings of KR-2002*, 158–169.

Brewka, G.; Truszczynski, M.; and Woltran, S. 2010. Representing preferences among sets. In *Proceedings of AAAI-2010*, 273–278.

Brewka, G. 2002. Logic programming with ordered disjunction. In *Proceedings of AAAI-2002*, 100–105.

Brewka, G. 2006. Answer sets and qualitative optimization. *Logic Journal of the IGPL* 14(3):413–433.

Cabalar, P. 2011. A logical characterisation of ordered disjunction. *AI Communications* 24(2):165–175.

Confalonieri, R.; Nieves, J.; Osorio, M.; and Vazquez-Salceda, J. 2010. Possibilistic semantics for logic programs with ordered disjunction. In *Proceedings of FoIKS-2010*, 133–152.

Delgrande, J.; Schaub, T.; and Tompits, H. 2004. A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence* 20:308–334.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence* 175(1):236–263.

Karger, P.; Lopes, N.; Olmedilla, D.; and Polleres, A. 2008. Towards logic programs with ordered and unordered disjunction. In *Proceedings of Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP 2008), 24th International Conference on Logic Programming (ICLP 2008)*.

Kaye, R., and Macpherson, D. 1994. *Automorphism of First-order Structures*. Clarendon Press.

Lee, J., and Meng, Y. 2011. First-order stable model semantics and first-order loop formulas. *Journal of Artificial Intelligence Research (JAIR)* 42:125–180.

Lin, F., and Zhou, Y. 2011. From answer set logic programming to circumscription via logic of gk. *Artificial Intelligence* 175(1):264–277.

Nieuwenborgh, D. V., and Vermeir, D. 2003. Order and negation as failure. In *Proceedings of ICLP-2003*, 194–208.

Zhang, Y., and Zhou, Y. 2010. On the progression semantics and boundedness of answer set programs. In *Proceedings of KR-2010*, 518–527.

Constructive Negation in Extensional Higher-Order Logic Programming

Angelos Charalambidis
Dept. of Informatics and Telecommunications
University of Athens
a.charalambidis@di.uoa.gr

Panos Rondogiannis
Dept. of Informatics and Telecommunications
University of Athens
prondo@di.uoa.gr

Abstract

Extensional higher-order logic programming has been recently proposed as an interesting extension of classical logic programming. An important characteristic of the new paradigm is that it preserves all the well-known properties of traditional logic programming. In this paper we enhance extensional higher-order logic programming with *constructive negation*. We argue that the main ideas underlying constructive negation are quite close to the existing proof procedure for extensional higher-order logic programming and for this reason the two notions amalgamate quite conveniently. We demonstrate the soundness of the resulting proof procedure and describe an actual implementation of a language that embodies the above ideas. In this way we obtain the first (to our knowledge) higher-order logic programming language supporting constructive negation and offering a new style of programming that genuinely extends that of traditional logic programming.

1 Introduction

Extensional higher-order logic programming has been recently proposed (Charalambidis et al. 2010; 2013) as an interesting extension of classical logic programming. The key idea behind the new paradigm is that all predicates defined in a program (even the higher-order ones) denote sets and therefore one can use standard extensional set theory in order to understand the meaning of programs and reason about them. A consequence of this fact is that the semantics and the proof theory of extensional higher-order logic programming smoothly extend the corresponding ones for traditional (ie., first-order) logic programming.

In this paper we extend the new paradigm of extensional higher-order logic programming with *constructive negation* (Chan 1988; 1989). The combination of higher-order characteristics and constructive negation in the new language, allows the programmer to specify in a compact way many interesting problems. For example, in the new language it is quite convenient to enumerate the sets that satisfy a specific property (such as all the colorings of a graph, all the subsets of a set that satisfy a condition, and so on). In order to theoretically justify the proposed higher-order language, we define its *completion semantics* and demonstrate the soundness of the proof procedure with respect to this semantics. Finally, we describe an implementation of the new language which embodies the above ideas. In conclusion, the main contributions of the present work are the following:

- We propose the first (to our knowledge) higher-order logic programming language with constructive negation. The amalgamation of the two ideas seems quite natural and expressive (but is non-trivial from a technical point of view).
- We provide a theoretical justification of our proposal by defining the completion semantics for higher-order logic programs and by demonstrating the soundness of the proposed proof procedure with respect to this semantics. In this way we provide a promising direction for the study of the interplay between higher-order logic programming and negation-as-failure.

The rest of the paper is organized as follows. Section 2 describes the motivation behind the present work. Section 3 provides the syntax and semantics of the proposed higher-order language and Section 4 presents the corresponding proof-procedure and discusses its implementation. Section 5 demonstrates the soundness of the proof-procedure. Section 6 concludes the paper with discussion on future work.

2 Motivation

The purpose of this paper is to enhance extensional higher-order logic programming with constructive negation. These two notions prove to be very compatible, for reasons that we are going to explain in detail below.

In (Charalambidis et al. 2013) the higher-order logic programming language $\mathcal{H}$ is defined. Intuitively, $\mathcal{H}$ extends classical logic programming with higher-order user-defined predicates. The main novel characteristic of $\mathcal{H}$ with respect to other existing higher-order logic programming languages (such as Hilog (Chen, Kifer, and Warren 1993) and λ-Prolog (Miller and Nadathur 1986)) is that it has a purely extensional semantics: program predicates denote sets of objects and the semantics of the language is an extension of the traditional semantics of classical logic programming. Since predicates in $\mathcal{H}$ denote sets, the programmer is allowed to use uninstantiated predicate variables in clause bodies and in queries; when the proof-procedure of $\mathcal{H}$ encounters such

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

a variable, it performs a systematic enumeration of the possible values that this variable can take. For further details regarding $\mathcal{H}$ the interested reader should consult (Charalambidis et al. 2013). The following two examples motivate the language. To simplify our presentation for this section, we use an extended Prolog-like syntax. The exact syntax of the language that will be the object of our study will be specified in the next section.

Example 1. Consider the program:

```
p(Q):-Q(0),Q(1).
```

The meaning of p can be understood in set-theoretic terms, ie., p is a predicate that is true of all relations that contain (at least) 0 and 1. Thus, it is meaningful to ask the query:

```
?-p(R).
```

which will respond with an answer of the form $\texttt{R} = \{0, 1\} \cup \texttt{L}$ (having the obvious meaning described above).

Example 2. Consider the following program stating that a band is a group that has at least a singer and a guitarist:

```
band(B):-singer(S),B(S),guitarist(G),B(G).
```

Suppose that we also have a database of musicians:

```
singer(sally).
singer(steve).
guitarist(george).
guitarist(grace).
```

We can then ask the following query:

```
?-band(B).
```

which returns answers of the form $\texttt{B} = \{\texttt{sally}, \texttt{george}\} \cup \texttt{L}$, and so on.

It should be noted that in $\mathcal{H}$ one can ask queries with free predicate variables of any *order*; this means that we can have queries that return sets of sets of objects, and so on.

As demonstrated in (Charalambidis et al. 2013), one can define a sound and complete proof procedure for $\mathcal{H}$ and based on it to obtain a reasonably efficient implementation of the language. There are however certain aspects of the language that require further investigation. One basic property of all the higher-order predicates that can be defined in $\mathcal{H}$ is that they are *monotonic*. Intuitively, the monotonicity property states that if a predicate is true of a relation R then it is also true of every superset of R. In the above example, it is clear that if band is true of a relation B then it is also true of any band that is a superset of B. However, there are many natural higher-order predicates that are *non-monotonic*. For example, consider the predicate disconnected(G) which succeeds if its input argument, namely a graph G, is disconnected. A graph is simply a set of pairs, and therefore disconnected is a second-order predicate. Notice now that disconnected is obviously non-monotonic: given graphs G1 and G2 with $\texttt{G1} \subseteq \texttt{G2}$, it is possible that disconnected(G1) succeeds but disconnected(G2) fails.

The obvious idea in order to add non-monotonicity to $\mathcal{H}$ is to enhance the language with negation-as-failure. However, this is not as straightforward as it sounds, because even the simpler higher-order programs with negation face the well-known problem of *floundering* (Lloyd 1987). In classical logic programming a computation is said to *flounder* if at some point a goal is reached that contains only non-ground negative literals. In higher-order logic programming this problem is even more extended since a goal may additionally contain uninstantiated higher-order variables.

Example 3. Consider the following simple modification to the program of Example 1:

```
p(Q):-Q(0),not(Q(1)).
```

Intuitively, p is true of all relations that contain 0 but they do not contain 1. The query:

```
?-p(R).
```

will not behave correctly if a simple-minded implementation of negation-as-failure is followed; this is due to the fact that in order to answer the given query one has to answer the subquery not(R(1)). Since the relation R is not fully known at the point of this call, it is not obvious how the implementation should treat such a subquery.

Constructive negation (Chan 1988; 1989) bypasses the problem of floundering. The main idea can be explained by a simple example. Consider the program:

```
p(1).
p(2).
```

and the query:

```
?-not(p(X)).
```

The original idea (Chan 1988) behind constructive negation is that in order to answer a negative query that contains uninstantiated variables, the following procedure must be applied: we run the positive version of the query and we collect the solutions as a disjunction; we then return the negation of the disjunction as the answer to the original query. This idea can be extended even to the case where a query has an infinite number of answers (Chan 1989). In our example, the corresponding positive query (namely ?-p(X).) has the answers X=1 and X=2. The negation of their disjunction is the conjunction $(\texttt{X} \neq 1) \wedge (\texttt{X} \neq 2)$. Observe now that the procedure behind constructive negation returns as answers not only substitutions (as it happens in negationless logic programming) but also inequalities.

Generalizing the above idea to the higher-order setting requires the ability to express some form of inequalities regarding the elements of sets. Intuitively, we would like to express the property that some element *does not belong to a set*. For example, given the query in Example 3, a reasonable answer would be $\texttt{R} = \{0\} \cup \{\texttt{X} \mid \texttt{X} \neq 1\}$.

We start with the syntax of $\mathcal{H}$ as introduced in (Charalambidis et al. 2013) and extend it in order to allow negation in clause bodies. The new language, $\mathcal{H}_{cn}$, has only one limitation with respect to the initial $\mathcal{H}$: the existential predicate variables that appear in queries or in clause bodies of

$\mathcal{H}_{cn}$ can only be sets of terms (and not, for example, sets of sets, or sets of sets of sets, and so on). The reasons for our restriction are two-fold: first, it is not straightforward to express inequalities regarding the elements of a set of sets; and second, it would be much more difficult to express the proof procedure if the queries were not restricted (and the implementation would also face significant complications).

It turns out that $\mathcal{H}_{cn}$ is quite appropriate in expressing problems that require a generate-and-test of sets in order to reach a solution. For example, many graph problems require the enumeration and testing of subgraphs of a given graph. In this respect, the language $\mathcal{H}_{cn}$ appears to have an application domain that is similar to that of modern ASP languages, which are being used for specifying computationally difficult problems (Gelfond and Lifschitz 1988; Lifschitz 2008). Of course, the higher-order approach is completely different than the ASP one in terms of syntax and semantics, and a comparative evaluation of the two paradigms deserves further investigation.

Example 4. Consider the following program which defines the subset relation:

```
subset(P,Q):-not(non_subset(P,Q)).
non_subset(P,Q):-P(X),not(Q(X)).
```

and assume that we have a predicate `q` which is true of `0`, `1` and `2`. Then, given the query

```
?-subset(P,q).
```

the interpreter produces all the subsets of the set $\{0, 1, 2\}$.

Example 5. Consider the following program which defines the relation `twocolor(G,R)` which is true if `R` is a subset of the set of vertices of `G` that can be painted with the same color in a two-coloring of `G`.

```
twocolor(G,R):-subset(R,vertex_set(G)),
               not(non_twocolor(G,R)).

vertex_set(G)(X):-G(X,_).
vertex_set(G)(X):-G(_,X).

non_twocolor(G,R):-G(X,Y),R(X),R(Y).
non_twocolor(G,R):-G(X,Y),not(R(X)),not(R(Y)).
```

Assume that `g` is a binary predicate defining a graph. By asking the query:

```
?-twocolor(g,R).
```

the implementation will enumerate all the possible relations `R` that constitute valid two-colorings of `g`.

Example 6. The following program can be used to enumerate all the subgraphs of a given graph that are cliques:

```
clique(G,R):-subset(R,vertex_set(G)),
             not(non_clique(G,R)).

non_clique(G,R):-R(X),R(Y),not(G(X,Y)).
```

3 Syntax and Semantics

This section defines in a formal way the syntax and the semantics of the higher-order language $\mathcal{H}_{cn}$.

The Syntax of $\mathcal{H}_{cn}$

The language $\mathcal{H}_{cn}$ is in fact a more austere version of the ad-hoc language used for the examples of the previous section.

Example 7. The `subset` predicate of Example 4 can be expressed in $\mathcal{H}_{cn}$ as follows:

$$\texttt{subset} \leftarrow \lambda\texttt{P}.\lambda\texttt{Q}.\ \sim\exists\texttt{X}((\texttt{P X})\wedge\sim(\texttt{Q X}))$$

The `subset` predicate is defined by a λ-expression (which obviates the need to have the formal parameters of the predicate in the left-hand side of the definition). Moreover, in the right-hand side we have an explicit existential quantifier for the variable `X` (in Prolog, if a variable appears in the body of a clause but not in the head, then it is implicitly existentially quantified). Consider the query:

$$\leftarrow \texttt{subset R p}$$

Notice that here the variable `R` is not explicitly existentially quantified: we assume that the free variables of the goal are the ones for which the proof procedure will attempt to find bindings in order for the goal to be satisfied. If a variable appears explicitly existentially quantified in the goal, then the implementation will not produce a binding for it.

The type system of $\mathcal{H}_{cn}$ is based on two base types: o, the type of the boolean domain, and ι, the type of the domain of individuals (data objects). For example, every classical logic programming term is of type ι.

The basic difference in the types of $\mathcal{H}_{cn}$ from those of $\mathcal{H}$ is the existence of a type μ which restricts the set of predicate variables that can be existentially quantified or appear free in goal clauses (as explained in the previous section).

Definition 1. A type can either be *functional*, *argument*, or *predicate*, denoted by σ, ρ and π and defined as:

$$\begin{array}{lcl} \sigma & := & \iota \mid (\iota \rightarrow \sigma) \\ \rho & := & \iota \mid \pi \\ \pi & := & o \mid (\rho \rightarrow \pi) \end{array}$$

The subtypes μ and κ of ρ and π are defined as follows:

$$\begin{array}{lcl} \mu & := & \iota \mid \kappa \\ \kappa & := & \iota \rightarrow o \mid (\iota \rightarrow \kappa) \end{array}$$

The types μ and κ will be called *existential* and *set* types respectively.

The operator $\rightarrow$ is right-associative. A functional type that is different from ι will often be written in the form $\iota^n \rightarrow \iota$, $n \geq 1$. Similarly, a set type will often be written in the form $\iota^n \rightarrow o$, $n \geq 1$. It can be easily seen that every predicate type π can be written in the form $\rho_1 \rightarrow \cdots \rightarrow \rho_n \rightarrow o$, $n \geq 0$ (for $n = 0$ we assume that $\pi = o$). We proceed with the alphabet of $\mathcal{H}_{cn}$:

Definition 2. The *alphabet* of $\mathcal{H}_{cn}$ consists of:

1. *Predicate variables* of every predicate type π (denoted by uppercase letters such as $\mathsf{R}, \mathsf{Q}, \ldots$).
2. *Predicate constants* of every predicate type π (denoted by lowercase letters such as $\mathsf{p}, \mathsf{q}, \mathsf{r}, \ldots$).
3. *Individual variables* of type ι (denoted by uppercase letters such as $\mathsf{X}, \mathsf{Y}, \mathsf{Z}, \ldots$).

4. *Individual constants* of type ι (denoted by lowercase letters such as $\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots$).
5. *Function symbols* of every functional type $\sigma \neq \iota$ (denoted by lowercase letters such as $\mathsf{f}, \mathsf{g}, \mathsf{h}, \ldots$).
6. The following logical constant symbols: the propositional constants false and true of type o; the equality constant $\approx$ of type $\iota \to \iota \to o$; the generalized disjunction and conjunction constants $\bigvee_\pi$ and $\bigwedge_\pi$ of type $\pi \to \pi \to \pi$, for every predicate type π; the generalized inverse implication constants $\leftarrow_\pi$ of type $\pi \to \pi \to o$, for every predicate type π; the equivalence constants $\leftrightarrow_\pi$ of type $\pi \to \pi \to o$, for every predicate type π; the existential quantifiers $\exists_\mu$ of type $(\mu \to o) \to o$, for every existential type μ; the negation constant $\sim$of type $o \to o$.
7. The abstractor λ and the parentheses "(" and ")".

The set consisting of the predicate variables and the individual variables of $\mathcal{H}_{cn}$ will be called the set of *argument variables* of $\mathcal{H}_{cn}$. Argument variables will be usually denoted by V, U and their subscripted versions.

Definition 3. The set of *body expressions* of the higher-order language $\mathcal{H}_{cn}$ is recursively defined as follows:

1. Every predicate variable (respectively, predicate constant) of type π is a body expression of type π; every individual variable (respectively, individual constant) of type ι is a body expression of type ι; the propositional constants false and true are body expressions of type o.
2. If f is an n-ary function symbol and $\mathsf{E}_1, \ldots, \mathsf{E}_n$ are body expressions of type ι, then $(\mathsf{f}\ \mathsf{E}_1 \cdots \mathsf{E}_n)$ is a body expression of type ι.
3. If E_1 is a body expression of type $\rho \to \pi$ and E_2 is a body expression of type ρ, then $(\mathsf{E}_1\ \mathsf{E}_2)$ is a body expression of type π.
4. If V is an argument variable of type ρ and E is a body expression of type π, then $(\lambda \mathsf{V}.\mathsf{E})$ is a body expression of type $\rho \to \pi$.
5. If $\mathsf{E}_1, \mathsf{E}_2$ are expressions of type π, then $(\mathsf{E}_1 \bigwedge_\pi \mathsf{E}_2)$ and $(\mathsf{E}_1 \bigvee_\pi \mathsf{E}_2)$ are body expressions of type π.
6. If $\mathsf{E}_1, \mathsf{E}_2$ are expressions of type ι, then $(\mathsf{E}_1 \approx \mathsf{E}_2)$ is a body expression of type o.
7. If E is an expression of type o and V is an existential variable of type μ, then $(\exists_\mu \mathsf{V}\, \mathsf{E})$ is a body expression of type o.
8. If E is a body expression of type o, then $(\sim \mathsf{E})$ is a body expression of type o.

The type subscripts (such as μ in $\exists_\mu$) will often be omitted when they are obvious or immaterial in the foregoing discussion. Moreover, we will write $\wedge$ and $\vee$ instead of $\bigwedge_o$ and $\bigvee_o$. We will use $type(\mathsf{E})$ to denote the type of the body expression E. The notions of *free* and *bound* variables of a body expression are defined as usual. A body expression is called *closed* if it does not contain any free variables. Given an expression E, we denote by $fv(\mathsf{E})$ the set of all free variables of E. By overloading notation, we will also write $fv(S)$, where S is a set of expressions.

We will often write $\hat{\mathsf{A}}$ to denote a (possibly empty) sequence $\langle \mathsf{A}_1, \ldots, \mathsf{A}_n \rangle$ of expressions. For example we will write $(\mathsf{E}\ \hat{\mathsf{A}})$ to denote an application $(\mathsf{E}\ \mathsf{A}_1 \cdots \mathsf{A}_n)$; $(\lambda \hat{\mathsf{X}}.\mathsf{E})$ to denote $(\lambda \mathsf{X}_1. \cdots \lambda \mathsf{X}_n.\mathsf{E})$; $(\exists \hat{\mathsf{V}}\ \mathsf{E})$ to denote $(\exists \mathsf{V}_1 \cdots \exists \mathsf{V}_n\ \mathsf{E})$. When $\hat{\mathsf{A}}$ or $\hat{\mathsf{X}}$ or $\hat{\mathsf{V}}$ is empty, then the corresponding expression is considered identical to E.

A body expression of the form $(\mathsf{E}_1 \approx \mathsf{E}_2)$ will be called an *equality* while an expression of the form $\sim\exists\hat{\mathsf{V}}(\mathsf{E}_1 \approx \mathsf{E}_2)$ will be called an *inequality*; in the latter case $\hat{\mathsf{V}}$ may be empty (in which case the inequality is of the form $\sim(\mathsf{E}_1 \approx \mathsf{E}_2)$). Let $\hat{\mathsf{E}} = \langle \mathsf{E}_1, \ldots, \mathsf{E}_n \rangle$ and $\hat{\mathsf{E}}' = \langle \mathsf{E}'_1, \ldots, \mathsf{E}'_n \rangle$, where all $\mathsf{E}_i, \mathsf{E}'_i$ are of type ι, we will write $(\hat{\mathsf{E}} \approx \hat{\mathsf{E}}')$ to denote the expression $(\mathsf{E}_1 \approx \mathsf{E}'_1) \wedge \cdots \wedge (\mathsf{E}_n \approx \mathsf{E}'_n)$; if $n = 0$, then the conjunction is the constant true.

Definition 4. The set of *clausal expressions* of the higher-order language $\mathcal{H}_{cn}$ is defined as follows:

1. If p is a predicate constant of type π and E is a closed body expression of type π then $\mathsf{p} \leftarrow_\pi \mathsf{E}$ is a clausal expression of $\mathcal{H}_{cn}$, also called a *program clause*.
2. If E is a body expression of type o and each free variable in E is of type μ then $\mathsf{false} \leftarrow_o \mathsf{E}$ (usually denoted by $\leftarrow_o \mathsf{E}$ or just $\leftarrow \mathsf{E}$) is a clausal expression of $\mathcal{H}_{cn}$, also called a *goal clause*.
3. If p is a predicate constant of type π and E is a closed body expression of type π then $\mathsf{p} \leftrightarrow_\pi \mathsf{E}$ is a clausal expression of $\mathcal{H}_{cn}$, also called a *completion expression*.

All clausal expressions of $\mathcal{H}_{cn}$ have type o.

Notice that (following the tradition of first-order logic programming) we will talk about the "empty clause" which is denoted by $\Box$ and is equivalent to $\leftarrow \mathsf{true}$.

Definition 5. A program of $\mathcal{H}_{cn}$ is a finite set of program clauses of $\mathcal{H}_{cn}$.

We proceed by defining the completion of a program:

Definition 6. Let P be a program and let p be a predicate constant of type π. Then, the *completed definition* for p with respect to P is obtained as follows:

- if there exist exactly $k > 0$ program clauses of the form $\mathsf{p} \leftarrow_\pi \mathsf{E}_i$ where $i \in \{1, \ldots, k\}$ for p in P, then the completed definition for p is the expression $\mathsf{p} \leftrightarrow_\pi \mathsf{E}$, where $\mathsf{E} = \mathsf{E}_1 \bigvee_\pi \cdots \bigvee_\pi \mathsf{E}_k$.
- if there are no program clauses for p in P then the completed definition for p is the expression $\mathsf{p} \leftrightarrow_\pi \mathsf{E}$, where E is of type π and $\mathsf{E} = \lambda \hat{\mathsf{X}}.\mathsf{false}$.

The expression E on the right-hand side of the completed definition of p will be called the *completed expression* for p with respect to P.

We can now extend the notion of completion (Clark 1977; Lloyd 1987) to apply to our higher-order programs:

Definition 7. Let P be a program. Then, the *completion* $comp(\mathsf{P})$ of P is the set consisting of all the completed definitions for all predicate constants that appear in P.

The Semantics of $\mathcal{H}_{cn}$

In this section we introduce the semantics of $\mathcal{H}_{cn}$. We start by defining the meaning of types of $\mathcal{H}_{cn}$; for the predicate types we also define a corresponding partial order (which will be used in the definition of the semantics of $\mathcal{H}_{cn}$).

Definition 8. Let D be a non-empty set. Then the meanings of the types of our language can be specified as follows:

- $[\![\iota]\!]_D = D$;
- $[\![\iota^n \to \iota]\!]_D = D^n \to D$;
- $[\![o]\!]_D = \{false, true\}$. A partial order $\sqsubseteq_o$ can be defined on the set $[\![o]\!]_D$ as follows: $false \sqsubseteq_o true$, $false \sqsubseteq_o false$ and $true \sqsubseteq_o true$;
- $[\![\rho \to \pi]\!]_D = [\![\rho]\!]_D \to [\![\pi]\!]_D$. A partial order $\sqsubseteq_{\rho\to\pi}$ can be defined as follows: for all $f, g \in [\![\rho \to \pi]\!]_D$, $f \sqsubseteq_{\rho\to\pi} g$ if and only if $f(d) \sqsubseteq_\pi g(d)$, for all $d \in [\![\rho]\!]_D$.

It can be easily shown that for every predicate type π, $[\![\pi]\!]_D$ is a complete lattice, and therefore for every subset of $[\![\pi]\!]_D$ there exists a *lub* function (denoted by $\bigsqcup_\pi$) and a *glb* function (denoted by $\bigsqcap_\pi$).

The notions of *interpretation* and *state* can be defined in a standard way:

Definition 9. An interpretation I of $\mathcal{H}_{cn}$ consists of:

1. a nonempty set D, called the *domain* of I
2. an assignment to each individual constant symbol c, of an element $I(\mathsf{c}) \in D$
3. an assignment to each predicate constant p of type π, of an element $I(\mathsf{p}) \in [\![\pi]\!]_D$
4. an assignment to each function symbol f of type $\iota^n \to \iota$, of a function $I(\mathsf{f}) \in D^n \to D$.

Definition 10. Let D be a nonempty set. Then, a *state s of $\mathcal{H}_{cn}$ over D* is a function that assigns to each argument variable V of type ρ of $\mathcal{H}_{cn}$ an element $s(\mathsf{V}) \in [\![\rho]\!]_D$.

In the following, $s[\mathsf{V}/d]$ is used to denote a state that is identical to s the only difference being that the new state assigns to V the value d.

Definition 11. Let I be an interpretation of $\mathcal{H}_{cn}$, let D be the domain of I and let s be a state over D. Then, the semantics of expressions of $\mathcal{H}_{cn}$ with respect to I and s is defined as follows:

1. $[\![\mathsf{false}]\!]_s(I) = false$
2. $[\![\mathsf{true}]\!]_s(I) = true$
3. $[\![\mathsf{c}]\!]_s(I) = I(\mathsf{c})$, for every individual constant c
4. $[\![\mathsf{p}]\!]_s(I) = I(\mathsf{p})$, for every predicate constant p
5. $[\![\mathsf{V}]\!]_s(I) = s(\mathsf{V})$, for every argument variable V
6. $[\![(\mathsf{f}\ \mathsf{E}_1 \cdots \mathsf{E}_n)]\!]_s(I) = I(\mathsf{f})\ [\![\mathsf{E}_1]\!]_s(I) \cdots [\![\mathsf{E}_n]\!]_s(I)$, for every n-ary function symbol f
7. $[\![(\mathsf{E}_1\mathsf{E}_2)]\!]_s(I) = [\![\mathsf{E}_1]\!]_s(I)([\![\mathsf{E}_2]\!]_s(I))$
8. $[\![(\lambda\mathsf{V}.\mathsf{E})]\!]_s(I) = \lambda d.[\![\mathsf{E}]\!]_{s[\mathsf{V}/d]}(I)$, where d ranges over $[\![type(\mathsf{V})]\!]_D$
9. $[\![(\mathsf{E}_1 \bigvee_\pi \mathsf{E}_2)]\!]_s(I) = \bigsqcup_\pi\{[\![\mathsf{E}_1]\!]_s(I), [\![\mathsf{E}_2]\!]_s(I)\}$, where $\bigsqcup_\pi$ is the least upper bound function on $[\![\pi]\!]_D$
10. $[\![(\mathsf{E}_1 \bigwedge_\pi \mathsf{E}_2)]\!]_s(I) = \bigsqcap_\pi\{[\![\mathsf{E}_1]\!]_s(I), [\![\mathsf{E}_2]\!]_s(I)\}$, where $\bigsqcap_\pi$ is the greatest lower bound function on $[\![\pi]\!]_D$
11. $[\![(\mathsf{E}_1 \approx \mathsf{E}_2)]\!]_s(I) = \begin{cases} true, & \text{if } [\![\mathsf{E}_1]\!]_s(I) = [\![\mathsf{E}_2]\!]_s(I) \\ false, & \text{otherwise} \end{cases}$
12. $[\![(\exists\mathsf{V}\ \mathsf{E})]\!]_s(I) = \begin{cases} true, & \text{if there exists } d \in [\![type(\mathsf{V})]\!]_D \\ & \text{such that } [\![\mathsf{E}]\!]_{s[\mathsf{V}/d]}(I) = true \\ false, & \text{otherwise} \end{cases}$
13. $[\![(\sim\mathsf{E})]\!]_s(I) = \begin{cases} true, & \text{if } [\![\mathsf{E}]\!]_s(I) = false \\ false, & \text{if } [\![\mathsf{E}]\!]_s(I) = true \end{cases}$
14. $[\![\mathsf{p} \leftarrow_\pi \mathsf{E}]\!]_s(I) = \begin{cases} true, & \text{if } [\![\mathsf{E}]\!]_s(I) \sqsubseteq_\pi I(\mathsf{p}) \\ false, & \text{otherwise} \end{cases}$
15. $[\![\mathsf{p} \leftrightarrow_\pi \mathsf{E}]\!]_s(I) = \begin{cases} true, & \text{if } [\![\mathsf{E}]\!]_s(I) = I(\mathsf{p}) \\ false, & \text{otherwise} \end{cases}$
16. $[\![\leftarrow \mathsf{E}]\!]_s(I) = \begin{cases} true, & \text{if } [\![\mathsf{E}]\!]_s(I) = false \\ false, & \text{otherwise} \end{cases}$

Definition 12. Let S be a set of expressions of type o and let I be an interpretation of $\mathcal{H}_{cn}$. We say that I is a *model* of S if for every $\mathsf{E} \in S$ and for every state s over the domain of I, $[\![\mathsf{E}]\!]_s(I) = true$.

Clark's Equality Theory can also be added to our fragment as follows:

Definition 13. Let I be an interpretation of $\mathcal{H}_{cn}$. We will say that I satisfies *Clark's Equality Theory (CET)* if for all states s over the domain of I, all the following hold:

1. $[\![(\mathsf{f}\ \hat{\mathsf{E}}) \approx (\mathsf{f}\ \hat{\mathsf{E}}')]\!]_s(I) = [\![\hat{\mathsf{E}} \approx \hat{\mathsf{E}}']\!]_s(I)$ for every n-ary function symbol f and for all $\hat{\mathsf{E}} = \langle \mathsf{E}_1, \ldots, \mathsf{E}_n\rangle$ and for all $\hat{\mathsf{E}}' = \langle \mathsf{E}'_1, \ldots, \mathsf{E}'_n\rangle$
2. $[\![\mathsf{c} \approx \mathsf{d}]\!]_s(I) = false$ for all different individual constants c and d
3. $[\![(\mathsf{f}\ \hat{\mathsf{E}}) \approx \mathsf{c}]\!]_s(I) = false$ for every function symbol f and every individual constant c
4. $[\![(\mathsf{f}\ \hat{\mathsf{E}}) \approx (\mathsf{g}\ \hat{\mathsf{E}}')]\!]_s(I) = false$ for all different function symbols f and g and all $\hat{\mathsf{E}}$ and $\hat{\mathsf{E}}'$
5. $[\![\mathsf{X} \approx \mathsf{E}]\!]_s(I) = false$ if $\mathsf{X} \in fv(\mathsf{E})$ and $\mathsf{X} \neq \mathsf{E}$

In the rest of this paper we will often talk about "models of the completion *comp*(P) of a program P"; in every such case we will implicitly assume that our models also satisfy Clark's Equality Theory.

4 Proof Procedure

In this section we define a proof procedure for $\mathcal{H}_{cn}$ and discuss its implementation. As in the case of classical logic programming, substitutions and unifiers also play an important role in our setting. However, they need to be significantly extended, as discussed below.

Substitutions and Unifiers

In classical logic programming, a substitution is a mapping from variables to terms. Since in our case queries may contain uninstantiated variables of set types, the notion of substitution must be extended to assign to these variables expressions that represent sets. In the following we define the notion of *basic expression* which generalizes the notion of term from classical logic programming. Intuitively, a basic element is either a term or a set of terms.

Definition 14. The set of *basic expressions* of $\mathcal{H}_{cn}$ of type μ is defined recursively as follows:

1. Every expression of $\mathcal{H}_{cn}$ of type ι is a basic expression of type ι.

2. Every predicate variable of type κ is a basic expression of type κ.
3. If $\mathsf{E}_1, \mathsf{E}_2$ are basic expressions of type κ then $\mathsf{E}_1 \bigvee_\kappa \mathsf{E}_2$ and $\mathsf{E}_1 \bigwedge_\kappa \mathsf{E}_2$ are basic expressions of type κ.
4. The expressions of the following form, are basic expressions of type $\iota^n \to o$:
   - $\lambda\hat{\mathsf{X}}.\exists\hat{\mathsf{V}}(\hat{\mathsf{X}} \approx \hat{\mathsf{A}})$
   - $\lambda\hat{\mathsf{X}}.\ \sim\exists\hat{\mathsf{V}}(\hat{\mathsf{X}} \approx \hat{\mathsf{A}})$

   where $\hat{\mathsf{X}} = \langle \mathsf{X}_1, \ldots, \mathsf{X}_n\rangle$, $\hat{\mathsf{A}} = \langle \mathsf{A}_1, \ldots, \mathsf{A}_n\rangle$, each X_i is a variable of type ι, each $\hat{\mathsf{A}}_i$ is a basic expressions of type ι and $\hat{\mathsf{V}}$ is a possibly empty subset of $fv(\hat{\mathsf{A}})$.

Example 8. The basic element $\lambda\mathtt{X}.\ \sim\exists\mathtt{Y}(\mathtt{X} \approx \mathtt{Y})$ represents the empty set while $\lambda\mathtt{X}.(\mathtt{X} \approx \mathtt{a}) \bigvee \lambda\mathtt{X}.(\mathtt{X} \approx (\mathtt{f\ b}))$ represents the set $\{\mathtt{a}, (\mathtt{f\ b})\}$. The basic element $\lambda\mathtt{X}.\exists\mathtt{Y}(\mathtt{X} \approx (\mathtt{f\ Y}))$ is the set that contains all the terms of the form $(\mathtt{f}\cdots)$. The basic element $\lambda\mathtt{X}.\ \sim\exists\mathtt{Y}(\mathtt{X} \approx (\mathtt{f\ Y}))$ is the set that contains all the terms that are not of the form $(\mathtt{f}\cdots)$. The basic element $\lambda\mathtt{X}.\exists\mathtt{Y}(\mathtt{X} \approx (\mathtt{f\ Y})) \bigwedge \lambda\mathtt{X}.\ \sim(\mathtt{X} \approx (\mathtt{f\ b}))$ is the set that contains all the terms of the form $(\mathtt{f}\cdots)$ with the exception of $(\mathtt{f\ b})$.

The definition of a substitution can be extended:

Definition 15. A substitution θ is a finite set of the form $\{\mathsf{V}_1/\mathsf{E}_1, \ldots, \mathsf{V}_n/\mathsf{E}_n\}$ where V_i are different argument variables of $\mathcal{H}_{cn}$ and each E_i is a body expression of $\mathcal{H}_{cn}$ having the same type as V_i. We write $dom(\theta) = \{\mathsf{V}_1, \ldots, \mathsf{V}_n\}$ and $range(\theta) = \{\mathsf{E}_1, \ldots, \mathsf{E}_n\}$. A substitution is called *basic* if all E_i are basic expressions. A substitution is called *zero-order* if the type of V_i is ι for all $i \in \{1, \ldots, n\}$. The substitution corresponding to the empty set will be called the *identity substitution* and will be denoted by ϵ.

In the following definition (and also in the rest of the paper) we assume that if expressions $\mathsf{E}_1, \ldots, \mathsf{E}_n$ occur in a certain mathematical context, then in these expressions all bound variables are chosen to be different from the free variables. This is called the *the bound variable convention* in (Barendregt 1984, pages 26–27) (and is actually equivalent to the α-renaming operation of λ-calculus).

Definition 16. Let θ be a substitution and let E be a body expression. Then, $\mathsf{E}\theta$ is an expression obtained as follows:

- $\mathsf{E}\theta = \mathsf{E}$, if E is false, true, c, or p
- $\mathsf{V}\theta = \theta(\mathsf{V})$ if $\mathsf{V} \in dom(\theta)$; otherwise, $\mathsf{V}\theta = \mathsf{V}$
- $(\mathsf{f}\ \mathsf{E}_1 \cdots \mathsf{E}_n)\theta = (\mathsf{f}\ \mathsf{E}_1\theta \cdots \mathsf{E}_n\theta)$
- $(\mathsf{E}_1\ \mathsf{E}_2)\theta = (\mathsf{E}_1\theta\ \mathsf{E}_2\theta)$
- $(\lambda\mathsf{V}.\mathsf{E})\theta = (\lambda\mathsf{V}.\mathsf{E}\theta)$
- $(\mathsf{E}_1 \bigvee_\pi \mathsf{E}_2)\theta = (\mathsf{E}_1\theta \bigvee_\pi \mathsf{E}_2\theta)$
- $(\mathsf{E}_1 \bigwedge_\pi \mathsf{E}_2)\theta = (\mathsf{E}_1\theta \bigwedge_\pi \mathsf{E}_2\theta)$
- $(\mathsf{E}_1 \approx \mathsf{E}_2)\theta = (\mathsf{E}_1\theta \approx \mathsf{E}_2\theta)$
- $(\exists\mathsf{V}\ \mathsf{E})\theta = (\exists\mathsf{V}\ \mathsf{E}\theta)$
- $(\sim\mathsf{E})\theta = \sim\mathsf{E}\theta$

In the following, if $\hat{\mathsf{E}} = \langle \mathsf{E}_1, \ldots, \mathsf{E}_n\rangle$, we will write $\hat{\mathsf{E}}\theta$ to denote the sequence $\langle \mathsf{E}_1\theta, \ldots, \mathsf{E}_n\theta\rangle$.

Definition 17. Let $\theta = \{\mathsf{V}_1/\mathsf{E}_1, \ldots, \mathsf{V}_m/\mathsf{E}_m\}$ and $\sigma = \{\mathsf{V}'_1/\mathsf{E}'_1, \ldots, \mathsf{V}'_m/\mathsf{E}'_m\}$ be substitutions. Then the composition $\theta\sigma$ of θ and σ is the substitution obtained from the set $\{\mathsf{V}_1/\mathsf{E}_1\sigma, \ldots, \mathsf{V}_m/\mathsf{E}_m\sigma, \mathsf{V}'_1/\mathsf{E}'_1, \ldots, \mathsf{V}_n/\mathsf{E}'_n\}$ by deleting any $\mathsf{V}_i/\mathsf{E}_i\sigma$ for which $\mathsf{V}_i = \mathsf{E}_i\sigma$ and deleting any $\mathsf{V}'_j/\mathsf{E}'_j$ for which $\mathsf{V}'_j \in dom(\theta)$.

The notions of "unifier" and "most general unifier" are the same as in classical first-order logic programming.

Definition 18. Let S be a set of terms of $\mathcal{H}_{cn}$ (ie., expressions of type ι). A zero-order substitution θ will be called a *unifier* of the expressions in S if the set $S\theta = \{\mathsf{E}\theta \mid \mathsf{E} \in S\}$ is a singleton. The zero-order substitution θ will be called a *most general unifier of* S (denoted by $mgu(S)$), if for every unifier σ of the expressions in S, there exists a zero-order substitution γ such that $\sigma = \theta\gamma$.

The most general unifier of a set of terms S can be computed using the unification algorithm described in (Lloyd 1987). The output of the unification algorithm is denoted by $unify(S)$.

The following *substitution lemma* can easily be established by structural induction on E:

Lemma 1 (Substitution Lemma). *Let I be an interpretation of $\mathcal{H}_{cn}$ and let s be a state over the domain of I. Let θ be a basic substitution and E be a body expression. Then, $[\![\mathsf{E}\theta]\!]_s(I) = [\![\mathsf{E}]\!]_{s'}(I)$, where $s'(\mathsf{V}) = [\![\theta(\mathsf{V})]\!]_s(I)$ if $\mathsf{V} \in dom(\theta)$ and $s'(\mathsf{V}) = s(\mathsf{V})$, otherwise.*

The above lemma will be used in the proof of the soundness of the resolution proof procedure that follows.

The Procedure

In this subsection we describe the actual proof procedure for $\mathcal{H}_{cn}$. Intuitively, we could say that our approach generalizes to the higher-order case the constructive negation procedure proposed by D. Chan in (Chan 1989). In order to define the overall proof procedure we utilize three definitions, described as follows:

- The central definition of our procedure is Definition 20 which defines the *single-step derivation* of goals. Intuitively, the single-step derivation is an extension of the single-step derivation of classical logic programming (see for example (Lloyd 1987, pages 50–41)) in order to apply to higher-order logic programming and to constructive negation. Notice that the single-step derivation is the only one among the three that may produce substitutions (whose final composition will lead to the computed answer for the initial goal).
- The *reduction* (Definition 21) is the simplest among the three definitions. Roughly speaking it corresponds to certain simple computational steps that advance the evaluation of the goal. For example, the β-reduction step for a λ-expression belongs to this reduction relation. Another simple reduction step is the replacement of a predicate constant with its defining expression in the program.
- The *negative reduction* (Definition 22) is the most complicated among the three. Intuitively, it corresponds to what D. Chan in (Chan 1989) refers to as "the negation of

answers" and the "normalization of answers". The definition is more complicated in our case since it applies to the more demanding setting of higher-order logic programming.

One difference between the proof procedures for constructive negation (such as for example the one given in (Chan 1989)) and the traditional SLD-resolution for negationless logic programming, is the following: in classical SLD-resolution, a derivation is successful if we reach the empty goal; however, in constructive negation, a derivation may be successful if we reach a goal that consists of inequalities that can not be further simplified. Therefore, the termination condition for constructive negation is different than that of SLD-resolution. Moreover, an *answer* in classical resolution is simply a substitution for some of the variables of the goal. In constructive negation, an answer is a pair consisting of a substitution and of a conjunction of inequalities (intuitively, the inequalities that remain at the end of the derivation)[1]. In the higher-order case that we are examining, an answer is also a pair, the only difference being that the domain of the substitution may contain higher-order variables.

Before we present the actual definitions, we need the following categorization of inequalities, which is pretty standard in the area of constructive negation (see for example (Chan 1989)):

Definition 19. An inequality $\sim\exists\hat{\mathsf{V}}(\mathsf{E}_1 \approx \mathsf{E}_2)$ is considered

- *valid* if E_1 and E_2 cannot be unified;
- *unsatisfiable* if there is a substitution θ that unifies E_1 and E_2 and contains only bindings of variables in $\hat{\mathsf{V}}$;
- *satisfiable* if it is not unsatisfiable.

An inequality will be called *primitive* if it is satisfiable, non valid and either E_1 or E_2 is a variable.

In the following definition we will refer to a conjunction $\mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_n$ of expressions. We will assume that each A_i is not a conjunction itself. The conjunction of a single expression is the expression itself and the conjunction of zero expressions is the expression true.

Definition 20. Let P a program and let G_k and G_{k+1} be goal clauses. Moreover, let G_k be a conjunction $\leftarrow \mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_n$, where each A_i is a body expression of type o. Let A_i be one of the $\mathsf{A}_1, \ldots, \mathsf{A}_n$ and let us call it the *selected expression*. Let $\mathsf{A}' = \mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_{i-1} \wedge \mathsf{A}_{i+1} \wedge \cdots \wedge \mathsf{A}_n$. We say that G_{k+1} is *derived in one step* from G_k using θ (denoted as $\mathsf{G}_k \xrightarrow{\theta} \mathsf{G}_{k+1}$) if one of the following conditions applies:

1. if A_i is true and $n > 1$ then $\mathsf{G}_{k+1} = \leftarrow \mathsf{A}'$ is derived from G_k using $\theta = \epsilon$;
2. if A_i is $(\mathsf{E}_1 \vee \mathsf{E}_2)$ then $\mathsf{G}_{k+1} = \leftarrow \mathsf{A}_1 \wedge \cdots \wedge \mathsf{E}_j \wedge \cdots \wedge \mathsf{A}_n$ is derived from G_k using $\theta = \epsilon$ where $j \in \{1, 2\}$;
3. if A_i is $(\exists \mathsf{V}\ \mathsf{E})$ then $\mathsf{G}_{k+1} = \leftarrow \mathsf{A}_1 \wedge \cdots \wedge \mathsf{E} \wedge \cdots \wedge \mathsf{A}_n$ is derived[2] from G_k using $\theta = \epsilon$;
4. if $\mathsf{A}_i \rightsquigarrow \mathsf{A}_i'$ then $\mathsf{G}_{k+1} = \leftarrow \mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_i' \wedge \cdots \wedge \mathsf{A}_n$ is derived from G_k using $\theta = \epsilon$;
5. if A_i is $(\mathsf{E}_1 \approx \mathsf{E}_2)$ then $\mathsf{G}_{k+1} = \leftarrow \mathsf{A}'\theta$ is derived from G_k using $\theta = mgu(\mathsf{E}_1, \mathsf{E}_2)$;
6. if A_i is $(\mathsf{R}\ \hat{\mathsf{E}})$ and R is a predicate variable of type κ then $\mathsf{G}_{k+1} = \leftarrow \mathsf{A}'\theta$ is derived from G_k using $\theta = \{\mathsf{R}/(\lambda\hat{\mathsf{X}}.(\hat{\mathsf{X}} \approx \hat{\mathsf{E}}) \bigvee_\kappa \mathsf{R}')\}$ where R' is a fresh[3] predicate variable of type κ;
7. if A_i is $\sim\exists\hat{\mathsf{V}}\ \mathsf{E}$ and A_i is negatively-reduced to A_i' then $\mathsf{G}_{k+1} = \leftarrow \mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_i' \wedge \cdots \wedge \mathsf{A}_n$ is derived from G_k using $\theta = \epsilon$;
8. if A_i is $\sim\exists\hat{\mathsf{V}}(\mathsf{R}\ \hat{\mathsf{E}})$ and R is a predicate variable of type κ and $\mathsf{R} \notin \hat{\mathsf{V}}$ then $\mathsf{G}_{k+1} = \leftarrow \mathsf{A}'\theta$ is derived from G_k using $\theta = \{\mathsf{R}/(\lambda\hat{\mathsf{X}}.\ \sim\exists\hat{\mathsf{V}}(\hat{\mathsf{X}} \approx \hat{\mathsf{E}}) \bigwedge_\kappa \mathsf{R}')\}$ where R' is a fresh predicate variable of type κ;
9. if A_i is $\sim\exists\hat{\mathsf{V}}\ \sim(\mathsf{R}\ \hat{\mathsf{E}})$ and R is a predicate variable of type κ and $\mathsf{R} \notin \hat{\mathsf{V}}$ then $\mathsf{G}_{k+1} = \leftarrow \mathsf{A}'\theta$ is derived from G_k using $\theta = \{\mathsf{R}/(\lambda\hat{\mathsf{X}}.\exists\hat{\mathsf{V}}(\hat{\mathsf{X}} \approx \hat{\mathsf{E}}) \bigvee_\kappa \mathsf{R}')\}$ where R' is a fresh predicate variable of type κ.

Definition 21. Let P be a program and E, E' be body expressions of type o. We say that E is reduced (wrt. to P) to E' (denoted as $\mathsf{E} \rightsquigarrow \mathsf{E}'$) if one of the following conditions applies:

1. $\mathsf{p}\ \hat{\mathsf{A}} \rightsquigarrow \mathsf{E}\ \hat{\mathsf{A}}$, where E is the completed expression for p with respect to P;
2. $(\lambda\mathsf{X}.\mathsf{E})\ \mathsf{B}\ \hat{\mathsf{A}} \rightsquigarrow \mathsf{E}\{\mathsf{X}/\mathsf{B}\}\ \hat{\mathsf{A}}$;
3. $(\mathsf{E}_1 \bigvee_\pi \mathsf{E}_2)\ \hat{\mathsf{A}} \rightsquigarrow (\mathsf{E}_1\ \hat{\mathsf{A}}) \vee_o (\mathsf{E}_2\ \hat{\mathsf{A}})$;
4. $(\mathsf{E}_1 \bigwedge_\pi \mathsf{E}_2)\ \hat{\mathsf{A}} \rightsquigarrow (\mathsf{E}_1\ \hat{\mathsf{A}}) \wedge_o (\mathsf{E}_2\ \hat{\mathsf{A}})$.

Definition 22. Let P be a program and B, B' be body expressions where $\mathsf{B} = \sim\exists\hat{\mathsf{U}}(\mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_n)$ and each A_i is a body expression except from conjunction. Let A_i be the selected expression and $\mathsf{A}' = \mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_{i-1} \wedge \mathsf{A}_{i+1} \wedge \cdots \wedge \mathsf{A}_n$. Then, we say that B is *negatively-reduced* to B' if one of the following conditions applies:

1. if A_i is false then $\mathsf{B}' = \mathsf{true}$;
2. if A_i is true and $n = 1$ then $\mathsf{B}' = \mathsf{false}$ else $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}\ \mathsf{A}'$;
3. if A_i is $(\mathsf{E}_1 \vee \mathsf{E}_2)$ then $\mathsf{B}' = \mathsf{B}_1' \wedge \mathsf{B}_2'$ where $\mathsf{B}_j' = \sim\exists\hat{\mathsf{U}}(\mathsf{A}_1 \wedge \cdots \wedge \mathsf{E}_j \wedge \cdots \wedge \mathsf{A}_n)$, $j \in \{1, 2\}$;
4. if A_i is $(\exists\mathsf{V}\ \mathsf{E})$ then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}\mathsf{V}(\mathsf{A}_1 \wedge \cdots \wedge \mathsf{E} \wedge \cdots \wedge \mathsf{A}_n)$;[4]
5. if $\mathsf{A}_i \rightsquigarrow \mathsf{A}_i'$ then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}(\mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_i' \wedge \cdots \wedge \mathsf{A}_n)$;
6. if A_i is $(\mathsf{E}_1 \approx \mathsf{E}_2)$, then
   (a) if $\sim\exists\hat{\mathsf{U}}(\mathsf{E}_1 \approx \mathsf{E}_2)$ is valid then $\mathsf{B}' = \mathsf{true}$;
   (b) if $\sim\exists\hat{\mathsf{U}}(\mathsf{E}_1 \approx \mathsf{E}_2)$ is non-valid and neither E_1 nor E_2 is a variable then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}(\mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_{i-1} \wedge (\hat{\mathsf{X}} \approx \hat{\mathsf{X}}\theta) \wedge \mathsf{A}_{i+1} \wedge \cdots \wedge \mathsf{A}_n)$ where $\theta = unify(\mathsf{E}_1, \mathsf{E}_2)$ and $\hat{\mathsf{X}} = dom(\theta)$;

[1] Actually, in (Chan 1989) the substitution is combined with the inequalities so as to obtain a sequence consisting of equalities and inequalities.

[2] Recall that by the Bound Variable Convention, V does not appear free in A'.

[3] This means a variable that has not appeared in goals and substitutions until this point of the derivation.

[4] Recall that by the Bound Variable Convention, V does not appear free in A'.

(c) if $\sim\exists\hat{\mathsf{U}}(\mathsf{E}_1 \approx \mathsf{E}_2)$ is unsatisfiable and either E_1 or E_2 is a variable in $\hat{\mathsf{U}}$, then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}(\mathsf{A}'\theta)$ where $\theta = \{\mathsf{X}/\mathsf{E}\}$ and X is the one expression that is a variable in $\hat{\mathsf{U}}$ and E is the other;

(d) if $\sim \exists\hat{\mathsf{U}}(\mathsf{E}_1 \approx \mathsf{E}_2)$ is primitive and $n > 1$ then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}_1\ \mathsf{A}_i \vee \exists\hat{\mathsf{U}}_1(\mathsf{A}_i \wedge \sim\exists\hat{\mathsf{U}}_2\ \mathsf{A}')$ where $\hat{\mathsf{U}}_1$ are the variables in $\hat{\mathsf{U}}$ that are free in A_i and $\hat{\mathsf{U}}_2$ the variables in $\hat{\mathsf{U}}$ not in $\hat{\mathsf{U}}_1$;

7. if A_i is $(\mathsf{R}\ \hat{\mathsf{E}})$ and R is a predicate variable, then

(a) if $\mathsf{R} \in \hat{\mathsf{U}}$ then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}'(\mathsf{A}'\theta)$ where substitution $\theta = \{\mathsf{R}/(\lambda\mathsf{X}.(\mathsf{X} \approx \mathsf{E}) \bigvee_\kappa \mathsf{R}')\}$, R' is a predicate variable of the same type with R and $\hat{\mathsf{U}}'$ is the same with $\hat{\mathsf{U}}$ but the variable R has been replaced with R';

(b) if $\mathsf{R} \notin \hat{\mathsf{U}}$ and $n > 1$ then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}_1\ \mathsf{A}_i \vee \exists\hat{\mathsf{U}}_1(\mathsf{A}_i \wedge \sim\exists\hat{\mathsf{U}}_2\ \mathsf{A}') \wedge \mathsf{B}$ where $\hat{\mathsf{U}}_1$ are the variables in $\hat{\mathsf{U}}$ that are free in A_i and $\hat{\mathsf{U}}_2$ the variables in $\hat{\mathsf{U}}$ not in $\hat{\mathsf{U}}_1$;

8. if A_i is $\sim\exists\hat{\mathsf{V}}\ \mathsf{E}$ and A_i is negatively-reduced to A'_i then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}(\mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}'_i \wedge \cdots \wedge \mathsf{A}_n)$;

9. if A_i is a primitive inequality $\sim\exists\hat{\mathsf{V}}(\mathsf{E}_1 \approx \mathsf{E}_2)$ then

(a) if A_i contains free variables in $\hat{\mathsf{U}}$ and A' is a conjunction of primitive inequalities then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}\ \mathsf{A}'$;

(b) if A_i does not contain any free variables in $\hat{\mathsf{U}}$ then $\mathsf{B}' = \exists\hat{\mathsf{V}}(\mathsf{E}_1 \approx \mathsf{E}_2) \vee \sim\exists\hat{\mathsf{U}}\ \mathsf{A}'$;

10. if A_i is $\sim\exists\hat{\mathsf{V}}(\mathsf{R}\ \hat{\mathsf{E}})$ and $\mathsf{R} \notin \hat{\mathsf{V}}$ is a predicate variable, then

(a) if $\mathsf{R} \in \hat{\mathsf{U}}$ then $\mathsf{B}' = \sim \exists\hat{\mathsf{U}}'(\mathsf{A}'\theta)$ where substitution $\theta = \{\mathsf{R}/(\lambda\mathsf{X}.\ \sim\exists\hat{\mathsf{V}}(\mathsf{X} \approx \mathsf{E}) \bigwedge_\kappa \mathsf{R}')\}$, R' is a predicate variable of the same type κ with R and $\hat{\mathsf{U}}'$ is the same with $\hat{\mathsf{U}}$ but the variable R has been replaced with R';

(b) if $\mathsf{R} \notin \hat{\mathsf{U}}$ and $n > 1$ then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}_1\ \mathsf{A}_i \vee \exists\hat{\mathsf{U}}_1(\mathsf{A}_i \wedge \sim\exists\hat{\mathsf{U}}_2\ \mathsf{A}') \wedge \mathsf{B}$ where $\hat{\mathsf{U}}_1$ are the variables in $\hat{\mathsf{U}}$ that are free in A_i and $\hat{\mathsf{U}}_2$ the variables in $\hat{\mathsf{U}}$ not in $\hat{\mathsf{U}}_1$;

(c) if $\mathsf{R} \notin \hat{\mathsf{U}}$, $n = 1$ and $\hat{\mathsf{V}}$ is nonempty then $\mathsf{B}' = \exists\hat{\mathsf{V}}\ \sim\exists\hat{\mathsf{U}}(\sim(\mathsf{R}\ \hat{\mathsf{E}}) \wedge \sim\exists\hat{\mathsf{V}}'(\mathsf{R}\ \mathsf{E}'))$;

11. if A_i is $\sim\exists\hat{\mathsf{V}}\ \sim(\mathsf{R}\ \hat{\mathsf{E}})$ and R is a predicate variable and $\mathsf{R} \notin \hat{\mathsf{V}}$, then

(a) if $\mathsf{R} \in \hat{\mathsf{U}}$ then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}'(\mathsf{A}'\theta)$ where substitution $\theta = \{\mathsf{R}/(\lambda\mathsf{X}.\exists\hat{\mathsf{V}}(\mathsf{X} \approx \mathsf{E}) \bigvee_\kappa \mathsf{R}')\}$, R' is a predicate variable of the same type κ with R and $\hat{\mathsf{U}}'$ is the same with $\hat{\mathsf{U}}$ but the variable R has been replaced with R';

(b) if $\mathsf{R} \notin \hat{\mathsf{U}}$ and $n > 1$ then $\mathsf{B}' = \sim\exists\hat{\mathsf{U}}_1\ \mathsf{A}_i \vee \exists\hat{\mathsf{U}}_1(\mathsf{A}_i \wedge \sim\exists\hat{\mathsf{U}}_2\ \mathsf{A}') \wedge \mathsf{B}$;

(c) if $\mathsf{R} \notin \hat{\mathsf{U}}$, $n = 1$ and $\hat{\mathsf{V}}$ is nonempty then $\mathsf{B}' = \exists\hat{\mathsf{V}}\ \sim\exists\hat{\mathsf{U}}((\mathsf{R}\ \hat{\mathsf{E}}) \wedge \sim\exists\hat{\mathsf{V}}'\ \sim(\mathsf{R}\ \mathsf{E}'))$.

One can easily check that all the three above definitions preserve the main property of $\mathcal{H}_{cn}$ goals, namely that all the free variables in them have existential types.

We proceed by defining the notion of a *multi-step derivation* and by characterizing when a multi-step derivation is successful and the form of the computed answer. We first define the notions of terminal and primitive goals:

Definition 23. Let P be a program and $\mathsf{G} = \leftarrow \mathsf{A}$ be a goal clause. Then, G will be called *terminal* if no goals can be derived in one step from G. The goal G is called *primitive* if A is a (possibly empty) conjunction of primitive inequalities.

Notice that every primitive goal is also terminal. Notice also that the empty goal $\square$ is primitive.

Definition 24. Let $\mathsf{G} = \leftarrow \mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_n$ be a primitive goal and S be a set of variables. We say that G_S is a primitive goal obtained by restricting G to S when for each inequality A_i in G_S every free variable of A_i is in S.

Definition 25. Let P be a program and G be a goal. Assume that $\mathsf{P} \cup \{\mathsf{G}\}$ has a finite derivation $\mathsf{G}_0 = \mathsf{G}, \mathsf{G}_1, \ldots, \mathsf{G}_n$ with basic substitutions $\theta_1, \ldots, \theta_n$, such that the goal G_n is primitive. Then we will say that $\mathsf{P} \cup \{\mathsf{G}\}$ has a successful derivation of length n with primitive goal G_n and using basic substitution $\theta = \theta_1 \cdots \theta_n$.

Notice that it can easily be verified that the composition of basic substitutions is indeed a basic substitution as implied by the above definition.

We can now define the important notion of *computed answer*:

Definition 26. Let P be a program, G be a goal and assume that $\mathsf{P} \cup \{\mathsf{G}\}$ has a successful derivation with primitive goal G' and basic substitution θ. Then (σ, G'') is a *computed answer* for $\mathsf{P} \cup \{\mathsf{G}\}$ where σ is the basic substitution obtained by restricting θ to the free variables of G and G'' is the primitive goal G' restricted to the free variables of G and the variables in $fv(range(\sigma))$.

Example 9. Consider the following simple definition for the predicate `q`

$$\mathtt{q} \leftarrow \lambda\mathtt{Z}_1\,.\,\lambda\mathtt{Z}_2\,.\,(\mathtt{Z}_1 \approx \mathtt{a}) \wedge (\mathtt{Z}_2 \approx \mathtt{b})$$

that holds only for the tuple `(a, b)`. Then, consider the goal

$$\leftarrow (\mathtt{R}\ \mathtt{X}) \wedge \sim(\mathtt{q}\ \mathtt{X}\ \mathtt{Y})$$

that requests bindings for the variables `R`, `X` and `Y`.

We will underline the selected expression in each step when it is not obvious. We will also omit the substitution in a step if the rule used produces an identity substitution.

0. $\leftarrow \underline{(\mathtt{R}\ \mathtt{X})} \wedge \sim(\mathtt{q}\ \mathtt{X}\ \mathtt{Y})$
1. $\leftarrow \sim(\underline{\mathtt{q}\ \mathtt{X}\ \mathtt{Y}})$
   using rule 20.6 and $\theta_1 = \{\mathtt{R}/\lambda\mathtt{Z}.(\mathtt{Z} \approx \mathtt{X}) \bigvee \mathtt{R}'\}$
2. $\leftarrow \sim(\underline{(\lambda\mathtt{Z}_1\,.\,\lambda\mathtt{Z}_2\,.\,(\mathtt{Z}_1 \approx \mathtt{a}) \wedge (\mathtt{Z}_2 \approx \mathtt{b}))\ \mathtt{X}\ \mathtt{Y}})$
   using rules 20.7, 22.5 and then 21.1
3. $\leftarrow \underline{\sim((\mathtt{X} \approx \mathtt{a}) \wedge (\mathtt{Y} \approx \mathtt{b}))}$
   using rules 20.7, 22.5 and then 21.2
4. $\leftarrow \sim(\mathtt{X} \approx \mathtt{a}) \vee (\mathtt{X} \approx \mathtt{a}) \wedge \sim(\mathtt{Y} \approx \mathtt{a})$
   using rules 20.7 and then 22.6(d)
5. $\leftarrow \sim(\mathtt{X} \approx \mathtt{a})$
   using rule 20.2

In step 4 the procedure generates two branches. The first one terminates immediately with a primitive inequality. The computed answer is $\sim(\mathtt{X} \approx \mathtt{a})$ and $\{\mathtt{R}/\lambda\mathtt{Z}.(\mathtt{Z} \approx \mathtt{X}) \bigvee \mathtt{R}'\}$. The second branch will continue as follows:

5. $\leftarrow \underline{(\mathtt{X} \approx \mathtt{a})} \wedge \sim(\mathtt{Y} \approx \mathtt{a})$
   using rule 20.2
6. $\leftarrow \sim(\mathtt{Y} \approx \mathtt{a})$
   using rule 20.5 and $\theta_6 = \{\mathtt{X}/\mathtt{a}\}$

The procedure will now terminate successfully yielding the substitution $\{\mathtt{R}/\lambda\mathtt{Z}.(\mathtt{Z} \approx \mathtt{a}) \bigvee \mathtt{R}', \mathtt{X}/\mathtt{a}\}$ and the primitive goal $\sim(\mathtt{Y} \approx \mathtt{a})$.

Implementation

In order to verify and experiment with the ideas presented in this paper, an implementation of the proposed proof procedure has been undertaken. For our implementation we extended an existing interpreter for the language $\mathcal{H}$ introduced in (Charalambidis et al. 2010; 2013). The modified interpreter supports the language $\mathcal{H}_{cn}$ and implements exactly the proof procedure presented in this section. The current version of the interpreter can be retrieved from `https://www.github.com/acharal/hopes`.

The source language[5] of the interpreter uses an extended Prolog-like syntax, similar to the one we used for motivation purposes in Section 2. Given a source program, the interpreter first derives the types of the predicates using a specialized but relatively straightforward type-inference algorithm. The well-typed program is then transformed to $\mathcal{H}_{cn}$-like syntax. The interpreter works interactively, ie., in the same way as ordinary Prolog interpreters. Every time the user specifies a goal, the type information that has been derived from the program is used to typecheck the goal. Subsequently, the goal together with the program is passed to a prover that implements the rules of the proposed proof procedure. The interpreter has been coded in Haskell and it implements the proof procedure as-is, namely it does not apply any low level compilation or any specialized optimizations.

5 The Soundness Theorem

In this section we state the soundness theorem for the proposed proof procedure.

To specify the soundness theorem for the proof procedure, one has to first define the notion of *correct answer*:

Definition 27. Let P be a program and G be a goal. An answer (θ, Q) for $\mathsf{P} \cup \{\mathsf{G}\}$ is a basic substitution θ and a primitive goal Q.

Definition 28. Let P be a program and let $\mathsf{G} = \leftarrow \mathsf{A}$ be a goal. An answer $(\theta, \leftarrow \mathsf{A}')$ is a *correct answer* for $comp(\mathsf{P}) \cup \{\mathsf{G}\}$ if for every model M of $comp(\mathsf{P})$ and for every state s over the domain of M, $[\![\mathsf{A}\theta]\!]_s(M) \sqsupseteq_o [\![\mathsf{A}']\!]_s(M)$.

The soundness theorem (Theorem 1 below) essentially amounts to demonstrating that every computed answer is a correct answer. To establish this fact, we need a lemma for each one of Definitions 21, 20 and 22. In particular, Lemma 2 demonstrates the correctness of Definition 21, Lemma 3 shows the correctness of Definition 22, and Lemma 4 the correctness of Definition 20 regarding single-step derivations. Lemma 5 actually shows the correctness of many-step derivations. The most demanding proof is that of Lemma 3. The proof of Lemma 3 (as-well-as the proof of the auxiliary Lemma 6 that will be given below) utilize one important and common assumption that exists in all previous works on constructive negation: we assume that "the language contains an infinite number of constants and functions" (see (Chan 1989, page 478)).

Lemma 2. *Let P be a program, let E, E' be body expressions of type o and assume that E is reduced to E'. Then, for every model M of $comp(\mathsf{P})$ and for every state s over the domain of M, it holds $[\![\mathsf{E}]\!]_s(M) = [\![\mathsf{E}']\!]_s(M)$.*

Proof. Straightforward using the semantics of $\mathcal{H}_{cn}$. □

Lemma 3. *Let P be a program, let E, E' be body expressions of type o and assume that E is negatively-reduced to E'. Then, for every model M of $comp(\mathsf{P})$ and for every state s over the domain of M, it holds $[\![\mathsf{E}]\!]_s(M) = [\![\mathsf{E}']\!]_s(M)$.*

Lemma 4. *Let P be a program, let $\mathsf{G} = \leftarrow \mathsf{A}$ and $\mathsf{G}' = \leftarrow \mathsf{A}'$ be goals and let θ be a basic substitution such that $\mathsf{G} \xrightarrow{\theta} \mathsf{G}'$. Then, for every model M of $comp(\mathsf{P})$ and for every state s over the domain of M, it holds $[\![\mathsf{A}\theta]\!]_s(M) \sqsupseteq_o [\![\mathsf{A}']\!]_s(M)$.*

Proof. By case analysis and standard logical arguments. □

Lemma 5. *Let P be a program and $\mathsf{G} = \leftarrow \mathsf{A}$ be a goal. Let $\mathsf{G}_0 = \mathsf{G}, \mathsf{G}_1 = \leftarrow \mathsf{A}_1, \ldots, \mathsf{G}_n = \leftarrow \mathsf{A}_n$ be a derivation of length n using basic substitutions $\theta_1, \ldots, \theta_n$. Then, for every model M of $comp(\mathsf{P})$ and for every state s over the domain of M, $[\![\mathsf{A}\theta_1 \cdots \theta_n]\!]_s(M) \sqsupseteq [\![\mathsf{A}_n]\!]_s(M)$.*

Proof. Using Lemma 4, Lemma 1 and induction on n. □

Finally, we will need another auxiliary lemma, which will be used to obtain the Soundness theorem:

Lemma 6. *Let E be a body expression of type o and G a conjunction of primitive inequalities. Then, for all states s and interpretations I if $[\![\mathsf{E}]\!]_s(I) \sqsupseteq_o [\![\mathsf{G}]\!]_s(I)$ then $[\![\mathsf{E}]\!]_s(I) \sqsupseteq_o [\![\mathsf{G}']\!]_s(I)$ where G' is a conjunction of primitive inequalities that is obtained from G by restricting it to the free variables of E.*

Using the above, we obtain the soundness theorem:

Theorem 1 (Soundness Theorem). *Let P be a program and G be a goal. Then, every computed answer for $\mathsf{P} \cup \{\mathsf{G}\}$ is a correct answer for $comp(\mathsf{P}) \cup \{\mathsf{G}\}$.*

Proof. A direct consequence of Lemma 5 using the auxiliary Lemma 6. □

6 Future Work

We have introduced the higher-order language $\mathcal{H}_{cn}$ which supports constructive negation. There have been various proposals for higher-order logic programming languages, the most notable among them being Hilog (Chen, Kifer, and Warren 1989; 1993) and λ-Prolog (Miller and Nadathur 1986; Nadathur and Miller 1990); to our knowledge however, none of these languages has ever before been extended to support constructive negation.

[5] The language is called *HOPES* standing for Higher-Order Prolog with Extensional Semantics.

One possible direction for future work is the investigation of completeness results for the proposed proof-procedure. This is an interesting venue for research but appears to be quite demanding due to the non-trivial nature of the proof procedure (note that, to our knowledge, no completeness results have been reported for the work of (Chan 1989)). Possibly a completeness result could be obtained for the case of finite-tree queries. An easier goal for future research would be to extend the language with other useful programming features that would enhance its higher-order style of programming (such as for example, type polymorphism).

References

Barendregt, H. P. 1984. *The Lambda Calculus – Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland.

Chan, D. 1988. Constructive negation based on the completed database. In *ICLP/SLP*, 111–125.

Chan, D. 1989. An extension of constructive negation and its application in coroutining. In *NACLP*, 477–493.

Charalambidis, A.; Handjopoulos, K.; Rondogiannis, P.; and Wadge, W. W. 2010. Extensional higher-order logic programming. In Janhunen, T., and Niemelä, I., eds., *JELIA*, volume 6341 of *Lecture Notes in Computer Science*, 91–103. Springer.

Charalambidis, A.; Handjopoulos, K.; Rondogiannis, P.; and Wadge, W. W. 2013. Extensional higher-order logic programming. *ACM Transactions on Computational Logic* 14(3).

Chen, W.; Kifer, M.; and Warren, D. S. 1989. Hilog as a platform for database languages. *IEEE Data Eng. Bull.* 12(3):37–44.

Chen, W.; Kifer, M.; and Warren, D. S. 1993. Hilog: A foundation for higher-order logic programming. *J. Log. Program.* 15(3):187–230.

Clark, K. L. 1977. Negation as failure. In *Logic and Data Bases*, 293–322.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R. A., and Bowen, K. A., eds., *ICLP/SLP*, 1070–1080. MIT Press.

Lifschitz, V. 2008. What is answer set programming? In Fox, D., and Gomes, C. P., eds., *AAAI*, 1594–1597. AAAI Press.

Lloyd, J. W. 1987. *Foundations of Logic Programming, 2nd Edition*. Springer.

Miller, D., and Nadathur, G. 1986. Higher-order logic programming. In Shapiro, E. Y., ed., *ICLP*, volume 225 of *Lecture Notes in Computer Science*, 448–462. Springer.

Nadathur, G., and Miller, D. 1990. Higher-order horn clauses. *J. ACM* 37(4):777–814.

The Well-Founded Semantics Is the Principle of Inductive Definition, Revisited

Marc Denecker
Department of Computer Science
K.U. Leuven
3001 Heverlee, Belgium
marc.denecker@cs.kuleuven.be

Joost Vennekens
Campus De Nayer | K.U. Leuven
Department of Computer Science
2860 Sint-Katelijne-Waver, Belgium
joost.vennekens@cs.kuleuven.be

Abstract

In the past, there have been several attempts to explain logic programming under the well-founded semantics as a logic of inductive definitions. A weakness in all is the absence of an obvious connection between how we understand various types of informal inductive definitions in mathematical text and the complex mathematics of the well-founded semantics. In this paper, we close this gap. We formalize the induction process in the most common principles and prove that the well-founded model construction generalizes them all.

Introduction

Ever since the well-founded semantics was first defined (Van Gelder, Ross, and Schlipf 1991), researchers have referred to the concept of *an inductive definition*, as we know it from mathematics, to explain the intuitions behind this formal semantics. This link was made explicit in several publications (Denecker 1998; Denecker, Bruynooghe, and Marek 2001; Denecker and Ternovska 2008), where we gave various arguments that, under the well-founded semantics, each rule in a set of rules can be seen as an (inductive or base) case of an inductive definition. A weakness in all is the absence of an obvious connection between how we understand various types of informal inductive definitions in mathematical text and the complex mathematics of the well-founded semantics. This paper aims to close this gap.

To open the discussion, let us consider two prototypical examples illustrating the two most common forms of inductive definitions: the monotone inductive Definition 1 of the transitive closure of a graph and the Definition 2 of the satisfaction relation of propositional logic by induction over the sub-formula order.

Definition 1. *The reachability graph R of a directed graph G is defined inductively:*

- $(x, y) \in R$ *if* $(x, y) \in G$*;*
- $(x, y) \in R$ *if there exists a vertex z such that* $(x, z), (z, y) \in R$*.*

Definition 2. *Given a propositional vocabulary Σ, the satisfaction relation $\models$ between Σ-structures and Σ-formulas of propositional logic is defined by induction over the structure of formulas:*

- $I \models P$ *if* P *is a propositional symbol and* $P \in I$*.*
- $I \models \alpha \wedge \beta$ *if* $I \models \alpha$ *and* $I \models \beta$*.*
- $I \models \alpha \vee \beta$ *if* $I \models \alpha$ *or* $I \models \beta$ *(or both).*
- $I \models \neg\alpha$ *if* $I \not\models \alpha$*.*

(Inductive) definitions serve to define formal objects, but they are not formal objects themselves. As such, we will refer to them as *informal definitions*. They are commonly phrased in natural language as collections of informal rules, possibly with an induction order. They define a set (or more than one, in case of simultaneous induction) *in terms of* other sets, which we will call *parameters*. E.g., Definition 1 has graph G as a parameter and Definition 2 the vocabulary Σ. Informal inductive definitions are broadly used in mathematics, broadly understood and, despite their informal nature, they are of mathematical precision. The set defined by it can be characterized in two quite different ways: "non-constructively", as the least set closed under rule application, and "constructively", as the set obtained by iterated rule application. By Tarski's cherished result on the least fixpoint of monotone operators, both principles coincide.

It comes as a surprise to many but the last statement is only half true. Def. 2 is non-monotone due to its 4th rule, and Tarski's result does not apply for it. There are infinitely many minimal sets that are closed under these rules and some are just weird: e.g., in some of them $\{P\} \models \neg P$ holds (See Example 1). What this shows is that the constructive principle is the more fundamental of the two. An inductive definition defines a set by describing how to construct it through an *induction process*. This process starts from the empty set, proceeds by applying rules till the set is closed under the rules. In case of induction over a well-founded order, rules must be applied "along" the specified order. This is the intuition that we guess all of us share and that will be formalized below.

The precision of informal inductive definitions makes them an ideal target for a formal empirical study. This is what we do in the next section, We define a simple formal, rule-based syntax with atomic heads and first order logic (FO) bodies. This leads to the following representation of the above informal definitions.

$$\left\{ \begin{array}{l} \forall x \forall y (R(x, y) \leftarrow G(x, y)) \\ \forall x \forall y (R(x, y) \leftarrow \exists z (R(x, z) \wedge R(z, y)) \end{array} \right\}$$

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

$$\left\{\begin{array}{l}\forall i \forall p(Sat(i,p) \leftarrow Atom(p) \wedge In(p,i)) \\ \forall i \forall f \forall g(Sat(i,And(f,g)) \leftarrow Sat(i,f) \wedge Sat(i,g)) \\ \forall i \forall f \forall g(Sat(i,Or(f,g)) \leftarrow Sat(i,f) \vee Sat(i,g)) \\ \forall i \forall f(Sat(i,Not(f)) \leftarrow \neg Sat(i,f))\end{array}\right\}$$

These two sets of rules, denoted respectively Δ_{TC} and $\Delta_{\models}$, will serve as examples throughout the paper. We then formalize several semantic aspects of informal inductive definitions: the induction process, the induction order, two sorts of inductive definitions and their generalization, etc. This study reveals several unexpected but fundamental aspects of inductive definitions. After that, we are on mathematical ground and we will prove that the set defined by a definition is its well-founded model.

The main contributions of the paper are then as follows. First, we provide a number of *definitions from first principles* that formalize intuitions and implicit conventions regarding informal inductive definitions. We do this for monotone inductive definitions, definitions by induction over a well-founded order as well as for their generalization, iterated inductive definitions. Second, we prove the equivalence of formal rule sets under the formalized semantics and the well-founded semantics.

Due to space restrictions, proofs are omitted.

Formal definitions and natural inductions

A vocabulary Σ consists of a set of non-logical predicate, function, constant and variable symbols. As usual, terms are built from the function, constant and variable symbols of Σ, and formulas are built from atomic formulas (a predicate symbol applied to a tuple of terms), the logical symbols $\mathbf{t}$ (true), $\mathbf{f}$ (false), $=$ (identity), connectives $\wedge, \vee, \neg$ and quantifiers $\exists, \forall$.

An occurrence of a variable x in a formula φ is free if it does not occur in a sub-formula $\exists x\psi$ or $\forall x\psi$ of φ. The set $free(\varphi)$ is the set of all variables with a free occurrence in φ. A sentence is a formula φ with $free(\varphi) = \emptyset$. An occurrence of a sub-formula φ in ψ is called *positive* in ψ if it occurs in the scope of an even number of negations, otherwise it is called *negative*. A formula φ is positive with respect to a set σ of predicate symbols if there are no atoms $P(\bar{t})$ with $P \in \sigma$ that have a negative occurrence in φ.

Σ-structures $\mathfrak{A}$ consist of a domain $D^{\mathfrak{A}}$ and an assignment of appropriate values $\tau^{\mathfrak{A}}$ to symbols $\tau \in \Sigma$. The value $P^{\mathfrak{A}}$ of a predicate symbol P/n is a function from $(D^{\mathfrak{A}})^n$ to $\{\mathbf{t}, \mathbf{f}\}$ (hence the characteristic function of a set of n-tuples). Let D be a domain. A *domain atom* of D is a pair $(P, \bar{a})$ with P/n a predicate symbol and $\bar{a} \in D^n$. Abusing notation, we write domain atoms as atoms $P(\bar{a}), Q(\bar{b})$. We use A, B, C as mathematical variables for domain atoms. We define $P(\bar{a})^{\mathfrak{A}} = P^{\mathfrak{A}}(\bar{a})$. For a given set σ of predicate symbols, we denote the set of domain atoms with predicates in σ by At_D^σ. There is a one-to-one correspondence between subsets of At_D^σ and σ-structures with domain D. We often exploit this correspondence in our notations by treating σ-structures as such sets. E.g., $\emptyset$ stands for the structure $\mathfrak{A}$ with $P^{\mathfrak{A}} = \emptyset$, for every $P \in \sigma$. We write $P(\bar{a}) \in \mathfrak{A}$ to denote that $P^{\mathfrak{A}}(\bar{a}) = \mathbf{t}$, and $\mathfrak{A} \subseteq \mathfrak{B}$ to denote that $P(\bar{a})^{\mathfrak{A}} \leq P(\bar{a})^{\mathfrak{B}}$, for every $P(\bar{a}) \in At_D^\sigma$. This order based on $\mathbf{f} < \mathbf{t}$ is sometimes called the *truth order*.

Definition 3. *A (formal)* definition *over Σ is a set of definition rules of the form*

$$\forall \bar{x}\ (P(\bar{t}) \leftarrow \phi)$$

where ϕ is a FO formula and $P(\bar{t})$ is an atomic formula over Σ such that P is not $=$.

We call $P(\bar{t})$ the head of the rule, and ϕ the body. The connective $\leftarrow$ is called *definitional implication*. Rules of this kind are similar in nature to and generalize *productions* in (Martin-Löf 1971). A predicate appearing in the head of a rule of a definition Δ is called a *defined predicate* of Δ; all other non-logical symbols in Δ are called *parameters* of Δ. The sets of defined predicates and parameters of Δ are denoted by $def(\Delta)$ and $pars(\Delta)$, respectively. We assume without loss of generality that every rule is of the form $\forall \bar{x}\ (P(\bar{x}) \leftarrow \phi)$ where $\bar{x}$ is a tuple of distinct variables. Indeed, every rule $\forall \bar{x}\ (P(\bar{t}) \leftarrow \phi)$ can be put into that form as $\forall \bar{y}\ (P(\bar{y}) \leftarrow \exists \bar{x}(\bar{y} = \bar{t} \wedge \phi))$. Below, a domain atom $P(\bar{a})$ of a defined predicate of Δ is called a *defined domain atom*. Note that this formal notion of definition does not (yet) include an induction order.

A definition is always evaluated in a *context* providing values for its parameters. We formalize this.

Definition 4. *We call a $pars(\Delta)$-structure $\mathcal{O}$ a* context *of Δ.*

Example 1. *The definition $\Delta_{\models}$ of Sat formalizes the informal Definition 2 of satisfaction. Its parameter symbols are $Atom, In, And, Or$ and Not. We view $\Delta_{\models}$ as a many-sorted definition, with two sorts for structures and formulas. Any set S of propositional symbols induces a context $\mathcal{O}_S$ which is the $pars(\Delta)$-structure defined as follows:*

- *$D^{\mathcal{O}_S} = PropF(S) \cup Struct(S)$, where $PropF(S)$ is the set of propositional formulas over S and $Struct(S)$ is the set of propositional S-structures.*
- *$And^{\mathcal{O}_S}$ is the function that maps pairs of formulas $(\psi, \phi) \in PropF^2$ to $\psi \wedge \phi$. The functions $Or^{\mathcal{O}_S}$ and $Not^{\mathcal{O}_S}$ are defined in a similar vein.*
- *Finally, $In^{\mathcal{O}_S}$ is $\{(I,p) | I \in Struct(S), p \in I\}$.*

As an example, defined domain atoms for $S = \{P\}$ are of the form $Sat(\{P\}, P), Sat(\{\}, P \wedge \neg P), \ldots$. The value $t^{\mathcal{O}_s}$ of the term $t = And(P, Not(P))$ is the formula $P \wedge \neg P$.

From here till the end of this section, we assume the presence of a definition Δ, and a context $\mathcal{O}$ for Δ with domain D. The induction process associated with Δ in context $\mathcal{O}$ will be a sequence of $def(\Delta)$-structures $\mathfrak{A}$ with domain D. By $\mathcal{O} \circ \mathfrak{A}$, we denote the structure $\mathfrak{B}$ with domain D and such that $P^{\mathfrak{B}} = P^{\mathfrak{A}}$ if $P \in def(\Delta)$ and $\tau^{\mathfrak{B}} = \tau^{\mathcal{O}}$ if $\tau \in pars(\Delta)$. In the sequel, we frequently evaluate formulas with respect to structures $\mathcal{O} \circ \mathfrak{A}$. Because $\mathcal{O}$ is given and fixed, we take the liberty to write only the "variable" part and write e.g., $\mathfrak{A} \models \varphi$ instead of $\mathcal{O} \circ \mathfrak{A} \models \varphi$, or $A^{\mathfrak{A}}$ instead of $A^{\mathcal{O} \circ \mathfrak{A}}$, etc.

The next definitions are formalizations of the concept of an element being *derivable* from a definition, and a set being *closed* or *saturated* under a definition.

Definition 5. *Given a context $\mathcal{O}$, we say that $P(\bar{a})$ is derivable from rule $\forall\bar{x}(P(\bar{x}) \leftarrow \varphi)$ in $(\mathcal{O}\circ)\mathfrak{A}$ if $\varphi^{\mathfrak{A}[\bar{x}:\bar{a}]} = \mathbf{t}$. We say that $P(\bar{a})$ is Δ-derivable in $\mathfrak{A}$ if it is derivable from a rule of Δ in $\mathfrak{A}$. Below, we denote this by $\mathfrak{A} \vdash_\Delta P(\bar{a})$.*

Definition 6. *Given a context $\mathcal{O}$, we say that $\mathfrak{A}$ is closed (or saturated) on a set S of defined domain atoms under Δ in $\mathcal{O}$ if for every $A \in S$, $\mathcal{O} \circ \mathfrak{A} \vdash_\Delta A$ implies $A \in \mathfrak{A}$. In general, $\mathfrak{A}$ is* closed *(or saturated) under Δ (in $\mathcal{O}$) if it is closed on $At_D^{def(\Delta)}$ under Δ in $\mathcal{O}$.*

Example 2. *Let Δ_{ev} be the following non-monotone definition:*

$$\left\{ \begin{array}{l} Even(0) \leftarrow \\ \forall x(Even(x+1) \leftarrow \neg Even(x)) \end{array} \right\} \quad (1)$$

It can be viewed as the formalization of an informal definition of the even numbers by induction over the standard order of numbers: 0 is even; if n is not even then n+1 is even. This definition is structurally equivalent to the first and last rule of Definition 2. For succinctness, we abbreviate $Even$ often to Ev.

The context $\mathcal{O}$ is the structure of the natural numbers, with the standard interpretation of $0, 1$ and $+$. Consider the following sets, for every $N \in \mathbb{N}$:

$$\{Ev(0), Ev(2), Ev(4), \dots\}$$
$$\{Ev(0), Ev(2), \dots, Ev(2N), Ev(2N+1), Ev(2N+3), \dots\}$$

Each of these sets is closed under Δ_{ev}. None has a subset that is closed. Thus, the defined set is not the least set closed under the rules. A similar phenomenon arises for Definition 2.

The main concept of this paper is that of the induction process. It is formalized as follows.

Definition 7. *A* natural induction *$\mathcal{N}$ of Δ in $\mathcal{O}$ (with domain D) is an increasing sequence $(\mathfrak{A}_\alpha)_{0\le\alpha\le\beta}$ of $def(\Delta)$-structures with domain D such that:*

- *$\mathfrak{A}_0$ is the empty structure $\emptyset$.*
- *For each successor ordinal $i+1 \le \beta$, for each domain atom $A \in \mathfrak{A}_{i+1} \setminus \mathfrak{A}_i$, A is derivable from Δ in $\mathfrak{A}_i$ ($\mathfrak{A}_i \vdash_\Delta A$). We say that A is derived at i and define $\|A\|_\mathcal{N} := i$, the rank of A in $\mathcal{N}$.*
- *For each limit ordinal $\lambda \le \beta$, $\mathfrak{A}_\lambda = \bigcup_{\alpha<\lambda} \mathfrak{A}_\alpha$.*

A natural induction is called terminal *if $\mathfrak{A}_\beta$ is closed under Δ (in $\mathcal{O}$).*

Natural inductions will be denoted compactly as a sequence of the (disjoint) sets of atoms that are derived at each step. E.g.,

$$\to \{A_1, \dots, A_n\} \to \{B_1, \dots, B_m\} \to \dots$$

derives the A_i's in step 0 and the B_j's in step 1. If such a set is a singleton we drop the brackets.

Example 3. *Consider the formal definition Δ_{TC} formalizing the transitive closure Definition 1. Take context $\mathcal{O}$ such that $D = \{a, b, c\}$, $G^\mathcal{O} = \{(a,a), (b,c), (c,b)\}$. All terminal natural inductions converge to $\{(a,a), (b,c), (c,b), (b,b), (c,c)\}$. E.g.,*

$$\to T(a,a) \to T(b,c) \to T(c,b) \to T(b,b) \to T(c,c)$$
$$\to \{T(c,b), T(b,c)\} \to T(c,c) \to T(b,b) \to T(a,a)$$

The example illustrates some basic points about informal inductive definitions. They indeed define the concept in terms of the parameters, by describing how to construct it through iterated rule application. The description of the construction process is however highly *non-deterministic*: rules can be applied in many orders. From a practical point of view, it is all-important that such induction sequences converge to the same fixpoint, otherwise the definition would be ambiguous. For a monotonic informal definition, such as Definition 1, the order of rule application is not important, because—given the graph G—all sequences converge to the intended set, which is the least relation that is closed under the rules. In the above framework of natural induction sequences, this can be proven formally. First we propose a formalization of the notion of monotone inductive definition.

Definition 8. *We call Δ monotone in $\mathcal{O}$ if for all pairs of $def(\Delta)$-structures $\mathfrak{A} \subseteq \mathfrak{B}$, for all defined domain atoms A, if $\mathfrak{A} \vdash_\Delta A$ then $\mathfrak{B} \vdash_\Delta A$.*

Proposition 1. *Each terminal natural induction of a monotone Δ in $\mathcal{O}$ converges to the least $def(\Delta)$-structure $\mathfrak{A}$ that is closed under Δ in $\mathcal{O}$.*

This proposition is not difficult to prove but follows from the general Theorem 3 below.

The convergence property does not hold for non-monotone definitions. The problem is that the body of a non-monotone rule may eventually become false, after it has already been true. Natural inductions that apply a rule during the "window" where its body holds will derive its head, whereas natural inductions that miss this window may not.

Example 4 (Continuation of Example 1). *Consider definition $\Delta_\models$ in the context of the structure $\mathcal{O}_S$ for the singleton vocabulary $S = \{P\}$. There are only two structures for the vocabulary S, namely, $\emptyset$ and $\{P\}$. Below is an initial segment of a stepwise natural induction that derives a wrong fact.*

$$\to Sat(\{P\}, \neg P) \to Sat(\{P\}, P) \to \dots$$

In the first step, with $\mathfrak{A}_0 = \emptyset$, all instances of the rule for negation are applicable. Here, we use it to derive $Sat(\{P\}, \neg P)$. However, the next step applies the base rule to derive $Sat(\{P\}, P)$, thus falsifying the condition of the rule that was applied in the first step.

We realize that the role of the induction order in informal definitions is to delay the application of rules until it is safe to do so, that is, until later rule applications cannot longer falsify the premise of a rule that has been applied before. We now formally define the notion of definition by induction over a well-founded order. For brevity, we call it an *ordered definition*.

Definition 9. *Given is a domain D. An ordered definition is a pair $(\Delta, \prec)$ with Δ a definition and $\prec$ a strict well-founded order on $At_D^{def(\Delta)}$.*

Recall that a strict order is irreflexive, transitive and asymmetric. A strict order $\prec$ is well-founded if it has no infinite descending chains $x_0 \succ x_1 \succ x_2 \succ \dots$.

Example 5. *The induction order of informal Definition 2 is the sub-formula order. In the context of definition $\Delta_{\models}$ and a structure $\mathcal{O}_S$, it corresponds to a strict well-founded order $\prec$ on domain atoms, where $Sat(I, \psi) \prec Sat(J, \phi)$ if $I = J$ and ψ is a strict sub-formula of ϕ.*

The induction order provided with an informal definition serves to constrain the order of rule application in natural inductions. How does this work? Intuition says that no rule should be applied to derive a fact as long as there are derivable but not yet derived facts that are strictly smaller. E.g., assume that at some point in the induction process $I \models \varphi$ is derivable. We are allowed to do so only if there is no strict subformula ϕ of φ for which $I \models \varphi$ is derivable but was not yet derived. In general, we can derive $P(\bar{a})$ if the current set $\mathfrak{A}_i$ is saturated on atoms preceding $P(\bar{a})$ in the induction order. This is formalized in the following definition.

Recall that the rank $\|A\|_{\mathcal{N}}$ of A in a natural induction $\mathcal{N}$ is the ordinal i such that $A \in \mathfrak{A}_{i+1} \setminus \mathfrak{A}_i$.

Definition 10. *A natural induction $\mathcal{N}$* respects $\prec$ *(w.r.t. Δ and $\mathcal{O}$) if for any domain atom A with $\|A\|_{\mathcal{N}} = i$, $\mathfrak{A}_i$ is saturated on $\{B \mid B \prec A\}$ (under Δ in $\mathcal{O}$).*

We say that $\mathcal{N}$ follows $\prec$ *if for every A and B derived by $\mathcal{N}$, $A \prec B$ implies $\|A\|_{\mathcal{N}} < \|B\|_{\mathcal{N}}$.*

Example 6. *The natural induction of Example 4:*

$$\rightarrow Sat(\{P\}, \neg P) \rightarrow Sat(\{P\}, P) \rightarrow \dots$$

does not respect the subformula order. The atom $Sat(\{P\}, \neg P)$ is derived in the first step, when the empty set is not saturated in $\{A \mid A \prec Sat(\{P\}, \neg P)\}$ since $Sat(\{P\}, P)$ is derivable.

In general the induction process is highly underspecified, even if an induction order is given.

Example 7. *(Example 5 continued). Natural inductions of the informal Definition 2 will derive $I \models \varphi$ only after the satisfaction of all sub-formulas has been derived. This constrains the order of rule application, but much freedom is left. There are infinitely many such natural inductions. A few non-terminal ones are:*

$$\rightarrow Sat(\{P\}, P) \rightarrow Sat(\{P\}, P \wedge P) \rightarrow Sat(\{P\}, \neg\neg P)$$

$$\rightarrow Sat(\{P\}, P) \rightarrow Sat(\{P\}, \neg\neg P) \rightarrow Sat(\{P\}, P \vee P)$$

Note that both natural inductions respect the sub-formula order and follow it. Intuition suggests that these sequences can be extended to converging terminal natural inductions, and this will be proven below.

Given our experience with informal definitions, we expect some "good" properties of natural inductions that respect the induction order $\prec$: (1) that they all converge, (2) that such a natural induction *follows* the induction order, (3) that once an element is derived, it remains derivable, and (4) that in the limit, the defined set is the intended one. However, none of these properties hold right now.

The major question is related to (1). It is essential that all natural inductions that respect $\prec$ converge. However, it is straightforward to see that this is not the case. Take the empty induction order $\emptyset$ for the definition $\Delta_{\models}$ in context $\mathcal{O}_S$. This order is a strict well-founded order and all natural inductions respect it in a trivial way. As we saw in Example 4, not all of these natural inductions converge.

As for (2), a counterexample is below.

Example 8. *Consider the order $P \prec Q$ and definition:*

$$\left\{ \begin{array}{l} Q \leftarrow \mathbf{t} \\ P \leftarrow Q \end{array} \right\}$$

Here is a terminal natural induction:

$$\rightarrow Q \rightarrow P$$

It obviously does not follow $\prec$ since $P \prec Q$. However, it respects $\prec$. In the first step, when Q is derived, the structure $\mathfrak{A}_0 = \emptyset$ is saturated on $\{A \mid A \prec Q\} = \{P\}$, since P is not derivable. In the second step, $\mathfrak{A}_1 = \{Q\}$ is trivially saturated on $\{A \mid A \prec P\} = \{\}$.

A counterexample for (3) and (4) is given below.

Example 9. *We reconsider Δ_{ev} and $\mathcal{O}$ from Example 2.*

$$\left\{ \begin{array}{l} Even(0) \leftarrow \\ \forall x(Even(x+1) \leftarrow \neg Even(x)) \end{array} \right\} \qquad (2)$$

Let $\prec$ be the order induced by the standard order on the natural numbers. That is, $Ev(n) \prec Ev(m)$ if $n < m$. This is a total order, and the unique terminal natural induction that respects and follows it constructs the set of even numbers:

$$\rightarrow Ev(0) \rightarrow Ev(2) \rightarrow Ev(4) \rightarrow \dots \rightarrow Ev(2n) \rightarrow \dots$$

Now take the following non-standard induction order:

$$Ev(1) \prec Ev(0) \prec Ev(2) \prec Ev(3) \prec \dots$$

Also this is a total strict well-founded order. The unique terminal natural induction that respects $\prec$ is:

$$\rightarrow Ev(1) \rightarrow Ev(0) \rightarrow Ev(3) \rightarrow Ev(5) \rightarrow \dots$$

Note that $Ev(1)$ is not longer derivable after step 2. Also, it clearly does not define the intended set.

In non-monotone informal definitions, we impose a well-founded induction order to obtain convergence of the induction process. However, it is clear from the above examples that in selecting the induction order, great care is required. In general, imposing an unsuitable induction order w.r.t. Δ and $\mathcal{O}$ may have a number of undesired effects as just shown.

Something clearly wrong with the second induction order in the above example is that it does not *match* the structure of the given rules. In particular, $Ev(1)$ is defined in terms of $Ev(0)$, even though $Ev(0)$ is strictly larger than $Ev(1)$ in the proposed induction order. This would be unacceptable in an informal definition.

The above examples expose one of the implicit conventions of informal inductive definitions. This is that the induction order should "match" the structure of the rules of a definition. Intuitively, this means that defined facts may only "depend" on facts that are strictly smaller in the induction order. We now formalize this.

First we formalize the notion of dependency relation. Below, for binary relation $\propto$, $\mathfrak{A}|_{\propto A}$ denotes $\mathfrak{A} \cap \{B \mid B \propto A\}$.

Definition 11. *A binary relation $\propto$ on $At_D^{def(\Delta)}$ is a* dependency relation *of Δ in $\mathcal{O}$ if $\propto$ is transitive and for all A and all $\mathfrak{A}, \mathfrak{B}$, if $\mathfrak{A}|_{\propto A} = \mathfrak{B}|_{\propto A}$ then $\mathfrak{A} \vdash_\Delta A$ iff $\mathfrak{B} \vdash_\Delta A$.*

If $\propto$ is a dependency relation, then for any defined atom A, the set $\{B \mid B \propto A\}$ is (a superset of) the set of atoms on which A depends. Indeed, in any pair of structures that coincide on this set, A is derivable in both or in none.

That an induction order "matches" the rules of a definition simply means that $\prec$ is a dependency relation.

Definition 12. *We say that $\prec$* strictly orders *Δ in $\mathcal{O}$ if $\prec$ is a strict well-founded order and a dependency relation of Δ in $\mathcal{O}$.*

Example 10. *In case of definition Δ_{ev} of even numbers and the structure $\mathcal{O}$ of Example 2, we see that the first order*

$$Ev(0) \prec Ev(1) \prec Ev(2) \prec Ev(3) \prec \dots$$

strictly orders Δ_{ev}, while the second order

$$Ev(1) \prec Ev(0) \prec Ev(2) \prec Ev(3) \prec \dots$$

does not. E.g., $\emptyset$ and $\{Ev(0)\}$ are identical on $\{B \mid B \prec Ev(1)\} = \emptyset$, but $\emptyset \vdash_\Delta Ev(1)$ while $\{Ev(0)\} \nvdash_\Delta Ev(1)$.

Natural inductions that respect an order $\prec$ that strictly orders Δ in $\mathcal{O}$ satisfy all the good properties (1-4) above, as shown by the following two propositions.

Proposition 2. *If $\prec$ strictly orders Δ then any natural induction $\mathcal{N}$ that respects $\prec$ also follows $\prec$.*

Proposition 3. *If $(\Delta, \prec)$ is an ordered definition in context $\mathcal{O}$ and $\prec$ strictly orders Δ in $\mathcal{O}$, then all terminal natural inductions that respect $\prec$ converge. Moreover the limit is independent of $\prec$. (It is the ultimate well-founded fixpoint of Δ in $\mathcal{O}$ - see Definition 19).*

Also this proposition follows from the stronger Theorem 3.

This theorem shows that an ordered definition in which $\prec$ strictly orders Δ unambiguously defines a set. The definition of an ordered definition can now be refined as follows.

Definition 13. *A definition Δ is a definition by well-founded induction over $\prec$ in $\mathcal{O}$ (or briefly, an ordered definition) if $\prec$ strictly orders Δ in $\mathcal{O}$. The structure defined by it is the limit of any terminal natural induction that respects $\prec$.*

Interestingly, the convergence property states that the limit is independent of the selected order. Sometimes this phenomenon can be seen in mathematical text.

Example 11. *We defined the satisfaction relation $\models$ over the subformula order but it is not uncommon to define it over alternative induction orders. For example, we could define $\models$ by induction on the* size *of formulas. Formally, we define $Sat(I, \psi) \prec Sat(J, \phi))$ if $I = J$ and the size of ψ (the number of nodes in its parse tree) is strictly less than the size of ϕ. Alternatively, we may define $\models$ by induction on the* depth *of formulas, i.e., the length of the longest branch in the parse tree of ϕ. The three orders lead to three variants of Definition 2. Intuition suggests that they are equivalent.*

It is indeed easy to verify that each of them strictly orders $\Delta_\models$ in $\mathcal{O}_S$: for each rule instance, the body refers to formulas that are strict subformula's, and have smaller size and depth than the formula in the head. Hence, it follows from the proposition that these definitions are indeed equivalent.

This does not mean that they have the same natural inductions. E.g., reconsider the natural induction of Example 7:

$$\rightarrow Sat(\{P\}, P) \rightarrow Sat(\{P\}, \neg\neg P) \rightarrow Sat(\{P\}, P \vee P)$$

This one respects and follows the subformula order and the size order. However, it does not respect the depth order, since $\mathfrak{A}_1$ is not saturated on $\{B \mid B \prec Sat(\{I\}, \neg\neg P)\}$. For instance, $Sat(\{P\}, P \vee P)$ is derivable but not derived and $P \vee P$ has strictly smaller depth than $\neg\neg P$.

Generalizing monotone and ordered definitions. There is an obvious similarity between Propositions 1 and 3. However, the former is not a generalization of the latter, because not all monotone definitions are ordered. For instance, there is no $\prec$ that strictly orders definition Δ_{TC} of transitive closure. Due to the transitivity rule, all defined domain atoms depend on each other; the only dependency relation is the total one and this is not a strict order. In this section, we define the more general class of *iterated inductive definitions*, which encompasses all ordered definitions as well as all monotone definitions. We will then prove a theorem for this more general class, which generalizes both of the earlier results.

The general idea of iterated inductive definitions is that they admit a dependency $\propto$ that is not a strict order. However, if atom A, B depend on each other (that is, $A \propto B \propto A$), then they depend *monotonically* on each other: deriving B may make A derivable and vice versa.

We define $A \prec_\propto B$ if $A \propto B$ and $B \not\propto A$. Since $\propto$ is transitive, $\prec_\propto$ is a strict order. $\prec_\propto$ divides the set of domain atoms into a set of strictly ordered "layers" such that, for all A, B, if $A \prec_\propto B$, then A is in a strictly lower layer than B, and if $A \propto B \propto A$, they are in the same layer.

Natural inductions of an iterated inductive definition proceed along the order $\prec_\propto$. Just as for an ordered definition, an atom A may be derived at step i only if $\mathfrak{A}_i$ is saturated on $\{B \mid B \prec_\propto A\}$. In this way, a natural induction closes layer by layer, and starts a new monotone induction in the next layer as soon as a layer is saturated. The following definition will serve to ensure that the "sub-inductions" that take place inside a single layer are monotone.

Definition 14. *A relation $\propto$ monotonically orders Δ in $\mathcal{O}$ if $\prec_\propto$ is a strict well-founded order and for all defined A, for all $\mathfrak{A}, \mathfrak{B}$ such that $\mathfrak{A}|_{\prec_\propto A} = \mathfrak{B}|_{\prec_\propto A}$ and $\mathfrak{A}|_{\propto A} \subseteq \mathfrak{B}|_{\propto A}$, if $\mathfrak{A} \vdash_\Delta A$ then $\mathfrak{B} \vdash_\Delta A$.*

Proposition 4. *If $\propto$ monotonically orders Δ in $\mathcal{O}$ then $\propto$ is a dependency relation of Δ in $\mathcal{O}$.*

Definition 15. *We say that a natural induction $\mathcal{N}$ respects (follows) a dependency relation $\propto$ if it respects (follows) $\prec_\propto$ according to Definition 10.*

Proposition 5. *If a natural induction $\mathcal{N}$ respects a relation $\propto$ that monotonically orders Δ in $\mathcal{O}$, then $\mathcal{N}$ follows $\propto$.*

Proposition 6. *Assume that $\propto$ monotonically orders Δ in $\mathcal{O}$. Then all terminal natural inductions that respect $\propto$ converge. Moreover, the limit is independent of $\propto$. (It is the ultimate well-founded fixpoint of Δ in $\mathcal{O}$.)*

Again, this proposition follows from Theorem 3.

We have already defined the concept of a monotone and ordered definition in context $\mathcal{O}$. Now, we also define the concept of an iterated inductive definition (in $\mathcal{O}$).

Definition 16. *A definition Δ is a definition by iterated induction over $\propto$ in $\mathcal{O}$ if $\propto$ monotonically orders Δ in $\mathcal{O}$. Its defined structure is the limit of any terminal natural induction.*

Now we can show that iterated inductive definitions generalize monotone and ordered definition. For a monotone definition, the entire set of all domain atoms can serve as a single layer. Let $\propto_t$ denote the total binary relation on $At_\Delta^{def(\Delta)}$. Note that $<_{\propto_t} = \emptyset$.

Proposition 7. *A definition Δ is monotone in $\mathcal{O}$ iff Δ is a definition by iterated induction over $\propto_t$ in $\mathcal{O}$. A natural induction of Δ in $\mathcal{O}$ (trivially) respects $\propto_t$.*

Proposition 8. *For a binary relation $\propto$, a definition Δ is a definition by well-founded induction over $\propto$ in $\mathcal{O}$ iff Δ is by iterated induction over $\propto$ in $\mathcal{O}$ and $\propto$ is irreflexive and asymmetric (hence, a strict order).*

As a consequence, Prop. 6 is a generalization of both Prop. 1 and Prop. 3.

Informal iterated inductive definitions are quite common in mathematical text although they are only very rarely formulated as sets of informal rules. To phrase them, formal scientists use other tools from their toolbox, for example fixpoints of operators. A well-known iterated inductive definition is the alternating fixpoint definition of the well-founded model in (Van Gelder 1993). In this theory, a (stable) operator $\mathcal{A}$ is defined on structures by defining $\mathcal{A}(\mathfrak{A})$ as the least fixpoint of a monotone operator $\lambda x T(x, \mathfrak{A})$. This stable operator is anti-monotone. The well-founded fixpoint is then characterised as the limit of an alternating fixpoint construction using $\mathcal{A}$. This is iterated induction in the sense that each of the steps involves itself a monotone inductive construction.

A rare case where iterated induction is explicitly available in rule form is in the definition of *stable theory* (Marek 1989) which is the set of modal propositional formulas closed under the standard inference rules and two additional ones:

$$\frac{\vdash \psi}{\vdash K\psi} \qquad \frac{\nvdash \psi}{\vdash \neg K\psi}$$

The second is a non-monotone rule. The set is computed by iterated induction for increasing modal nesting depth of modal formulas.

In knowledge representation, there are many applications of iterated inductive definitions that have a natural representation as rule sets. This is for instance the case in representations of dynamic systems with ramifications (Denecker and Ternovska 2007).

Summary of informal definitions. The above theory exposes several important issues of informal definitions. First, that the "non-constructive" characterization is incorrect in case of non-monotone (ordered) definitions. Second, that the induction process is highly non-deterministic, and therefore that convergence is all-important. In mathematical practice, we typically take this property for granted. In fact, it is not trivial at all. It is a fundamentally important property of inductive definitions.

Third, in mathematical texts, we have a certain degree of freedom when it comes to choosing the induction order for an inductive definition. Nevertheless, the order is far from arbitrary and needs to match the structure of the rules. Our exposition clarifies the role and nature of the induction order, and how it constrains the order of rule application.

Finally, as shown by Proposition 6, the choice of the induction order is irrelevant as long as it matches the rules. The order does not affect the semantics of the definition.

In view of this, one may wonder why an induction order is specified at all in mathematical text? One possible explanation is that it serves to help the reader better understand the definition. Moreover, the specified order may help him/her as a kind of *parity check* of the soundness of the definition. Indeed, not all sets of informal rules form sensible definitions (far from). The induction order helps in verifying that the (informal) rules indeed form a sensible definition (see also the "parity check" discussion following Corollary 1).

Developing a logic of definitions. In the mathematical logic of ordered and iterated induction (IID) presented in (Buchholz et al. 1981), an iterated inductive definition is expressed via SO formula's that express a definition Δ and, independently, an induction order $\prec$. They use this logic system to study proof-theoretic strength and expressivity of iterated definitions. From a representational point of view however, we see two problems with an approach in which the induction order is explicitly expressed. First, expressing an induction order in logic might be as complex as expressing the definition itself, if not more. It is a needless complication of the knowledge representation process. Second, it also makes the knowledge representation process more error-prone. Even though the logic of (Buchholz et al. 1981) imposes strong additional constraints on the induction process so that convergence can be guaranteed, it is still possible to express combinations of a definition and an induction order that will converge to the wrong limit. For example, one can encode the definition Δ_{ev} with the non-matching order $Ev(1) \prec Ev(0) \prec Ev(2) \prec \ldots$, in which case the unintended set $\{Ev(1), Ev(0), Ev(3), Ev(5), \ldots\}$ will be constructed.

It is preferable to design a logic of definitions in which only the rules need to be represented and the order is left implicit. Indeed, Proposition 6 gives us license to do this, because it shows that all induction orders that fit the structure of the rules of Δ produce the same unique limit of their terminal natural inductions.

Designing a logic in which all and only iterated inductive definitions can be expressed is impossible. A logic's syntax should be decidable, while it is easy to see that it is undecidable whether a rule set is a monotone, ordered or iterated definition in a structure $\mathcal{O}$. One option out is by imposing syntactical constraints. For monotone definitions, we could impose the simple syntactic criterion that only rules with

positive bodies are allowed. Beyond monotone definitions, however, this seems unfeasible. One issue is that whether a rule sets admits a $\propto$ that strictly or monotonically orders Δ in $\mathcal{O}$ may depend on the context $\mathcal{O}$.

Example 12. *The definition Δ_{ev} is an ordered inductive definition over the standard order in the natural numbers, but does not admit an induction order in the context $\mathcal{O}$ with domain $\{0,1\}$, $0^{\mathcal{O}} = 0, 1^{\mathcal{O}} = 1, +^{\mathcal{O}} = \{(0,0,0),(0,1,1),(1,0,1),(1,1,1)\}$. Indeed, in this structure, the two instances of the inductive rule are: $even(1) \leftarrow \neg even(0)$ and $even(1) \leftarrow \neg even(1)$. The least dependency relation is $even(0) \propto even(1) \propto even(1)$, but $\propto$ does not strictly or monotonically order Δ_{ev} in $\mathcal{O}$. Similarly, definition $\Delta_{\models}$ is not a legal definition in contexts where $Not^{\mathcal{O}}$ contains cycles.*

Requiring syntactic stratification is a possibility but it would eliminate interesting cases such as definition $\Delta_{\models}$. A more refined condition like local stratification (Przymusinski 1988) offers more liberty but it applies only to conjunctive rule bodies, in Herbrand contexts, is also undecidable and moreover is very brittle. Simple equivalence preserving transformations may transform a locally stratified rule set into one that is not. For instance, transforming rules $\forall \bar{x}(P(\bar{t}) \leftarrow \varphi)$ to $\forall \bar{y}(P(\bar{y}) \leftarrow \exists \bar{x}(\bar{t} = \bar{y} \wedge \varphi))$ breaks local stratification. This is not a good idea.

The alternative option is to waive any syntactic restriction but to design a "partial" semantics that assigns the "right" values to the defined predicates if Δ is a sensible inductive definition in $\mathcal{O}$ and otherwise does not assign a value. It is here that the well-founded semantics comes in. It will help us to cope with the absence of an explicit induction order in two ways. First, if this semantics assigns a value, then we can be certain that the definition is correct and that the assigned value is indeed the intended one. Second, as we will see, this semantics also provides mathematical tools that we can use to understand the workings of an inductive definition, without making reference to a specific induction order.

Well-founded inductions

The goal of this section is to show that the well-founded semantics allows us to perform the natural induction along some induction order, without actually knowing this order up front. There exist many formalizations of the well-founded semantics, but they all have in common that they construct a sequence of three-valued structures of increasing precision. This is quite different from a natural induction, which is an increasing sequence of two-valued structures. The following observation gives a clue as to how such sequences might be related.

Consider a (stepwise) natural induction of Δ in context $\mathcal{O}$ that respects $\prec_{\propto}$.

$$\rightarrow A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_\beta$$

At each i, the structure $\mathfrak{A}_i = \{A_1, \ldots, A_i\}$ already provides the following partial information about the limit $\mathfrak{A}_\beta$:

- $A \in \mathfrak{A}_\beta$ if $A \in \mathfrak{A}_i$;
- $A \notin \mathfrak{A}_\beta$ if $\mathfrak{A}_i \nvdash_\Delta A$ and $\mathfrak{A}_i$ is saturated in $\{B | B \propto A\}$. Indeed, since $\mathfrak{A}_i$ is saturated on $\{B | B \propto A\}$, it is saturated on every $\{B \mid B \propto C\} \subseteq \{B \mid B \propto A\}$. As a consequence, no atoms $C \propto A$ will ever be derived in the future. Since A is not derivable now, it will never become derivable.
- It is unknown whether $A \in \mathfrak{A}_\beta$ otherwise.

This tells us how to construct from each $\mathfrak{A}_i$ a three-valued structure $\mathcal{I}_i$ that "approximates" the defined set $\mathfrak{A}_\beta$. As such, a natural induction implicitly specifies a sequence of three-valued structures of increasing precision, that converges to $\mathfrak{A}_\beta$. Let us explore this in a few of the examples.

Example 13. *The transitive closure definition Δ_{TC} in context $\mathcal{O}$ of Example 3 is an iterated inductive definition over the total dependency relation $\propto_t$ ($\prec_{\propto_t} = \emptyset$). This induces a unique layer consisting of all domain atoms. Consider the following terminal natural induction:*

$$\rightarrow R(a,a) \rightarrow R(b,c) \rightarrow R(c,b) \rightarrow R(b,b) \rightarrow R(c,c)$$

The intermediate structures $\mathfrak{A}_i$ ($i < 5$) in this sequence are not saturated and no negative information is available. Hence, for each $i < 5$, all atoms not in $\mathfrak{A}_i$ are unknown in $\mathcal{I}_i$. However, the limit $\mathfrak{A}_5 = R$ is saturated, and only then it can be derived that all atoms not in $\mathfrak{A}_5$ are false.

Example 14. *The definition Δ_{ev} in context $\mathcal{O}$ of Example 2 is by induction over the standard order. The unique natural induction is:*

$$\rightarrow Ev(0) \rightarrow Ev(2) \rightarrow \ldots \rightarrow Ev(2n) \rightarrow \ldots$$

At each step i, one can verify that $\mathfrak{A}_i$ is saturated on $\{Ev(j) \mid j \leq 2i-1\}$. It follows that, for $0 \leq j \leq i$, $\mathcal{I}_i \models \neg Ev(2j-1)$. The following sequence describes the derivation of positive and negative information during this natural induction:

$$\rightarrow \{\} \rightarrow Ev(0) \rightarrow \neg Ev(1) \rightarrow Ev(2) \rightarrow \neg Ev(3) \rightarrow \ldots$$

Thus, natural inductions correspond to sequences of three-valued structures $\langle \mathcal{I}_i \rangle_{0 \leq i \leq \beta}$. As we have seen, however, building a correct natural induction requires an induction order $\prec_{\propto}$. We will now show that we can exploit the additional information that is present in three-valued structures to construct $\mathcal{I}_{i+1}$ from $\mathcal{I}_i$, without using the induction order at all.

In the discussion below, we assume without loss of generality that a (finite) Δ contains exactly one rule $\forall \bar{x}(P(\bar{x}) \leftarrow \varphi)$ per defined predicate P (we can bundle finitely many multiple rules in one using a disjunction). We denote the body φ of this unique rule as φ_P. Below, when $A = P(\bar{a})$, we write $\varphi_A^{\mathfrak{A}}$ as a shorthand for $\varphi_P^{\mathfrak{A}[\bar{x}:\bar{a}]}$.

We first recall the formalization of the parametrized well-founded semantics of (Denecker and Vennekens 2007). We consider three-valued structures on $def(\Delta)$ on domain D. For each $P/n \in def(\Delta)$, $P^{\mathcal{I}}$ is a function $D^n \rightarrow \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$. The precision order on the three-valued structures is the point-wise extension of the partial order $\mathbf{u} \leq_{\mathrm{p}} \mathbf{t}$, $\mathbf{u} \leq_{\mathrm{p}} \mathbf{f}$. If U is a set of domain atoms, then $\mathcal{I}[U : \mathbf{f}]$ is identical to $\mathcal{I}$ except that every atom $A \in U$ is false; likewise for $\mathcal{I}[U : \mathbf{t}]$.

Rules are evaluated using a three-valued truth function $\varphi^{\mathcal{I}}$, defined for formulas φ and three-valued structures $\mathcal{I}$ that interpret all free symbols of φ. At this moment, this truth function is generic. All we require here is that it satisfies two properties: (1) if $\mathcal{I}_1 \leq_{\mathrm{p}} \mathcal{I}_2$ then $\varphi^{\mathcal{I}_1} \leq_{\mathrm{p}} \varphi^{\mathcal{I}_2}$ ($\leq_{\mathrm{p}}$-monotonicity) and (2) if $\mathfrak{A}$ is a (two-valued) structure, $\varphi^{\mathfrak{A}}$ is the standard truth value of φ in $\mathfrak{A}$.

Definition 17. *Given is a definition Δ and context $\mathcal{O}$. We say that $\mathcal{I}'$ is a Δ-refinement of $\mathcal{I}$ (in $\mathcal{O}$) if there is a set U of unknown domain atoms in $\mathcal{I}$ such that one of the following conditions is satisfied:*

- *for each $A \in U$, ${\varphi_A}^{\mathcal{I}} = \mathbf{t}$ and $\mathcal{I}' = \mathcal{I}[U : \mathbf{t}]$; or*
- *for each $A \in U$, ${\varphi_A}^{\mathcal{I}'} = \mathbf{f}$ and $\mathcal{I}' = \mathcal{I}[U : \mathbf{f}]$.*

As before, we used $\varphi^{\mathcal{I}}$ as a shorthand for $\varphi^{\mathcal{O} \circ \mathcal{I}}$.

Note the asymmetry: to derive A, its body must be true in $\mathcal{I}$ itself; but to derive $\neg A$, it suffices that its body is false in $\mathcal{I}[U : \mathbf{f}]$, i.e., we are free to assume that atoms are $\mathbf{f}$, as long as this prophecy turns out to fulfill itself. One recognizes the familiar concept of unfounded set in the second sort of refinement (Van Gelder, Ross, and Schlipf 1991).

Definition 18. *A* well-founded induction *of Δ in (context) $\mathcal{O}$ is a sequence $\langle \mathcal{I}_i \rangle_{0 \leq i \leq \beta}$ of three-valued $def(\Delta)$-structures with domain D such that $\mathcal{I}_0$ is $\mathbf{u}$ (the mapping of all defined domain atoms to $\mathbf{u}$), for each ordinal i, $\mathcal{I}_{i+1}$ is a Δ-refinement of $\mathcal{I}_i$ in $\mathcal{O}$, and for limit ordinals λ, $\mathcal{I}_\lambda$ is the $\leq_{\mathrm{p}}$-limit of $\langle \mathcal{I}_i \rangle_{i<\lambda}$. We call a well-founded induction terminal if its limit $\mathcal{I}_\beta$ has no strictly more precise Δ-refinement.*

A well-founded induction increases in precision, and hence, it has a $\leq_{\mathrm{p}}$-limit.

Theorem 1 ((Denecker and Vennekens 2007)). *All well-founded inductions in $\mathcal{O}$ can be extended to a terminal well-founded induction in $\mathcal{O}$. All terminal well-founded inductions in $\mathcal{O}$ converge.*

Definition 19. *The* well-founded model of Δ relative to $\mathcal{O}$ *(and the selected truth function) is the limit of any terminal well-founded induction of Δ in $\mathcal{O}$.*

We explain our intuition underlying well-founded inductions. It not only constructs elements of the defined set but also non-elements. Given such a partial set $\mathcal{I}_i$ at intermediate stage i, a new element A can be derived if what is known of the defined set, suffices to ascertain that A is derivable, that is ${\varphi_A}^{\mathcal{I}_i} = \mathbf{t}$. This explains the first refinement $\mathcal{I}_i[U : \mathbf{t}]$. A non-element A may be derived likewise, if it is certain that its condition does not hold, that is if ${\varphi_A}^{\mathcal{I}_i} = \mathbf{f}$. However, this is not enough. For example, in a positive definition such as Δ_{TC}, if we wait till ${\varphi_A}^{\mathcal{I}_i} = \mathbf{f}$ to derive that A is a non-element, we are waiting forever. A principle is needed that "looks ahead"; a principle that finds out that there are no more options to derive one or more domain atoms through natural induction from the current state. If U is such that $\mathcal{I}_i[U : \mathbf{f}]$ falsifies the premise φ_A of every $A \in U$, then no ways are left to construct such A's. This explains the second refinement $\mathcal{I}_i[U : \mathbf{f}]$.

The well-founded model is usually defined for normal logic programs, which correspond to the fragment of our rule based formalism that satisfies the following conditions: Δ is a set of normal rules (i.e., conjunctions of literals in the body), $pars(\Delta)$ contains only constant and function symbols (i.e., all predicates are defined), and $\mathcal{O}$ is the unique Herbrand interpretation of $pars(\Delta)$. If, in addition, we choose the standard Kleene three-valued truth function as our function $\varphi^{\mathcal{I}}$, then we obtain precisely the standard well-founded model of Δ.

An alternative three-valued truth function is the *supervaluation*, defined by $\varphi^{\mathcal{I}} = glb_{\leq_{\mathrm{p}}}\{\varphi^{\mathfrak{A}} \mid \mathcal{I} \leq_{\mathrm{p}} \mathfrak{A}$ and $\mathfrak{A}$ is two-valued $\}$ (van Fraassen 1966). It is more precise than Kleene's: e.g, for $P^{\mathcal{I}} = \mathbf{u}$ the supervaluation of $P \vee \neg P$ is $\mathbf{t}$ while Kleene's valuation is $\mathbf{u}$. We can also construct the well-founded model w.r.t. this truth function. This is the *ultimate* well-founded model of Δ relative to $\mathcal{O}$ (Denecker, Marek, and Truszczyński 2004). The standard well-founded model is less or equally precise than the ultimate one. This implies that if it is two-valued (i.e., maximally precise), it is also the ultimate well-founded model.

We are now ready to link well-founded and natural inductions. Let Δ be a definition by iterated induction over $\propto$ in context $\mathcal{O}$ with domain D. Let $\mathcal{N}$ be a natural induction that respects $\propto$:

$$\mathfrak{A}_0 \rightarrow \mathfrak{A}_1 \rightarrow \ldots \rightarrow \mathfrak{A}_\beta$$

With $\mathcal{N}$, we now associate the following sequence $WFI(\mathcal{N})$ of three-valued structures:

$$\mathcal{I}_0 \rightarrow \mathcal{I}_{0,1} \rightarrow \mathcal{I}_1 \rightarrow \mathcal{I}_{1,2} \rightarrow \mathcal{I}_2 \rightarrow \ldots \mathcal{I}_\beta \rightarrow \mathcal{I}_{\beta,\beta+1}$$

which we define by induction:

- $\mathcal{I}_0 = \mathbf{u}$ (for all domain atoms A, $A^{\mathcal{I}_0} = \mathbf{u}$).
- $\mathcal{I}_{i,i+1} = \mathcal{I}_i[U : \mathbf{f}]$ where $U = \{A \mid A^{\mathcal{I}_i} = \mathbf{u}, \mathfrak{A}_i \not\models_\Delta A$ and $\mathfrak{A}_i$ is saturated on $\{B \mid B \propto A\}\}$.
- $\mathcal{I}_{i+1} = \mathcal{I}_{i,i+1}[U : \mathbf{t}]$ where $U = \mathfrak{A}_{i+1} \setminus \mathfrak{A}_i$.
- $\mathcal{I}_\lambda$ is the $\leq_{\mathrm{p}}$-limit of its predecessors, for limit ordinals λ.

For the following theorem, we select supervaluation as three-valued truth function.

Theorem 2. *Let Δ be a definition by iterated induction over $\propto$ in $\mathcal{O}$. If $\mathcal{N}$ is a terminal natural induction of Δ in $\mathcal{O}$ that respects $\propto$ then the sequence $WFI(\mathcal{N})$ is a terminal well-founded induction and $\mathcal{I}_{\beta,\beta+1} = \mathfrak{A}_\beta$.*

It is on this theorem that the thesis of this paper rests. Indeed, it has the following theorem as a corollary, from which we directly obtain Proposition 6.

Theorem 3. *Let Δ be a definition by iterated induction over $\propto$ in $\mathcal{O}$. For any terminal natural induction $\mathcal{N}$ of Δ in $\mathcal{O}$ that respects $\propto$, its limit is the ultimate well-founded model of Δ in $\mathcal{O}$.*

Arguably, the construction process of a well-conceived formal or informal definition should result in a well-defined set. In our setting, this mean that the well-founded model in $\mathcal{O}$ should be two-valued. In this case we call Δ *total* in $\mathcal{O}$.

Corollary 1. *If Δ is a definition by iterated induction over $\propto$ in $\mathcal{O}$, then Δ is total in $\mathcal{O}$.*

Thus an iterated definition Δ has the desirable property of being total in $\mathcal{O}$. While the induction order has no semantical role in a definition, it gives insight in the structure of the definition and it gives a "parity check" of the correctness/totality of the definition.

From a technical point of view, Corollary 1 shows that one way to prove that a rule set has a 2-valued well-founded model is to prove that it has a dependency relation that monotonically orders it. This generalizes a condition in (Denecker and Ternovska 2008) and, to the best of our knowledge, it is the most general such condition currently known.

A comparison of natural and well-founded inductions. Below we write a well-founded induction as:

$$\rightarrow S_1 \rightarrow S_2 \rightarrow \dots$$

where S_i is either the set of atoms U such that $\mathfrak{A}_i = \mathfrak{A}_{i-1}[U : \mathbf{t}]$ or the set of negations $\neg A$ of atoms A in the set U such that $\mathfrak{A}_i = \mathfrak{A}_{i-1}[U : \mathbf{f}]$.

Example 15. *For the natural induction $\mathcal{N}$ of Example 3:*

$$\rightarrow R(a,a) \rightarrow R(b,c) \rightarrow R(c,b) \rightarrow R(b,b) \rightarrow R(c,c)$$

the corresponding well-founded induction $WFI(\mathcal{N})$ is:

$$\rightarrow \emptyset \rightarrow R(a,a) \rightarrow \emptyset \rightarrow R(b,c) \rightarrow \emptyset \rightarrow R(c,b) \rightarrow$$
$$\emptyset \rightarrow R(b,b) \rightarrow \emptyset \rightarrow R(c,c) \rightarrow \{\neg R(x,y) \mid (x,y) \notin R\}.$$

Example 16. *The natural induction of Example 2:*

$$\rightarrow Ev(0) \rightarrow Ev(2) \rightarrow \dots$$

has the corresponding well-founded induction $WFI(\mathcal{N})$:

$$\rightarrow \emptyset \rightarrow Ev(0) \rightarrow \neg Ev(1) \rightarrow Ev(2) \rightarrow \neg Ev(3) \rightarrow \dots$$

Many definitions admit well-founded inductions that are not natural inductions. Well-founded inductions can go dramatically "faster" than natural inductions.

Example 17. *A well-founded induction of the satisfaction definition $\Delta_{\models}$ that does not respect the induction order, but is nevertheless a correct well-founded induction is:*

$$\rightarrow Sat(\{P\}, P) \rightarrow \{Sat(\{P\}, P \vee \varphi) \mid \varphi \text{ a formula over } S\}$$

Indeed, after the first step $Sat(\{P\}, P)$ is true in the three-valued structure $\mathcal{I}_1$. Hence, we have $[Sat(\{P\}, P) \vee Sat(\{P\}, \varphi)]^{\mathcal{I}_1} = \mathbf{t}$ for every φ. It follows that $\mathfrak{A}_1[U : \mathbf{t}]$ is a refinement of $\mathfrak{A}_1$ for $U = \{Sat(\{P\}, P \vee \varphi) | \varphi$ a formula over $S\}$.

The well-founded induction in the example matches how humans derive $\mathfrak{A} \models \varphi \vee \psi$: if a disjunct φ is derived to be satisfied, we jump to the conclusion that $\varphi \vee \psi$ is satisfied, even if the value of ψ is unknown. Strictly speaking, here we are violating the induction order! It is nevertheless safe: any fact derived during a well-founded induction is correct.

The next example shows that there are sensible definitions that are not iterated definitions.

Example 18. *Let $\mathcal{O}$ be the natural number context of Example 2 and Δ_{ev1} the following variant definition of even numbers and an additional predicate $Next$:*

$$\left\{ \begin{array}{ll} \forall x \forall y (Next(x,y) \leftarrow x = y + 1) \\ \forall x (Even(x) & \leftarrow x = 0 \vee \\ & \quad \exists y (Next(x,y) \wedge \neg Even(y))) \end{array} \right\}$$

The definition defines and uses $Next$ as the successor predicate. Its well-founded model $\mathfrak{A}$ interprets $Next$ as the successor relation and $Even$ as the set of even numbers, as expected. However, this definition is not an iterated induction. Indeed, in any dependency relation of Δ_{ev1}, it holds that $Ev(n) \propto Ev(m)$ for all $n, m \in \mathbb{N}$. This follows from the fact that $Ev(m)$ is derivable in the structure $\{Next(m,n)\}$ but not in $\{Next(m,n), Ev(n)\}$. However, as the same two structures show, no such $\propto$ monotonically orders Δ_{ev1} in $\mathcal{O}$.

The above example shows a disturbing brittleness of the concept of a definition by iterated induction as defined in Definition 16, that fortunately is not shared with the rule formalism under the well-founded semantics.

To summarize, the concept of well-founded induction seems to provide a superior formalization of the induction process, one that is independent of the order, faster, more like humans reason on informal definitions and more robust under innocent syntactical variance.

Standard versus ultimate well-founded semantics. The result of the previous section (the convergence of natural and well-founded inductions) are formulated for the ultimate well-founded semantics, the one derived using the supervaluation. Here, we investigate to what extend the result holds for the standard well-founded semantics.

The first observation is that if the standard well-founded model is two-valued, it is the ultimate well-founded model. It follows that if the standard well-founded model of an iterated inductive definition is two-valued, it is the structure defined by this definition. In general however, an iterated inductive definition as defined here, may not have a two-valued standard well-founded model. An example is $\Delta = \{P \leftarrow P \vee \neg P\}$: it is monotone according to Definition 8 and its defined structure and ultimate well-founded model is $\{P\}$. On the other hand, P is unknown in Δ's standard well-founded model. Thus, the standard well-founded semantics is too weak for this definition.

How serious is the gap? To address this, we make two observations. First, standard and ultimate well-founded semantics break apart only in quite specific circumstances: when there is a *case splitting* involving one or multiple rule bodies leading to a loop over negation (see above example). This is a pattern that we *never* observe in our applications. In practice, standard and ultimate semantics coincide.

The second observation might explain the first one. Although our formal definitions of monotone, ordered and iterated inductive definitions admit rule sets showing such a pattern, the conventions of informal definitions seem to forbid it. Consider, for instance, a variant of the informal definition of satisfaction (Definition 2), that is obtained by replacing its first rule by the following one:

- $I \models P$ if either $I \models P$ and $P \in I$, or if $I \not\models P$ and $P \in I$;

Or after splitting this rule:

- $I \models P$ if $I \models P$ and $P \in I$;
- $I \models P$ if $I \not\models P$ and $P \in I$;

One could argue that these new rules "obviously" are equivalent to the original one by appealing to the fact that "$I \models P$

or $I \not\models P$" is tautologically true. But the modified rules mismatch the induction order and we think such a definition would not be accepted in mathematical text (we would not). This suggests that an implicit convention of informal definitions is not yet explicit in our theory: our definitions are too liberal. We think that some conventions regarding informal definitions remain to be discovered, and that their formalization will preclude rule sets of the above form and lead to a notion of iterated inductive definition that will have a two-valued standard well-founded model. In other words, we think that if one follows the conventions of informal inductive definitions while expressing them as a formal rule set, the standard well-founded model will be two-valued.

In summary, the computationally cheaper standard well-founded semantics, in practice, almost always coincides with the more expensive ultimate semantics. Moreover, in cases where they differ, it is far from obvious that the rule sets correspond to legal inductive definitions, and that the answer provided by the ultimate semantics is better than that of the standard well-founded model.

Conclusion. The paper is relevant from two angles. Informal (inductive) definitions are an important form of human expert knowledge and have many applications in scientific texts as well as in knowledge representation. In this paper we studied the general knowledge representation problem of representing informal definitions. From a very different angle, this paper studies the old but unresolved problem of the *informal semantics* of logic programming.

The first part of the paper was concerned with formalizing familiar but implicit intuitions and conventions of informal definitions. This was done by a series of definitions "from first principles" (Definitions 3-16). The central concept is our formalization of the induction process, the "natural induction". The analysis exposed non-trivial properties of informal definitions: the all-importance of convergence of the induction process, the role of the induction order, its tight link with the rules and ultimately, its irrelevance. To the best of our knowledge, our analysis provides the most detailed semantical account of the studied forms of informal definitions to date. The second part of the paper showed that the well-founded inductions underlying the well-founded semantics provide a natural, efficient and robust generalization of natural inductions. Taken together, this provides, for the first time, a direct argument why rule formalisms under the well-founded semantics correctly formalize informal inductive definitions.

We finish this paper with a speculation. Given that our conscious understanding of inductive definitions is quite partial, where does our proficiency to reason with them come from? Indeed, it is not because someone consciously (and erroneously) believes that the satisfaction relation is the least set closed under the familiar rules, that he or she is not capable to correctly reason about it. Indeed, daily practice provides ample evidence to the contrary. Arguably, this shows a gap between our conscious and subconscious understanding of inductive definitions. However, in contrast to similar gaps, e.g., in the context of reasoning on statistical information (Kahneman 2011), it is here the (slow) conscious understanding that is erroneous, and the (fast) subconscious reasoning that is correct (people do derive that $\{P\} \not\models \neg P$)! The explanation that we see for this phenomenon is that the principle(s) of inductive definition is not a primitive of human cognition, but is just a manifestation of a deeper common sense principle that is hard wired in the cognitive machinery of the human mind. As we have argued in several papers in the past, we believe that this base cognitive principle is that of causal reasoning: the induction process as a causal process that creates the defined object.

References

Buchholz, W.; Feferman, S.; Pohlers, W.; and Sieg, W. 1981. *Iterated Inductive Definitions and Subsystems of Analysis : Recent Proof-Theoretical Studies*, volume 897 of *Lecture Notes in Mathematics*. Springer.

Denecker, M., and Ternovska, E. 2007. Inductive situation calculus. *Artificial Intelligence* 171(5-6):332–360.

Denecker, M., and Ternovska, E. 2008. A logic of nonmonotone inductive definitions. *ACM Transactions on Computational Logic (TOCL)* 9(2):14:1–14:52.

Denecker, M., and Vennekens, J. 2007. Well-founded semantics and the algebraic theory of non-monotone inductive definitions. In Baral, C.; Brewka, G.; and Schlipf, J. S., eds., *LPNMR*, volume 4483 of *LNCS*, 84–96. Springer.

Denecker, M.; Bruynooghe, M.; and Marek, V. W. 2001. Logic programming revisited: Logic programs as inductive definitions. *ACM Transactions on Computational Logic (TOCL)* 2(4):623–654.

Denecker, M.; Marek, V. W.; and Truszczyński, M. 2004. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Information and Computation* 192(1):84–121.

Denecker, M. 1998. The well-founded semantics is the principle of inductive definition. In Dix, J.; del Cerro, L. F.; and Furbach, U., eds., *JELIA*, volume 1489 of *LNCS*, 1–16. Springer.

Kahneman, D. 2011. *Thinking, fast and slow*. Farrar, Strays and Giroux.

Marek, W. 1989. Stable theories in autoepistemic logic. *Fundamenta Informaticae* 12(2):243–254.

Martin-Löf, P. 1971. Hauptsatz for the intuitionistic theory of iterated inductive definitions. In Fenstad, J., ed., *Second Scandinavian Logic Symposium*, 179–216.

Przymusinski, T. C. 1988. On the declarative semantics of deductive databases and logic programs. In *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann. 193–216.

van Fraassen, B. 1966. Singular terms, truth-value gaps and free logic. *Journal of Philosophy* 63(17):481–495.

Van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM* 38(3):620–650.

Van Gelder, A. 1993. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences* 47(1):185–221.

The Semantics of Gringo and Infinitary Propositional Formulas

Amelia Harrison and **Vladimir Lifschitz** and **Fangkai Yang**
University of Texas, Austin, Texas, USA
{ameliaj, vl, fkyang}@cs.utexas.edu

Abstract

Input languages of answer set solvers are based on the mathematically simple concept of a stable model. But many useful constructs available in these languages, including local variables, conditional literals, and aggregates, cannot be easily explained in terms of stable models in the sense of the original definition of this concept and its straightforward generalizations. Manuals written by designers of answer set solvers usually explain such constructs using examples and informal comments that appeal to the user's intuition, without references to any precise semantics. We propose to approach the problem of defining the semantics of GRINGO programs by translating them into the language of infinitary propositional formulas. This semantics allows us to study equivalent transformations of GRINGO programs using natural deduction in infinitary propositional logic, so that the properties of these programs can be more precisely characterized. In this way, we aim to create a foundation on which important issues such as the correctness of GRINGO programs and optimization methods may be more formally studied.

1 Introduction

In this note, Gringo is the name of the input language of the grounder GRINGO,[1] which is used as the front end in many answer set programming (ASP) systems. Several releases of GRINGO have been made public, and more may be coming in the future; accordingly, we can distinguish between several "dialects" of the language Gringo. We concentrate here on Version 4, released in March of 2013. (It differs from Version 3, described in the *User's Guide* dated October 4, 2010,[2] in several ways, including the approach to aggregates—it is modified as proposed by the ASP Standardization Working Group.[3])

The basis of Gringo is the language of logic programs with negation as failure, with the syntax and semantics defined in (Gelfond and Lifschitz 1988). Our goal here is to extend that semantics to a larger subset of Gringo. Specifically, we would like to cover arithmetical functions and comparisons, conditions, and aggregates.[4]

Our proposal is based on the informal and sometimes incomplete description of the language in the *User's Guide*, on the discussion of ASP programming constructs in (Gebser et al. 2012), on experiments with GRINGO, and on the clarifications provided in response to our questions by its designers.

The proposed semantics uses a translation from Gringo into the language of infinitary propositional formulas—propositional formulas with infinitely long conjunctions and disjunctions. Including infinitary formulas is essential, as we will see, when conditions or aggregates use variables ranging over infinite sets (for instance, over integers).

Alternatively, the semantics of Gringo programs can be approached using quantified equilibrium logic (Pearce and Valverde 2004) or its syntactic counterpart defined in (Ferraris, Lee, and Lifschitz 2011). This method involves translating rules into the language of first-order logic. For instance, the rule

$$p(Y) \leftarrow \mathit{count}\{X, Y : q(X, Y)\} \geq 1 \tag{1}$$

can be represented by the sentence

$$\forall y(\exists x Q(x, y) \rightarrow P(y)).$$

However, this approach is not very general. For instance, it is not clear how to represent the rule

$$\begin{array}{l}\mathit{total\_hours}(N) \\ \quad \leftarrow \mathit{sum}\{H, C : \mathit{enroll}(C), \mathit{hours}(H, C)\} = N\end{array} \tag{2}$$

from Section 3.1.10 of the Gringo 3 *User's Guide* with a first-order formula. One reason is that the aggregate *sum* is used here instead of *count*. The second difficulty is that the variable N is used in it rather than a constant.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] http://potassco.sourceforge.net/.

[2] The *User's Guide* can be downloaded from the Potassco website (Footnote 1). It is posted also at http://www.cs.utexas.edu/users/vl/teaching/lbai/clingo_guide.pdf.

[3] https://www.mat.unical.it/aspcomp2013/ASPStandardization.

[4] The subset of Gringo discussed in this note includes also constraints, disjunctive rules, and choice rules, treated along the lines of (Gelfond and Lifschitz 1991) and (Ferraris and Lifschitz 2005). The first of these papers introduces also "classical" (or "strong") negation—a useful feature that we do not include. Extending our semantics of Gringo to programs with classical negation is straightforward, using the process of eliminating classical negation in favor of additional atoms described in (Gelfond and Lifschitz 1991, Section 4).

General aggregate expressions allowed in Gringo can be represented by first-order formulas with generalized quantifiers. Stable models of formulas with generalized quantifiers are defined by Lee and Meng (2012a, 2012b, 2012c). The advantage of infinitary propositional formulas as the target language is that we have more mathematical results regarding the properties of these formulas. We may be able to prove, for instance, that two Gringo programs have the same stable models by observing that the corresponding infinitary formulas are equivalent in one of the natural deduction systems discussed in (Harrison, Lifschitz, and Truszczynski 2014). We give here several examples of reasoning about Gringo programs based on this idea.

The process of converting Gringo programs into infinitary propositional formulas defined in this note uses substitutions to eliminate variables. This form of grounding is quite different, of course, from the process of intelligent instantiation implemented in GRINGO and other grounders. Mathematically, it is much simpler than intelligent instantiation; as a computational procedure, it is much less efficient, not to mention the fact that sometimes it produces infinite objects. Like grounding in the original definition of a stable model (Gelfond and Lifschitz 1988), it is modular, in the sense that it applies to the program rule by rule, and it is applicable even if the program is not safe. From this perspective, GRINGO's safety requirement is an implementation restriction.

Our description of the syntax of Gringo disregards some of the features related to representing programs as strings of ASCII characters, such as using `:-` to separate the head from the body, using semicolons, rather than parentheses, to indicate the boundaries of a conditional literal, and representing falsity (which we denote here by $\bot$) as `#false`. Since the subset of Gringo discussed in this note does not include assignments, we can disregard also the requirement that equality be represented by two characters `==`.

The syntax and semantics of the fragment of Gringo studied in this note are defined in Sections 2 and 3. In Sections 4 and 5 we give examples of reasoning about stable models of Gringo programs on the basis of the proposed semantics. The semantics of aggregate expressions, based on (Ferraris 2005), is rather complicated, and in Section 6 we show how it can be simplified in the case of monotone and anti-monotone aggregate expressions. Definitions and theorems regarding stable models of infinitary formulas used in this paper are reproduced in Section 3.1 and in the appendix.

A preliminary report on this project was presented at the 2013 Workshop on Answer Set Programming and Other Computing Paradigms.

2 Syntax

We begin with a signature σ in the sense of first-order logic that includes, among others,

- numerals—object constants representing all integers,
- arithmetical functions—binary function constants $+$, $-$, $\times$,
- comparisons—binary predicate constants $<$, $>$, $\leq$, $\geq$.

We will identify numerals with the corresponding elements of the set $\mathbf{Z}$ of integers. Object, function, and predicate symbols which do not fall into any of the categories listed above will be called *symbolic*. A term over σ is *arithmetical* if it does not contain symbolic object or function constants. A ground term is *precomputed* if it does not contain arithmetical functions.

We assume that in addition to the signature, a set of symbols called *aggregate names* is specified, and that for each aggregate name α, the symbol $\widehat{\alpha}$ represents *the function denoted by* α. This function maps every tuple of precomputed terms to an element of $\mathbf{Z} \cup \{\infty, -\infty\}$.

Examples The functions denoted by the aggregate names *count*, *max*, and *sum* are defined as follows. For any set T of tuples of precomputed terms,

- $\widehat{count}(T)$ is the cardinality of T if T is finite, and ∞ otherwise;
- $\widehat{max}(T)$ is the least upper bound of the set of the integers t_1 over all tuples $(t_1, \dots, t_m)$ from T in which t_1 is an integer;
- $\widehat{sum}(T)$ is the sum of the integers t_1 over all tuples $(t_1, \dots, t_m)$ from T in which t_1 is a positive integer; it is ∞ if there are infinitely many such tuples.[5]

A *literal* is an expression of one of the forms

$$p(t_1, \dots, t_k),\ t_1 = t_2,\ t_1 \neq t_2,\ \mathit{not}\ p(t_1, \dots, t_k)$$

where p is a symbolic predicate constant of arity k, and each t_i is a term over σ, or of the form

$$t_1 \prec t_2$$

where $\prec$ is a comparison, and t_1, t_2 are arithmetical terms. A *conditional literal* is an expression of the form $H : \mathbf{L}$, where H is a literal or the symbol $\bot$, and $\mathbf{L}$ is a list of literals, possibly empty. The members of $\mathbf{L}$ will be called *conditions*. If $\mathbf{L}$ is empty then we will drop the colon after H, so that every literal can be viewed as a conditional literal.

Example If *available* and *person* are unary predicate symbols then

$$\mathit{available}(X) : \mathit{person}(X)$$

and

$$\bot : (\mathit{person}(X), \mathit{not}\ \mathit{available}(X))$$

are conditional literals.

An *aggregate expression* is an expression of the form

$$\alpha\{\mathbf{t}_1 : \mathbf{L}_1; \dots; \mathbf{t}_n : \mathbf{L}_n\} \prec s \tag{3}$$

($n > 0$), where

- α is an aggregate name,
- each $\mathbf{t}_i$ is a non-empty list of terms,
- each $\mathbf{L}_i$ is a non-empty list of literals,

[5] To allow negative numbers in this example, we would have to define summation for a set that contains both infinitely many positive numbers and infinitely many negative numbers. It is unclear how to do this in a natural way.

- $\prec$ is a comparison or one of the symbols $=, \neq$,
- s is an arithmetical term.

Example If *enroll* is a unary predicate symbol and *hours* is a binary predicate symbol then

$$sum\{H, C : enroll(C), hours(H, C)\} = N$$

is an aggregate expression.

A *rule* is an expression of the form

$$H_1 \mid \cdots \mid H_m \leftarrow B_1, \ldots, B_n \tag{4}$$

($m, n \geq 0$), where each H_i is a conditional literal, and each B_j is a conditional literal or an aggregate expression. A *program* is a finite set of rules.

If p is a symbolic predicate constant of arity k, and $\mathbf{t}$ is a k-tuple of terms, then

$$\{p(\mathbf{t})\} \leftarrow B_1, \ldots, B_n$$

is shorthand for

$$p(\mathbf{t}) \mid not\ p(\mathbf{t}) \leftarrow B_1, \ldots, B_n.$$

Example For any positive integer n,

$$\begin{array}{ll} \{p(i)\} & \leftarrow \\ & \leftarrow p(X), p(Y), p(X+Y) \end{array} \qquad (i = 1, \ldots, n), \tag{5}$$

is a program.

3 Semantics

We will define the semantics of Gringo using a syntactic transformation τ. It converts Gringo rules into infinitary propositional combinations of atoms of the form $p(\mathbf{t})$, where p is a symbolic predicate constant, and $\mathbf{t}$ is a tuple of precomputed terms. Then the stable models of a program will be defined as stable models, in the sense of (Truszczynski 2012), of the set consisting of the translations of all rules of the program. Truszczynski's definition of stable models for infinitary propositional formulas is reviewed below.

Prior to defining the translation τ for rules, we will define it for ground literals, conditional literals, and aggregate expressions.

3.1 Review: Stable Models of Infinitary Formulas

Let σ be a propositional signature, that is, a set of propositional atoms. The sets $\mathcal{F}_0^\sigma, \mathcal{F}_1^\sigma, \ldots$ are defined as follows:

- $\mathcal{F}_0^\sigma = \sigma$,
- $\mathcal{F}_{i+1}^\sigma$ is obtained from $\mathcal{F}_i^\sigma$ by adding expressions $\mathcal{H}^\wedge$ and $\mathcal{H}^\vee$ for all subsets $\mathcal{H}$ of $\mathcal{F}_i^\sigma$, and expressions $F \to G$ for all $F, G \in \mathcal{F}_i^\sigma$.

The elements of $\bigcup_{i=0}^\infty \mathcal{F}_i^\sigma$ are called *(infinitary) formulas* over σ.

A set $\mathcal{H}$ of formulas is *bounded* if it is contained in one of the sets $\mathcal{F}_i^\sigma$. For a bounded set $\mathcal{H}$ of formulas, $\mathcal{H}^\wedge$ and $\mathcal{H}^\vee$ are infinitary formulas.

The symbols $\top$ and $\bot$ will be understood as abbreviations for $\emptyset^\wedge$ and $\emptyset^\vee$ respectively; $\neg F$ stands for $F \to \bot$, and $F \leftrightarrow G$ stands for $(F \to G) \wedge (G \to F)$.

We will write $\{F, G\}^\wedge$ as $F \wedge G$, and $\{F, G\}^\vee$ as $F \vee G$. This convention allows us to view finite propositional formulas over σ as a special case of infinitary formulas. For any bounded family $\{F_\alpha\}_{\alpha \in A}$ of formulas, we denote the formula $\{F_\alpha : \alpha \in A\}^\wedge$ by $\bigwedge_{\alpha \in A} F_\alpha$, and similarly for disjunctions.

Subsets of a signature σ will be also called its *interpretations*. The satisfaction relation between an interpretation and a formula is defined in a natural way.

The *reduct* F^I of a formula F w.r.t. an interpretation I is defined as follows:

- For $p \in \sigma$, $p^I = \bot$ if $I \not\models p$; otherwise $p^I = p$.
- $(\mathcal{H}^\wedge)^I = \{G^I \mid G \in \mathcal{H}\}^\wedge$.
- $(\mathcal{H}^\vee)^I = \{G^I \mid G \in \mathcal{H}\}^\vee$.
- $(G \to H)^I = \bot$ if $I \not\models G \to H$; otherwise $(G \to H)^I = G^I \to H^I$.

An interpretation I is a *stable model* of a set $\mathcal{H}$ of formulas if it is minimal w.r.t. set inclusion among the interpretations satisfying the reducts of all formulas from $\mathcal{H}$.

3.2 Semantics of Well-Formed Ground Literals

A term t is *well-formed* if it contains neither symbolic object constants nor symbolic function constants in the scope of arithmetical functions. For instance, all arithmetical terms and all precomputed terms are well-formed; $c{+}2$ is not well-formed. The definition of "well-formed" for literals, aggregate expressions, and so forth is the same.

For every well-formed ground term t, by $[t]$ we denote the precomputed term obtained from t by evaluating all arithmetical functions, and similarly for tuples of terms. For instance, $[f(2{+}2)]$ is $f(4)$.

The translation τL of a well-formed ground literal L is defined as follows:

- $\tau(p(\mathbf{t}))$ is $p([\mathbf{t}])$;
- $\tau(t_1 \prec t_2)$, where $\prec$ is $=$, $\neq$, or a comparison is $\top$ if the relation $\prec$ holds between $[t_1]$ and $[t_2]$, and $\bot$ otherwise;
- $\tau(not\ A)$ is $\neg \tau A$.

For instance, $\tau(not\ \ p(f(2\ +\ 2)))$ is $\neg p(f(4))$, and $\tau(2{+}2{=}\,4)$ is $\top$.

Furthermore, $\tau\bot$ stands for $\bot$, and, for any list $\mathbf{L}$ of ground literals, $\tau\mathbf{L}$ is the conjunction of the formulas τL for all members L of $\mathbf{L}$.

3.3 Global Variables

About a variable we say that it is *global*

- in a conditional literal $H : \mathbf{L}$, if it occurs in H but does not occur in $\mathbf{L}$;
- in an aggregate expression (3), if it occurs in the term s;
- in a rule (4), if it is global in at least one of the expressions H_i, B_j.

For instance, the head of the rule (2) is a literal with the global variable N, and its body is an aggregate expression with the global variable N. Consequently N is global in the rule as well.

A conditional literal, an aggregate expression, or a rule is *closed* if it has no global variables. An *instance* of a rule R is any well-formed closed rule that can be obtained from R by substituting precomputed terms for global variables. For instance,

$$\begin{array}{l} \mathit{total\_hours}(6) \leftarrow \\ \quad \mathit{sum}\{H, C : \mathit{enroll}(C), \mathit{hours}(H, C)\} = 6 \end{array}$$

is an instance of rule (2). It is clear that if a rule is not well-formed then it has no instances.

3.4 Semantics of Closed Conditional Literals

If t is a term, $\mathbf{x}$ is a tuple of distinct variables, and $\mathbf{r}$ is a tuple of terms of the same length as $\mathbf{x}$, then the term obtained from t by substituting $\mathbf{r}$ for $\mathbf{x}$ will be denoted by $t^{\mathbf{x}}_{\mathbf{r}}$. Similar notation will be used for the result of substituting $\mathbf{r}$ for $\mathbf{x}$ in expressions of other kinds, such as literals and lists of literals.

The result of applying τ to a closed conditional literal $H : \mathbf{L}$ is the conjunction of the formulas

$$\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \rightarrow \tau(H^{\mathbf{x}}_{\mathbf{r}})$$

where $\mathbf{x}$ is the list of variables occurring in $H : \mathbf{L}$, over all tuples $\mathbf{r}$ of precomputed terms of the same length as $\mathbf{x}$ such that both $\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}$ and $H^{\mathbf{x}}_{\mathbf{r}}$ are well-formed. For instance,

$$\tau(\mathit{available}(X) : \mathit{person}(X))$$

is the conjunction of the formulas

$$\mathit{person}(r) \rightarrow \mathit{available}(r)$$

over all precomputed terms r;

$$\tau(\bot : p(2 \times X))$$

is the conjunction of the formulas $\neg p(2 \times i)$ over all numerals i.

When a conditional literal occurs in the head of a rule, it is translated in a different way.[6] By $\tau_h(H : \mathbf{L})$ we denote the disjunction of the formulas

$$(\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \rightarrow \tau(H^{\mathbf{x}}_{\mathbf{r}})) \wedge \neg\neg(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}})$$

where $\mathbf{x}$ and $\mathbf{r}$ are as above. For instance,

$$\tau_h(\mathit{available}(X) : \mathit{person}(X))$$

is the disjunction of the formulas

$$(\mathit{person}(r) \rightarrow \mathit{available}(r)) \wedge \neg\neg \mathit{person}(r)$$

over all precomputed terms r.

3.5 Semantics of Closed Aggregate Expressions

In this section, the semantics of ground aggregates proposed in (Ferraris 2005, Section 4.1) is adapted to closed aggregate expressions. Let E be a closed aggregate expression (3), and let $\mathbf{x}_i$ be the list of variables occurring in $\mathbf{t}_i : \mathbf{L}_i$ $(1 \leq i \leq n)$. By A_i we denote the set of tuples $\mathbf{r}$ of precomputed terms of the same length as $\mathbf{x}_i$ such that both $(\mathbf{t}_i)^{\mathbf{x}_i}_{\mathbf{r}}$ and $(\mathbf{L}_i)^{\mathbf{x}_i}_{\mathbf{r}}$ are well-formed. By A we denote the set $\{(i, \mathbf{r}) : i \in \{1, \ldots, n\}, \mathbf{r} \in A_i\}$.

About a subset Δ of A we say that it *justifies* E if the relation $\prec$ holds between $\widehat{\alpha}(\{[(\mathbf{t}_i)^{\mathbf{x}_i}_{\mathbf{r}}] : (i, \mathbf{r}) \in \Delta\})$ and $[s]$. We define τE as the conjunction of the implications

$$\bigwedge_{(i,\mathbf{r}) \in \Delta} \tau((\mathbf{L}_i)^{\mathbf{x}_i}_{\mathbf{r}}) \rightarrow \bigvee_{(i,\mathbf{r}) \in A \setminus \Delta} \tau((\mathbf{L}_i)^{\mathbf{x}_i}_{\mathbf{r}}) \qquad (6)$$

over all sets Δ that do not justify E.

Example Consider the aggregate expression

$$\mathit{count}\{X : p(X);\ N : q(2 \times N)\} > 0. \qquad (7)$$

In this case, A_1 is the set of all precomputed terms, and A_2 is the set of all numerals. Therefore, A is the set of all pairs whose first member is either 1 or 2 and whose second member is a precomputed term when the first member is 1 and is a numeral when the first member is 2. A subset Δ of A justifies (7) iff

$$\widehat{\mathit{count}}\left(\left\{X^X_{\mathbf{r}} : (1, \mathbf{r}) \in \Delta\right\} \cup \left\{N^N_{\mathbf{r}} : (2, \mathbf{r}) \in \Delta\right\}\right) > 0.$$

Since

$$\{X^X_{\mathbf{r}} : (1, \mathbf{r}) \in \Delta\} = \{\mathbf{r} : (1, \mathbf{r}) \in \Delta\}$$

and

$$\{N^N_{\mathbf{r}} : (2, \mathbf{r}) \in \Delta\} = \{\mathbf{r} : (2, \mathbf{r}) \in \Delta\}$$

we conclude that the set Δ justifies (7) iff

$$\widehat{\mathit{count}}(\Delta) > 0,$$

that is, iff the set Δ is non-empty. Since the only $\Delta \subseteq A$ that does not justify (7) is $\emptyset$, the result of applying τ to (7) has only one conjunctive term of the form (6). Its antecedent is the trivial empty conjunction; its consequent is

$$\bigvee_{x \in A_1} p(x) \vee \bigvee_{n \in A_2} q(2n).$$

If E is an aggregate expression of the form

$$\alpha\{\mathbf{t}_1 : \mathbf{L}_1; \ldots; \mathbf{t}_n : \mathbf{L}_n\} = s,$$

$E_{\leq}$ is

$$\alpha\{\mathbf{t}_1 : \mathbf{L}_1; \ldots; \mathbf{t}_n : \mathbf{L}_n\} \leq s,$$

and $E_{\geq}$ is

$$\alpha\{\mathbf{t}_1 : \mathbf{L}_1; \ldots; \mathbf{t}_n : \mathbf{L}_n\} \geq s,$$

then $\tau(E)$ has the same conjunctive terms as

$$\tau(E_{\leq}) \wedge \tau(E_{\geq}).$$

In this sense, E has the same meaning of the conjunction of $E_{\leq}$ and $E_{\geq}$.

In the special case when E has the form $\alpha\{\mathbf{t} : \mathbf{L}\} \prec s$, the definition of τE can be stated as follows. Let $\mathbf{x}$ be the list of variables occurring in $\mathbf{t} : \mathbf{L}$, and let A be the set of tuples $\mathbf{r}$ of precomputed terms of the same length as $\mathbf{x}$ such that both $\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}$ and $\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}$ are well-formed. A subset Δ of A *justifies* E if the relation $\prec$ holds between $\widehat{\alpha}(\{[\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}] : \mathbf{r} \in \Delta\})$ and $[s]$. We define τE as the conjunction of the implications

$$\bigwedge_{\mathbf{r} \in \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \rightarrow \bigvee_{\mathbf{r} \in A \setminus \Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}})$$

[6] Torsten Schaub, Martin Gebser, and Roland Kaminski, personal communication, October 2013.

over all subsets Δ of A that do not justify E.

Examples Consider the aggregate expression

$$\mathit{sum}\{H, C : \mathit{enroll}(C), \mathit{hours}(H, C)\} = 6. \tag{8}$$

In this case, A is the set of pairs (h, c) of precomputed terms. The subset $\{(3, \mathit{cs101}), (3, \mathit{cs102})\}$ of A justifies (8), because

$$\begin{aligned}\widehat{\mathit{sum}}\left(\left\{(H,C)^{H,C}_{3,\mathit{cs101}}, (H,C)^{H,C}_{3,\mathit{cs102}}\right\}\right) &= \\ \widehat{\mathit{sum}}(\{(3, \mathit{cs101}), (3, \mathit{cs102})\}) &= 3 + 3 = 6.\end{aligned}$$

More generally, a set of pairs of precomputed terms justifies (8) whenever it contains finitely many pairs (h, c) in which h is a positive integer, and the sum of the integers h over all these pairs is 6. The conjunctive terms of τE are the formulas

$$\begin{aligned}\bigwedge_{(h,c)\in\Delta} (\mathit{enroll}(c) \wedge \mathit{hours}(h, c)) \rightarrow \\ \bigvee_{(h,c)\notin\Delta} (\mathit{enroll}(c) \wedge \mathit{hours}(h, c)).\end{aligned} \tag{9}$$

The conjunctive term corresponding to $\{(3, \mathit{cs101})\}$ as Δ says: if I am enrolled in CS101 for 3 hours then I am enrolled in at least one other course. In Section 6.1 we will see how the result of applying τ to (8) can be simplified.

As a final example illustrating the semantics of aggregate expressions, consider a pair of closed aggregate expressions of the form

$$\mathit{sum}\{1, X : \mathbf{L}\} \prec s \quad \text{and} \quad \mathit{count}\{X : \mathbf{L}\} \prec s, \tag{10}$$

where the tuple $\mathbf{L}$ of literals contains no variables other than X. Intuitively, these expressions have the same meaning; accordingly, the translation τ transforms them to identical formulas. Note first that the result of applying τ to each of them is the conjunction of implications of the form

$$\bigwedge_{r\in\Delta} \tau(\mathbf{L}^X_r) \rightarrow \bigvee_{r\in A\setminus\Delta} \tau(\mathbf{L}^X_r).$$

To check that both conjunctions extend over the same set Δ of precomputed terms, observe that

$$\widehat{\mathit{sum}}(\{(1, X)^X_r : r \in \Delta\}) = \widehat{\mathit{sum}}(\{(1, r) : r \in \Delta\})$$

and

$$\begin{aligned}\widehat{\mathit{count}}(\{X^X_r : r \in \Delta\}) &= \widehat{\mathit{count}}(\{r : r \in \Delta\}) \\ &= \widehat{\mathit{count}}(\Delta),\end{aligned}$$

so that both expressions evaluate to the cardinality of Δ.

3.6 Semantics of Rules and Programs

For any rule R, τR stands for the conjunction of the formulas

$$\tau B_1 \wedge \cdots \wedge \tau B_n \rightarrow \tau_h H_1 \vee \cdots \vee \tau_h H_m$$

for all instances (4) of R. A *stable model* of a program Π is a stable model, in the sense of (Truszczynski 2012), of the set consisting of the formulas τR for all rules R of Π.

Example Consider the rules of program (5). If R is the rule $\{p(i)\}$ then τR is

$$p(i) \vee \neg p(i) \tag{11}$$

$(i = 1, \ldots, n)$. If R is the rule

$$\leftarrow p(X), p(Y), p(X+Y)$$

then the instances of R are rules of the form

$$\leftarrow p(i), p(j), p(i+j)$$

for all numerals i, j. (Substituting precomputed ground terms other than numerals would produce a rule that is not well-formed.) Consequently τR is in this case the infinite conjunction

$$\bigwedge_{\substack{i,j,k\in\mathbf{Z}\\ i+j=k}} \neg(p(i) \wedge p(j) \wedge p(k)). \tag{12}$$

The stable models of program (5) are the stable models of formulas (11), (12), that is, sets of the form $\{p(i) : i \in S\}$ for all sum-free subsets S of $\{1, \ldots, n\}$.

4 Reasoning about Gringo Programs

As an example, we will show how we can reason about the stable models of the program Π consisting of rule (2) and an arbitrary finite set S of atoms of the forms $\mathit{hours}(h, c)$ and $\mathit{enroll}(c)$, where h is a positive integer and c is an object constant.

Let h^* be the sum of the numbers h over all pairs (h, c) such that both $\mathit{hours}(h, c)$ and $\mathit{enroll}(c)$ are in S. As could be expected, the set

$$S \cup \{\mathit{total\_hours}(h^*)\} \tag{13}$$

is the only stable model of Π.

To justify this claim, consider the result of applying τ to Π. This set of formulas consists of the atoms S and the implications

$$\begin{aligned}\tau(\mathit{sum}\{H, C : \mathit{enroll}(C), \mathit{hours}(H, C)\} = i) \\ \rightarrow \mathit{total\_hours}(i)\end{aligned} \tag{14}$$

for all numerals i. The conjunction of the formulas in $\tau\Pi$ is a supertight infinitary program (see Appendix A.1). Consequently, it is sufficient to show that (13) is the only supported model of $\tau\Pi$, which can be derived from the following fact:

Lemma *Let I be an interpretation such that S is the set of all atoms of the forms hours(h,c) and enroll(c) that are satisfied by I. For any numeral i, I satisfies the antecedent of (14) if and only if $i = h^*$.*

5 Equivalent Transformations of Gringo Programs

In this section we give examples of reasoning about Gringo programs on the basis of the semantics defined above. These examples use the theorem from (Harrison, Lifschitz, and Truszczynski 2014) reproduced in the appendix.

5.1 Simplifying a Rule from Example 3.7 of User's Guide

Consider the rule[7]

$$\mathit{weekdays} \leftarrow \mathit{day}(X) : (\mathit{day}(X), \mathit{not\ weekend}(X)). \quad (15)$$

Replacing this rule with the fact *weekdays* within any program will not affect the set of stable models. Indeed, the result of applying translation τ to (15) is the formula

$$\bigwedge_r (\mathit{day}(r) \wedge \neg \mathit{weekend}(r) \rightarrow \mathit{day}(r)) \rightarrow \mathit{weekdays}, \quad (16)$$

where the conjunction extends over all precomputed terms r. The formula

$$\mathit{day}(r) \wedge \neg \mathit{weekend}(r) \rightarrow \mathit{day}(r)$$

is intuitionistically provable. By the theorem from Section A.2, it follows that replacing (16) with the atom *weekdays* within any set of formulas does not affect the set of stable models.

5.2 Simplifying the Sorting Rule

The rule

$$\begin{aligned} \mathit{order}(X,Y) \leftarrow\ & p(X),\ p(Y),\ X<Y, \\ & \mathit{not}\ p(Z) : (p(Z), X<Z, Z<Y) \end{aligned} \quad (17)$$

can be used for sorting.[8] It can be replaced by either of the following two shorter rules within any program without changing that program's stable models.

$$\begin{aligned} \mathit{order}(X,Y) \leftarrow\ & p(X),\ p(Y),\ X<Y, \\ & \bot : (p(Z), X<Z, Z<Y) \end{aligned} \quad (18)$$

$$\begin{aligned} \mathit{order}(X,Y) \leftarrow\ & p(X),\ p(Y),\ X<Y, \\ & \mathit{not}\ p(Z) : (X<Z, Z<Y) \end{aligned} \quad (19)$$

Let's prove this claim for rule (18). By the theorem from Section A.2, it is sufficient to show that the result of applying τ to (17) is equivalent in the extended system to the result of applying τ to (18). The instances of (17) are the rules

$$\begin{aligned} \mathit{order}(i,j) \leftarrow\ & p(i),\ p(j),\ i<j, \\ & \mathit{not}\ p(Z) : (p(Z), i<Z, Z<j), \end{aligned}$$

and the instances of (18) are the rules

$$\begin{aligned} \mathit{order}(i,j) \leftarrow\ & p(i),\ p(j),\ i<j, \\ & \bot : (p(Z), i<Z, Z<j) \end{aligned}$$

where i and j are arbitrary numerals. The result of applying τ to (17) is the conjunction of the formulas

$$\begin{aligned} & p(i) \wedge p(j) \wedge \tau(i<j) \wedge \\ & \bigwedge_k (\neg p(k) \wedge \tau(i<k) \wedge \tau(k<j) \rightarrow p(k)) \rightarrow \\ & \qquad \mathit{order}(i,j) \end{aligned} \quad (20)$$

for all numerals $i,\ j$. The result of applying τ to (18) is the conjunction of the formulas

$$\begin{aligned} & p(i) \wedge p(j) \wedge \tau(i<j) \wedge \\ & \bigwedge_k (\neg p(k) \wedge \tau(i<k) \wedge \tau(k<j) \rightarrow \bot) \rightarrow \\ & \qquad \mathit{order}(i,j). \end{aligned} \quad (21)$$

[7] This rule is similar to a rule from Example 3.7 of the Gringo 3 *User's Guide* (see Footnote 2).

[8] This rule was communicated to us by Roland Kaminski on October 21, 2012.

It remains to observe that for any numerals i, j, k,

$$p(k) \wedge \tau(i<k) \wedge \tau(k<j) \rightarrow \neg p(k)$$

is intuitionistically equivalent to

$$p(k) \wedge \tau(i<k) \wedge \tau(k<j) \rightarrow \bot.$$

The proof for rule (19) is similar. Rule (18), like rule (17), is safe; rule (19) is not.

5.3 Eliminating Choice in Favor of a Conditional Literal

Replacing the rule

$$\{p(X)\} \leftarrow q(X) \quad (22)$$

with

$$p(X) \leftarrow q(X), \bot : \mathit{not}\ p(X) \quad (23)$$

within any program will not affect the set of stable models. Indeed, the result of applying translation τ to (22) is

$$\bigwedge_r (q(r) \rightarrow p(r) \vee \neg p(r)) \quad (24)$$

where the conjunction extends over all precomputed terms r, and the result of applying τ to (23) is

$$\bigwedge_r (q(r) \wedge \neg\neg p(r) \rightarrow p(r)). \quad (25)$$

Each conjunctive term of (24) is equivalent to the corresponding conjunctive term of (25) in the extension of intuitionistic logic obtained by adding the axiom schema

$$\neg F \vee \neg\neg F,$$

and consequently in the extended system presented in the appendix. It follows that (24) is equivalent to (25) in the extended system as well.

5.4 Eliminating a Trivial Aggregate Expression

Rule (1) says, informally speaking, that we can conclude $p(Y)$ once we establish that there exists at least one X such that $q(X,Y)$. Replacing this rule with

$$p(Y) \leftarrow q(X,Y) \quad (26)$$

within any program will not affect the set of stable models.

To prove this claim, we need to calculate the result of applying τ to rule (1). The instances of (1) are the rules

$$p(t) \leftarrow \mathit{count}\{X, t : q(X,t)\} \geq 1 \quad (27)$$

for all precomputed terms t. Consider the aggregate expression E in the body of (27). Any precomputed term r is admissible w.r.t. E. A set Δ of precomputed terms justifies E if

$$\widehat{\mathit{count}}(\{(r,t) : r \in \Delta\}) \geq 1,$$

that is to say, if Δ is non-empty. Consequently τE consists of only one implication (6), with the empty Δ. The antecedent of this implication is the empty conjunction $\top$,

and its consequent is the disjunction $\bigvee_u q(u,t)$ over all precomputed terms u. Then the result of applying τ to (1) is

$$\bigwedge_t \left(\bigvee_u q(u,t) \ \rightarrow \ p(t) \right). \tag{28}$$

On the other hand, the result of applying τ to (26) is

$$\bigwedge_{t,u} (q(u,t) \rightarrow p(t)).$$

This formula is equivalent to (28) in the extended system defined in the appendix (see Example 2 from (Harrison, Lifschitz, and Truszczynski 2014)).

6 Monotone and Anti-Monotone Aggregate Expressions

A closed aggregate expression E is *monotone* if for any set Δ that justifies E, all supersets of Δ also justify E. Similarly, E is *anti-monotone* if for any set Δ that justifies E, all subsets of Δ also justify E.

For example, if α is one of the symbols *count*, *max*, or *sum*, then (3) is monotone when $\prec$ is $>$ or $\geq$, and anti-monotone when $\prec$ is $<$ or $\leq$. (According to the definition for $\widehat{sum}$ given in this paper negative summands are disregarded.) As observed in Section 3.5, if $\prec$ is $=$ then E can be equivalently replaced by the pair of aggregate expressions $E_{\leq}, E_{\geq}$, which are anti-monotone and monotone respectively when α is one of the symbols listed above.

6.1 Simplifying Monotone and Anti-Monotone Aggregate Expressions

Recall that τE for a closed aggregate expression E is defined as the conjunction of the implications (6) over all sets Δ that do not justify E. The two theorems stated below show that the antecedent in (6) can be dropped if E is monotone, and the consequent can be dropped if E is anti-monotone. These equivalences can be proved in the extended system described in the appendix. The theorems are similar to the properties of monotone and anti-monotone ground aggregates stated in (Ferraris 2005).

Theorem 1 *If a closed aggregate expression E is monotone, then τE is equivalent in the extended system to*

$$\bigwedge_{\Delta} \bigvee_{(i,\mathbf{r}) \in A \setminus \Delta} \tau((\mathbf{L}_i)^{\mathbf{x}_i}_{\mathbf{r}}), \tag{29}$$

where the conjunction extends over the subsets Δ of A that do not justify E.

Theorem 2 *If a closed aggregate expression E is anti-monotone, then τE is equivalent in the extended system to*

$$\bigwedge_{\Delta} \neg \bigwedge_{(i,\mathbf{r}) \in \Delta} \tau((\mathbf{L}_i)^{\mathbf{x}_i}_{\mathbf{r}}), \tag{30}$$

where the conjunction extends over all subsets Δ of A that do not justify E.

Example We saw in Section 3.5 that the result of applying τ to (8) is the conjunction of formulas (9) over all sets Δ of pairs (h,c) of precomputed terms that do not justify (8). Theorems 1 and 2 allow us to simplify this conjunction. Note first that the result of applying τ to (8) can be rewritten as the conjunction of

$$\tau(sum\{H, C : enroll(C), hours(H,C)\} \leq 6) \tag{31}$$

and

$$\tau(sum\{H, C : enroll(C), hours(H,C)\} \geq 6). \tag{32}$$

By Theorem 1, (32) is equivalent to

$$\bigwedge_{\widehat{sum}(\Delta) < 6} \ \bigvee_{(h,c) \notin \Delta} (enroll(c) \wedge hours(h,c)),$$

and by Theorem 2, (31) is equivalent to

$$\bigwedge_{\widehat{sum}(\Delta) > 6} \ \neg \bigwedge_{(h,c) \in \Delta} (enroll(c) \wedge hours(h,c)),$$

where Δ ranges over pairs (h,c) of precomputed terms.

In Section 6.2 we show how Theorem 2 can be used for simplifying and aggregate expression. Proofs of Theorems 1 and 2 are given in Section 6.3.

6.2 Eliminating an Aggregate Expression in Favor of a Conditional Literal

We will show that the rule

$$q \leftarrow count\{X : p(X)\} < 1 \tag{33}$$

can be replaced by the rule

$$q \leftarrow \bot : p(X) \tag{34}$$

within any program without changing its stable models. The bodies of both rules express that the set p is empty, but the first uses an aggregate expression and the second a conditional literal.

Since the aggregate expression in the body of (33) is anti-monotone, by Theorem 2, the result of applying τ to it is equivalent (in the extended system) to

$$\bigwedge_{\Delta \neq \emptyset} \neg \bigwedge_{a \in \Delta} p(a) \tag{35}$$

where Δ ranges over arbitrary sets of precomputed terms.

On the other hand, the result of applying τ to the body of (34) is

$$\bigwedge_{a} \neg p(a) \tag{36}$$

where a ranges over all precomputed terms.

It remains to observe that all conjunctive terms of (35) can be derived in the extended system from the conjunctive terms in which Δ is a singleton, and that the conjunctive terms in which Δ is a singleton are identical to the conjunctive terms of (36).

6.3 Proofs of Theorems 1 and 2

Proof of Theorem 1 The implication from (29) to τE is obvious. Now, assume τE, and consider the conjunctive term of (29) corresponding to a set Δ_0 which does not justify E. Since E is monotone, from τE we can derive the conjunction of all terms (6) where $\Delta \subseteq \Delta_0$. Any formula of the form

$$\left(F \to \bigvee_{i \in I} G_i\right) \to \left(\bigvee_{i \in I} (F \to G_i)\right)$$

is provable in the extended system defined in the appendix (see Section 7 of (Harrison, Lifschitz, and Truszczynski 2014)). Consequently, we can derive the conjunction of the disjunctions

$$\bigvee_{(i,\mathbf{r}) \in A \setminus \Delta} \left(\left(\bigwedge_{(j,\mathbf{s}) \in \Delta} \tau((\mathbf{L}_j)_{\mathbf{s}}^{\mathbf{x}_j})\right) \to \tau((\mathbf{L}_i)_{\mathbf{r}}^{\mathbf{x}_i})\right)$$

over all $\Delta \subseteq \Delta_0$. By distributivity, this is equivalent in the extended system to the disjunction of the conjunctions

$$\bigwedge_{\Delta \subseteq \Delta_0} \left(\bigwedge_{(j,\mathbf{s}) \in \Delta} \tau((\mathbf{L}_j)_{\mathbf{s}}^{\mathbf{x}_j}) \to \tau\left((\mathbf{L}_{f(\Delta)})_{g(\Delta)}^{\mathbf{x}_{f(\Delta)}}\right)\right) \quad (37)$$

over all pairs of functions (f, g) defined on all subsets Δ of Δ_0 such that the pair $(f(\Delta), g(\Delta))$ is an element of $A \setminus \Delta$. Now we reason by cases, with one case corresponding to each disjunctive term (37). For a particular disjunctive term (37), we wish to show that we can derive from it $\tau((\mathbf{L}_i)_{\mathbf{r}}^{\mathbf{x}_i})$ for some pair $(i, \mathbf{r})$ in $A \setminus \Delta_0$. Consider the set Δ^* of all pairs $(i, \mathbf{r})$ such that $\tau((\mathbf{L}_i)_{\mathbf{r}}^{\mathbf{x}_i})$ is derivable from (37) in the extended system. We will show that Δ^* is not a subset of Δ_0. Assume $\Delta^* \subseteq \Delta_0$, so that

$$(f(\Delta^*), g(\Delta^*)) \in A \setminus \Delta^*, \quad (38)$$

and one of the conjunctive terms of (37) is

$$\bigwedge_{(j,\mathbf{s}) \in \Delta^*} \tau((\mathbf{L}_j)_{\mathbf{s}}^{\mathbf{x}_j}) \to \tau\left((\mathbf{L}_{f_{\Delta^*}})_{g(\Delta^*)}^{\mathbf{x}_{f(\Delta^*)}}\right).$$

By the definition of Δ^*, every conjunctive term in the antecedent of this implication is derivable from (37). It follows that the consequent is derivable as well, so that $(f(\Delta^*), g(\Delta^*))$ belongs to Δ^*, which contradicts (38). Therefore, Δ^* is not a subset of Δ_0, so that at least one disjunctive term of (29) is derivable from (37).

Proof of Theorem 2 The implication from (30) to τE is obvious. We will derive (30) from τE using the disjunction

$$\bigvee_{\Delta \subseteq A} \left(\neg \bigvee_{(i,\mathbf{r}) \in A \setminus \Delta} \tau((\mathbf{L}_i)_{\mathbf{r}}^{\mathbf{x}_i}) \wedge \neg\neg \bigwedge_{(i,\mathbf{r}) \in \Delta} \tau((\mathbf{L}_i)_{\mathbf{r}}^{\mathbf{x}_i})\right). \quad (39)$$

This is a special case of the generalized law of weak excluded middle given in Section 7 of (Harrison, Lifschitz, and Truszczynski 2014). Each disjunctive term (39) corresponds to a subset Δ of A. We will reason by cases with one case corresponding to each disjunctive term D_Δ of (39). Each D_Δ is equivalent to

$$\neg\left(\bigwedge_{(i,\mathbf{r}) \in \Delta} \tau((\mathbf{L}_i)_{\mathbf{r}}^{\mathbf{x}_i}) \to \bigvee_{(i,\mathbf{r}) \in A \setminus \Delta} \tau((\mathbf{L}_i)_{\mathbf{r}}^{\mathbf{x}_i})\right) \quad (40)$$

in the extended system described in the appendix. If Δ does not justify E then (40) contradicts one of the conjunctive terms from (6) and (30) follows. Consider now the case when Δ justifies E. Assume

$$\bigwedge_{(i,\mathbf{r}) \in \Delta_0} \tau((\mathbf{L}_i)_{\mathbf{r}}^{\mathbf{x}_i}) \quad (41)$$

for some Δ_0 that does not justify E. Since E is anti-monotone Δ_0 is not a subset of Δ. We conclude that Δ_0 and $A \setminus \Delta$ overlap on at least one element. It follows that

$$\bigvee_{(i,\mathbf{r}) \in A \setminus \Delta} \tau((\mathbf{L}_i)_{\mathbf{r}}^{\mathbf{x}_i})$$

can be derived from (41) since at least one of its disjunctive terms can be derived. This disjunction contradicts (40), and so we may conclude the negation of (41).

7 Conclusion

In this note, we approached the problem of defining the semantics of Gringo by reducing Gringo programs to infinitary propositional formulas. We argued that this approach to semantics may allow us to study equivalent transformations of programs using natural deduction in infinitary propositional logic.

In the absence of a precise semantics, it is impossible to put the study of some important issues on a firm foundation. This includes the correctness of ASP programs, grounders, solvers, and optimization methods, and also the relationship between input languages of different solvers (for instance, the equivalence of the semantics of aggregate expressions in Gringo to their semantics in the ASP Core language and in the language proposed in (Gelfond 2002) under the assumption that aggregates are used nonrecursively). As future work, we are interested in addressing some of these tasks on the basis of the semantics proposed in this note. Proving the correctness of the intelligent instantiation algorithms implemented in GRINGO will provide justification for our informal claim that for a safe program, the semantics proposed here correctly describes the output produced by GRINGO.

Acknowledgements

Many thanks to Roland Kaminski and Torsten Schaub for helping us understand the input language of GRINGO. Roland, Michael Gelfond, Yuliya Lierler, Joohyung Lee, and anonymous referees provided valuable comments on drafts of this note.

Appendix

We review here conditions for characterizing the stable models of a set of infinitary formulas in terms of supportedness

(Lifschitz and Yang 2013), and transformations of infinitary formulas which preserve their stable models (Harrison, Lifschitz, and Truszczynski 2014).

A.1 Tight Infinitary Programs

An *infinitary program* is a conjunction of (possibly infinitely many) infinitary formulas of the form $G \to A$, where A is an atom. We say that an infinitary program Π is *supertight* if it is impossible to find an infinite sequence of atoms $A_1, A_2, \dots$ such that for all i, Π contains a conjunctive term $G \to A_i$, where A_{i+1} occurs in G. We say that an interpretation I is *supported* by Π if each atom $A \in I$ is the consequent of a conjunctive term $G \to A$ of Π such that $I \models G$.

Theorem *For any model I of a supertight infinitary program Π, I is stable iff I is suported by Π.*

This theorem is a special case of Lemma 2 from (Lifschitz and Yang 2013). In the statement of that lemma Π is required to be "tight on I." That condition is significantly more general than supertightness as defined above.

A.2 Equivalent Transformations of Infinitary Formulas

The extended system of natural deduction presented in (Harrison, Lifschitz, and Truszczynski 2014) includes the axiom schema

$$F \Rightarrow F,$$

the axiom schema

$$\Rightarrow F \vee (F \to G) \vee \neg G$$

that characterizes (in the finite case) the logic of here-and-there (Hosoi 1966), and the following additional axiom schemas:

$$\Rightarrow \neg \bigwedge_{F \in \mathcal{H}} F \to \bigvee_{F \in \mathcal{H}} \neg F, \tag{42}$$

$$\Rightarrow \left(\bigwedge_{i \in I} \bigvee_{F \in \mathcal{H}_i} F \right) \to \left(\bigvee_{\{F_i\}_{i \in I}} \bigwedge_{i \in I} F_i \right), \tag{43}$$

and

$$\Rightarrow \left(\bigwedge_{\{F_i\}_{i \in I}} \bigvee_{i \in I} F_i \right) \to \left(\bigvee_{i \in I} \bigwedge_{F \in \mathcal{H}_i} F \right). \tag{44}$$

In formula (42) $\mathcal{H}$ is a bounded set of formulas. In formulas (43) and (44) $\{\mathcal{H}_i\}_{i \in I}$ is an arbitrary non-empty family of sets of formulas such that its union is bounded. The disjunction in the consequent of (43) and the conjunction in the antecedent of (44) extend over all elements $\{F_i\}_{i \in I}$ of the Cartesian product of the family $\{\mathcal{H}_i\}_{i \in I}$.

The inference rules are the introduction and elimination rules for the propositional connectives

$$(\wedge I) \quad \frac{\Gamma \Rightarrow H \quad \text{for all } H \in \mathcal{H}}{\Gamma \Rightarrow \mathcal{H}^\wedge}$$

$$(\wedge E) \quad \frac{\Gamma \Rightarrow \mathcal{H}^\wedge}{\Gamma \Rightarrow H} \quad (H \in \mathcal{H})$$

$$(\vee I) \quad \frac{\Gamma \Rightarrow H}{\Gamma \Rightarrow \mathcal{H}^\vee} \quad (H \in \mathcal{H})$$

$$(\vee E) \quad \frac{\Gamma \Rightarrow \mathcal{H}^\vee \qquad \Delta, H \Rightarrow F \quad \text{for all } H \in \mathcal{H}}{\Gamma, \Delta \Rightarrow F}$$

$$(\to I) \quad \frac{\Gamma, F \Rightarrow G}{\Gamma \Rightarrow F \to G}$$

$$(\to E) \quad \frac{\Gamma \Rightarrow F \quad \Delta \Rightarrow F \to G}{\Gamma, \Delta \Rightarrow G},$$

where $\mathcal{H}$ is a bounded set of formulas, and the weakening rule

$$(W) \ \frac{\Gamma \Rightarrow F}{\Gamma, \Delta \Rightarrow F}.$$

The set of *theorems of the extended system* is the smallest set of sequents that includes the axioms of the system and is closed under the application of its inference rules. We say that formulas F and G are *equivalent in the extended system* if $\Rightarrow F \leftrightarrow G$ is a theorem of the extended system.

Theorem *For any set $\mathcal{H}$ of formulas, if formulas F and G are equivalent in the extended system then $\mathcal{H} \cup \{F\}$ and $\mathcal{H} \cup \{G\}$ have the same stable models.*

References

Ferraris, P., and Lifschitz, V. 2005. Weight constraints as nested expressions. *Theory and Practice of Logic Programming* 5:45–74.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence* 175:236–263.

Ferraris, P. 2005. Answer sets for propositional theories. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 119–131.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R., and Bowen, K., eds., *Proceedings of International Logic Programming Conference and Symposium*, 1070–1080. MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Gelfond, M. 2002. Representing knowledge in A-Prolog. *Lecture Notes in Computer Science* 2408:413–451.

Harrison, A.; Lifschitz, V.; and Truszczynski, M. 2014. On equivalence of infinitary formulas under the stable model semantics. *Theory and Practice of Logic Programming*. To appear.

Hosoi, T. 1966. The axiomatization of the intermediate propositional systems S_n of Gödel. *Journal of the Faculty of Science of the University of Tokyo* 13:183–187.

Lee, J., and Meng, Y. 2012a. Stable models of formulas with generalized quantifiers. In *Working Notes of the 14th International Workshop on Non-Monotonic Reasoning (NMR)*.

Lee, J., and Meng, Y. 2012b. Stable models of formulas with generalized quantifiers (preliminary report). In *Technical Communications of the 28th International Conference on Logic Programming (ICLP)*, 61–71.

Lee, J., and Meng, Y. 2012c. Two new definitions of stable models of logic programs with generalized quantifiers. In *Working Notes of the 5th Workshop of Answer Set Programming and Other Computing Paradigms (ASPOCP)*.

Lifschitz, V., and Yang, F. 2013. Lloyd-Topor completion and general stable models. *Theory and Practice of Logic Programming* 13(4–5).

Pearce, D., and Valverde, A. 2004. Towards a first order equilibrium logic for nonmonotonic reasoning. In *Proceedings of European Conference on Logics in Artificial Intelligence (JELIA)*, 147–160.

Truszczynski, M. 2012. Connecting first-order ASP and the logic FO(ID) through reducts. In *Correct Reasoning: Essays on Logic-Based AI in Honor of Vladimir Lifschitz*. Springer.

A SCC Recursive Meta-Algorithm for Computing Preferred Labellings in Abstract Argumentation

Federico Cerutti
School of Natural and Computing Science
King's College
University of Aberdeen
AB24 3UE, Aberdeen, United Kingdom
e-mail: f.cerutti@abdn.ac.uk

Massimiliano Giacomin
Department of Information Engineering
University of Brescia
via Branze, 38
25123, Brescia, Italy
e-mail: massimiliano.giacomin@ing.unibs.it

Mauro Vallati
School of Computing and Engineering
University of Huddersfield
Huddersfield, HD1 3DH, United Kingdom
e-mail: m.vallati@hud.ac.uk

Marina Zanella
Department of Information Engineering
University of Brescia
via Branze, 38
25123, Brescia, Italy
e-mail: marina.zanella@ing.unibs.it

Abstract

This paper presents a meta-algorithm for the computation of preferred labellings, based on the general recursive schema for argumentation semantics called SCC-Recursiveness. The idea is to recursively decompose a framework so as to compute semantics labellings on restricted sub-frameworks, in order to reduce the computational effort. The meta-algorithm can be instantiated with a specific "base algorithm", applied to the base case of the recursion, which can be obtained by generalizing existing algorithms in order to compute labellings in restricted sub-frameworks. We devise for this purpose a generalization of a SAT-based algorithm, and provide an empirical investigation to show the significant improvement of performances obtained by exploiting the SCC-recursive schema.

Introduction

Dung's theory of abstract argumentation (Dung 1995) is a unifying framework able to encompass a large variety of specific formalisms in the areas of nonmonotonic reasoning, logic programming and computational argumentation. It is based on the notion of argumentation framework (AF), consisting of a set of arguments and an *attack* relation between them. Different *argumentation semantics* introduce in a declarative way the criteria to determine which arguments emerge as justified from the conflict, by identifying a number of *extensions*, i.e. sets of arguments that can "survive the conflict together". In (Dung 1995) four "traditional" semantics were introduced, namely *complete*, *grounded*, *stable*, and *preferred* semantics. For an introduction on alternative semantics, see (Baroni, Caminada, and Giacomin 2011).

The main computational problems in abstract argumentation include *decision* and *construction* problems, and turn out to be computationally intractable for most of argumentation semantics (Dunne and Wooldridge 2009). In this paper we focus on the *extension enumeration* problem, i.e. constructing *all* extensions prescribed for a given AF: its solution provides complete information concerning the justification status of arguments and subsumes the solutions to the other problems.

On the practical side, few results are available on the development of efficient algorithms for abstract argumentation and their empirical assessment. In particular, in (Cerutti et al. 2013; 2014) a SAT-based approach has been proposed to solve the extension enumeration problem for preferred semantics. Preferred semantics represents the main contribution in Dung's theory, as it allows multiple extensions (differently from grounded semantics), the existence of extensions is always guaranteed (differently from stable semantics), and no extension is a proper subset of another extension (differently from complete semantics). The proposed approach basically performs a search in the space of complete extensions to enumerate the maximal ones, exploiting a SAT solver to identify the relevant search states. In (Cerutti et al. 2014) it has been shown to be competitive w.r.t. other state-of-the-art systems including ASPARTIX (Egly, Alice Gaggl, and Woltran 2010), ASPARTIX-META (Dvořák et al. 2011) and the system presented in (Nofal, Dunne, and Atkinson 2012). A limitation of this approach is that it is always applied to the AF as a whole, without dividing the enumeration problem into a number of simpler sub-problems.

In this paper, we aim at showing (i) an approach for dividing the problem of enumerating the preferred extensions into sub-problems, and (ii) that such an approach reduces the overall computational effort. In particular, we rely on the SCC-recursive schema, first introduced in (Baroni, Giacomin, and Guida 2005), which is a semantics definition schema where extensions are defined at the level of the

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

sub-frameworks identified by the strongly connected components. It is worth to mention that the focus of this paper is on preferred semantics, but the devised meta-algorithm can be instantiated with various specific "base algorithms" and applied to a variety of semantics.

The paper is organised as follows. After recallling some necessary background in the first section, the SCC-recursive schema is reviewed in the next one. Then the parametric meta-algorithm is presented, and the generalization of the SAT-based algorithm of (Cerutti et al. 2014) is described. Next section describes the test setting and comments the experimental results, while the last section provides a final discussion and concludes the paper. Proofs are omitted or sketched due to space limitations.

Background

An argumentation framework (Dung 1995) consists of a set of arguments[1] and a binary attack relation between them.

Definition 1. *An* argumentation framework *(AF) is a pair $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ where $\mathcal{A}$ is a set of arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$. We say that $\boldsymbol{b}$* attacks *$\boldsymbol{a}$ iff $\langle \boldsymbol{b}, \boldsymbol{a} \rangle \in \mathcal{R}$, also denoted as $\boldsymbol{b} \rightarrow \boldsymbol{a}$. The set of attackers of an argument $\boldsymbol{a}$ will be denoted as $\boldsymbol{a}^- \triangleq \{\boldsymbol{b} : \boldsymbol{b} \rightarrow \boldsymbol{a}\}$, the set of arguments attacked by $\boldsymbol{a}$ will be denoted as $\boldsymbol{a}^+ \triangleq \{\boldsymbol{b} : \boldsymbol{a} \rightarrow \boldsymbol{b}\}$. We also extend these notations to sets of arguments, i.e. given $E \subseteq \mathcal{A}$, $E^- \triangleq \{\boldsymbol{b} \mid \exists \boldsymbol{a} \in E, \boldsymbol{b} \rightarrow \boldsymbol{a}\}$ and $E^+ \triangleq \{\boldsymbol{b} \mid \exists \boldsymbol{a} \in E, \boldsymbol{a} \rightarrow \boldsymbol{b}\}$.*

An argument $\mathbf{a}$ without attackers, i.e. such that $\mathbf{a}^- = \emptyset$, is said *initial*. Moreover, each argumentation framework has an associated directed graph where the vertices are the arguments, and the edges are the attacks.

The basic properties of conflict–freeness, acceptability, and admissibility of a set of arguments are fundamental for the definition of argumentation semantics.

Definition 2. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$:*

- *a set $S \subseteq \mathcal{A}$ is a* conflict–free *set of Γ if $\nexists\, \boldsymbol{a}, \boldsymbol{b} \in S$ s.t. $\boldsymbol{a} \rightarrow \boldsymbol{b}$;*
- *an argument $\boldsymbol{a} \in \mathcal{A}$ is* acceptable *with respect to a set $S \subseteq \mathcal{A}$ of Γ if $\forall \boldsymbol{b} \in \mathcal{A}$ s.t. $\boldsymbol{b} \rightarrow \boldsymbol{a}$, $\exists\, \boldsymbol{c} \in S$ s.t. $\boldsymbol{c} \rightarrow \boldsymbol{b}$;*
- *a set $S \subseteq \mathcal{A}$ is an* admissible set *of Γ if S is a conflict–free set of Γ and every element of S is acceptable with respect to S of Γ.*

An argumentation semantics σ prescribes for any AF Γ a set of *extensions*, denoted as $\mathcal{E}_\sigma(\Gamma)$, namely a set of sets of arguments satisfying the conditions dictated by σ. Here we need to recall the definitions of complete (denoted as $\mathcal{CO}$), grounded (denoted as $\mathcal{GR}$) and preferred (denoted as $\mathcal{PR}$) semantics only.

Definition 3. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$:*

- *a set $S \subseteq \mathcal{A}$ is a* complete extension *of Γ, i.e. $S \in \mathcal{E}_{\mathcal{CO}}(\Gamma)$, iff S is admissible and $\forall \boldsymbol{a} \in \mathcal{A}$ s.t. $\boldsymbol{a}$ is acceptable w.r.t. S, $\boldsymbol{a} \in S$;*
- *a set $S \subseteq \mathcal{A}$ is the* grounded extension *of Γ, i.e. $S \in \mathcal{E}_{\mathcal{GR}}(\Gamma)$, iff S is the minimal (w.r.t. set inclusion) complete extension of Γ. Its existence and uniqueness have been proved in (Dung, Mancarella, and Toni 2006);*
- *a set $S \subseteq \mathcal{A}$ is a* preferred extension *of Γ, i.e. $S \in \mathcal{E}_{\mathcal{PR}}(\Gamma)$, iff S is a maximal (w.r.t. set inclusion) complete extension of Γ.*

It can be noted that each extension S implicitly defines a three-valued *labelling* of arguments: an argument $\mathbf{a}$ is labelled `in` iff $\mathbf{a} \in S$; is labelled `out` iff $\exists\, \mathbf{b} \in S$ s.t. $\mathbf{b} \rightarrow \mathbf{a}$; is labelled `undec` if neither of the above conditions holds. In the light of this correspondence, argumentation semantics can be equivalently defined in terms of labellings rather than of extensions (Caminada 2006; Baroni, Caminada, and Giacomin 2011). For technical reasons, we introduce the notion of labelling both for argumentation frameworks and for arbitrary sets of arguments.

Definition 4. *Given a set of arguments S, a* labelling *of S is a total function $\mathcal{L}ab : S \longrightarrow \{\mathtt{in}, \mathtt{out}, \mathtt{undec}\}$. The set of all* labellings *of S is denoted as $\mathfrak{L}_S$. Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, a* labelling *of Γ is a labelling of $\mathcal{A}$. The set of all* labellings *of Γ is denoted as $\mathfrak{L}(\Gamma)$.*

In particular, complete labellings can be defined as follows.

Definition 5. *Let $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework. A labelling $\mathcal{L}ab \in \mathfrak{L}(\Gamma)$ is a* complete labelling *of Γ iff it satisfies the following conditions for any $\boldsymbol{a} \in \mathcal{A}$:*

- *$\mathcal{L}ab(\boldsymbol{a}) = \mathtt{in} \Leftrightarrow \forall \boldsymbol{b} \in \boldsymbol{a}^- \mathcal{L}ab(\boldsymbol{b}) = \mathtt{out}$;*
- *$\mathcal{L}ab(\boldsymbol{a}) = \mathtt{out} \Leftrightarrow \exists \boldsymbol{b} \in \boldsymbol{a}^- : \mathcal{L}ab(\boldsymbol{b}) = \mathtt{in}$;*

The grounded and preferred labelling can then be defined on the basis of complete labellings.

Definition 6. *Let $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework. A labelling $\mathcal{L}ab \in \mathfrak{L}(\Gamma)$ is* the grounded labelling *of Γ if it is the complete labelling of Γ minimizing the set of arguments labelled* `in`*, and it is a* preferred labelling *of Γ if it is a complete labelling of Γ maximizing the set of arguments labelled* `in`*.*

In order to show the connection between extensions and labellings, let us recall the definition of the function `Ext2Lab`, returning the labelling corresponding to a conflict–free set of arguments S.

Definition 7. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a conflict–free set $S \subseteq \mathcal{A}$, the corresponding labelling $\mathtt{Ext2Lab}(S)$ is defined as $\mathtt{Ext2Lab}(S) \equiv \mathcal{L}ab$, where*

- *$\mathcal{L}ab(\boldsymbol{a}) = \mathtt{in} \Leftrightarrow \boldsymbol{a} \in S$*
- *$\mathcal{L}ab(\boldsymbol{a}) = \mathtt{out} \Leftrightarrow \exists\, \boldsymbol{b} \in S$ s.t. $\boldsymbol{b} \rightarrow \boldsymbol{a}$*
- *$\mathcal{L}ab(\boldsymbol{a}) = \mathtt{undec} \Leftrightarrow \boldsymbol{a} \notin S \wedge \nexists\, \boldsymbol{b} \in S$ s.t. $\boldsymbol{b} \rightarrow \boldsymbol{a}$*

(Caminada 2006) shows that there is a bijective correspondence between the complete, grounded, preferred extensions and the complete, grounded, preferred labellings, respectively.

Proposition 1. *Given an an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, $\mathcal{L}ab$ is a complete (grounded, preferred) labelling of Γ if and only if there is a complete (grounded, preferred) extension S of Γ such that $\mathcal{L}ab = \mathtt{Ext2Lab}(S)$.*

[1] In this paper we consider only *finite* sets of arguments: see (Baroni et al. 2013) for a discussion on infinite sets of arguments.

The set of complete labellings of Γ is denoted as $\mathcal{L}_{\mathcal{CO}}(\Gamma)$, the set of preferred labellings as $\mathcal{L}_{\mathcal{PR}}(\Gamma)$, while $\mathcal{L}_{\mathcal{GR}}(\Gamma)$ denotes the set including the grounded labelling.

SCC-Recursiveness Revisited

In (Baroni and Giacomin 2004) it has been recognised that each argumentation framework can be partitioned into a set of sub-frameworks in such a way that most common argumentation semantics (originally defined at the global level) can be equivalently defined at the level of these sub-frameworks. In particular, an extension-based semantics definition schema has been introduced, called *SCC (strongly connected component)-recursiveness*, based on the graph-theoretical notion of SCCs i.e. the equivalence classes induced by the path equivalence (i.e. mutual reachability) relation between the nodes of the associated graph of an argumentation framework (given the direct correspondence between each AF and its associated graph, in the following we will equivalently refer to both of them as AF).

Definition 8. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, the binary relation of* path-equivalence *between nodes, denoted as $PE_\Gamma \subseteq \mathcal{A} \times \mathcal{A}$, is defined as follows:*

- $\forall \boldsymbol{a} \in \mathcal{A}, \langle \boldsymbol{a}, \boldsymbol{a} \rangle \in PE_\Gamma$;
- $\forall \boldsymbol{a}, \boldsymbol{b} \in \mathcal{A}$, $\boldsymbol{a} \neq \boldsymbol{b}$, $\langle \boldsymbol{a}, \boldsymbol{b} \rangle \in PE_\Gamma$ *iff there is a path from* $\boldsymbol{a}$ *to* $\boldsymbol{b}$ *and vice versa.*

Two main features of the SCC-recursive schema are worth remarking.

First, the SCC-recursive schema exploits the partial order of SCCs induced by the attack relation and can be regarded as a constructive procedure to incrementally build extensions following such partial order. At the beginning, the extensions of the frameworks restricted to the *initial* SCCs (i.e. those not receiving attacks from others) are computed and combined together. Then each SCC which is attacked only from initial SCCs is considered, and for each extension E already obtained, the extensions of such a SCC are locally computed and merged with E. The process is then applied to all SCCs following their partial order, until no remaining SCCs are left to process.

A second feature is that the schema, as the name suggests, is recursive. In particular, for every SCC considered in the above procedure, the local computation is performed as follows. First, all arguments attacked by the extension selected in the previous SCCs are suppressed. Then, the procedure is recursively applied to the remaining part of the SCC. The base of the recursion is reached when there is one SCC only: in this case, a base function $\mathcal{BF}$ is called. Such a function receives as input an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$: Γ is a sub-framework of the original AF $\Gamma' = \langle \mathcal{A}', \mathcal{R}' \rangle$, and C contains the arguments that are "externally accepted" (i.e. their attackers in $\mathcal{A}' \setminus \mathcal{A}$ are attacked by the part of the extension constructed so far).

The following definitions introduce the SCC-recursive schema in its original extension-based form (Baroni, Giacomin, and Guida 2005).

First, let us recall the definition of *restriction* of an AF Γ to a set of arguments I, in symbol $\Gamma{\downarrow_I}$.

Definition 9. *Given an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $I \subseteq \mathcal{A}$, the restriction of Γ to I is defined as $\Gamma{\downarrow_I} \equiv (\mathcal{A} \cap I, \mathcal{R} \cap (I \times I))$.*

Then, Definition 10 introduces the function $\mathcal{GF}(\Gamma, C)$ which recursively computes the semantics extensions on the basis of the SCCs of Γ. Let us denote as SCC_Γ the set including the SCCs of an argumentation framework Γ.

Definition 10. *A given argumentation semantics σ is SCC-recursive if for any argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$, $\mathcal{E}_\sigma(\Gamma) = \mathcal{GF}(\Gamma, \mathcal{A}) \subseteq 2^{\mathcal{A}}$. For any $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and for any set $C \subseteq \mathcal{A}$, $E \in \mathcal{GF}(\Gamma, C)$ if and only if*

- $E \in \mathcal{BF}_\sigma(\Gamma, C)$ *if* $|SCC_\Gamma| = 1$
- $\forall S \in SCC_\Gamma \ (E \cap S) \in \mathcal{GF}(\Gamma{\downarrow_{S \setminus (E \setminus S)^+}}, U_\Gamma(S, E) \cap C)$ *otherwise*

where

- $\mathcal{BF}_\sigma(\Gamma, C)$ *is a function, called* base function*, that, given an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ such that $|SCC_\Gamma| = 1$ and a set $C \subseteq \mathcal{A}$, gives a subset of $2^{\mathcal{A}}$*
- $U_\Gamma(S, E) = \{\boldsymbol{a} \in S \setminus (E \setminus S)^+ \mid \forall \boldsymbol{b} \in (\boldsymbol{a}^- \setminus S), \boldsymbol{b} \in E^+\}$

As shown below, this schema is based on the notions of admissible set, complete, grounded, preferred extension of an AF *in a set of arguments*.

Definition 11. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, a set $E \subseteq \mathcal{A}$ is:*

- *an* admissible set of Γ in C *if and only if E is an admissible set of Γ and $E \subseteq C$*
- *a* complete extension of Γ in C *if and only if E is an admissible set of Γ in C, and every argument $\alpha \in C$ which is acceptable with respect to E belongs to E*
- *the* grounded extension of Γ in C *if and only if it is the least (with respect to set inclusion) complete extension of Γ in C*
- *a* preferred extension of Γ in C *if and only if it is a maximal (with respect to set inclusion) complete extension of Γ in C.*

The existence and uniqueness of the grounded extension in C, as well as the existence of at least a preferred extension in C, have been proved in (Baroni, Giacomin, and Guida 2005). The set of admissible sets in C is denoted as $\mathcal{E}_{\mathcal{AD}}(\Gamma, C)$, the set of complete extensions in C as $\mathcal{E}_{\mathcal{CO}}(\Gamma, C)$, the set of preferred extensions in C as $\mathcal{E}_{\mathcal{PR}}(\Gamma, C)$, while $\mathcal{E}_{\mathcal{GR}}(\Gamma, C)$ denotes the set including the grounded extension in C.

(Baroni, Giacomin, and Guida 2005) proves that $\mathcal{GF}(\Gamma, C)$, as defined in Def. 10, returns $\mathcal{E}_\sigma(\Gamma, C)$ (with $\sigma \in \{\mathcal{CO}, \mathcal{GR}, \mathcal{PR}\}$), provided that $\mathcal{BF}_\sigma(\Gamma, C)$ returns the complete, grounded, and preferred extensions in C, respectively. The correctness of the schema then follows from the fact that, according to Definition 11, $\mathcal{E}_\sigma(\Gamma, \mathcal{A}) = \mathcal{E}_\sigma(\Gamma)$.

A Parametric Meta-Algorithm to Enumerate Preferred Labellings

In this section, we develop a "meta-algorithm" based on the SCC-recursive schema for the computation of preferred labellings. We use the term meta-algorithm since it is parametric w.r.t. an algorithm B-PR to compute the base function

Algorithm 1 Enumerating the preferred labellings of an AF

PREF(Γ)

1: **Input:** $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$
2: **Output:** $E_p \in 2^{\mathfrak{L}(\Gamma)}$
3: **return R-PREF**$(\Gamma, \mathcal{A})$

$\mathcal{BF}_{\mathcal{PR}}$, without committing on the way B-PR performs its computation.

The meta-algorithm consists of Algorithms 1 and 2. It represents an implementation of the SCC-recursive schema with two main improvements. First, the schema has been adapted to the labelling-based approach: the following definition introduces the labelling-based counterpart of the complete, grounded and preferred extension in C.

Definition 12. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, a labelling $\mathcal{L}ab \in \mathfrak{L}(\Gamma)$ is a* complete (grounded, preferred) labelling of Γ in C *if there is a complete (grounded, preferred) extension S of Γ in C such that $\mathcal{L}ab = \texttt{Ext2Lab}(S)$. The set of complete and preferred labellings of Γ in C are denoted as $\mathcal{L}_{\mathcal{CO}}(\Gamma, C)$ and $\mathcal{L}_{\mathcal{PR}}(\Gamma, C)$, respectively, and the set including the grounded labelling of Γ in C is denoted as $\mathcal{L}_{\mathcal{GR}}(\Gamma, C)$.*

Finally, a pre-processing step computing the grounded labelling $\mathcal{L}ab^*$ in C is performed. The labellings of the whole argumentation framework can then be obtained by extending $\mathcal{L}ab^*$, on the basis to the following result.

Proposition 2. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and let $C \subseteq \mathcal{A}$ a set of arguments. Considering the grounded labelling $\mathcal{L}ab^*$ of Γ in C and the set U including the* undec*-labelled arguments according to $\mathcal{L}ab^*$, it holds that $\mathcal{L}_{\mathcal{PR}}(\Gamma, C) = \{\mathcal{L}ab^* \cup E \mid E \in \mathcal{L}_{\mathcal{PR}}(\Gamma{\downarrow_U}, C \cap U)\}$.*

Sketch of proof. This extends the known result (Baroni, Caminada, and Giacomin 2011) that the grounded extension is a subset of the intersections of preferred extensions. □

Let us turn to the description of the meta-algorithm. The function PREF (Algorithm 1) receives as input an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and returns the set of preferred labellings of Γ. This is simply achieved by invoking (at line 3) R-PREF$(\Gamma, \mathcal{A})$, where the function R-PREF ($\mathcal{GF}$ in Def. 10) receives as input an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, and computes the set $\mathcal{L}_{\mathcal{PR}}(\Gamma, C)$, i.e. the set of preferred labellings of Γ in C.

Algorithm 2 implements the function R-PREF. First, a pre-processing step which computes the grounded labelling in C is executed at line 3 by means of the call $(\mathcal{L}ab, U) =$ GROUNDED(Γ, C).

The procedure GROUNDED (Alg. 3) receives as input an argumentation framework $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, and returns $\mathcal{L}ab$ and U: $\mathcal{L}ab$ is the restriction of the grounded labelling to the arguments that are `in` or `out`-labelled; U is the set including the remaining (`undec`-labelled) arguments. The GROUNDED procedure iteratively labels `in` each argument $\mathbf{a} \in C$ which either is initial, or receives attacks from arguments labelled `out`, and then labels `out` each argument attacked by $\mathbf{a}$.

Algorithm 2 Enumerating the preferred labellings in a set C of an AF

R-PREF(Γ, C)

1: **Input:** $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle, C \subseteq \mathcal{A}$
2: **Output:** $E_p \in 2^{\mathfrak{L}(\Gamma)}$
3: $(\mathcal{L}ab, U) =$ GROUNDED(Γ, C)
4: $E_p := \{\mathcal{L}ab\}$
5: $\Gamma = \Gamma{\downarrow_U}$
6: $(S_1, \ldots, S_n) :=$ SCCSSEQ(Γ)
7: **for** $i \in \{1, \ldots, n\}$ **do**
8: $\quad E'_p := \emptyset$
9: $\quad$ **for** $\mathcal{L}ab \in E_p$ **do**
10: $\quad\quad (O, I) :=$ BOUNDCOND$(\Gamma, S_i, \mathcal{L}ab)$
11: $\quad\quad$ **if** $I = \emptyset$ **then**
12: $\quad\quad\quad \mathcal{L}ab = \mathcal{L}ab \cup \{(\mathbf{a}, \texttt{out}) \mid \mathbf{a} \in O\} \cup \{(\mathbf{a}, \texttt{undec}) \mid \mathbf{a} \in S_i \setminus O\}$
13: $\quad\quad\quad E'_p = E'_p \cup \{\mathcal{L}ab\}$
14: $\quad\quad$ **else**
15: $\quad\quad\quad$ **if** $O = \emptyset$ **then**
16: $\quad\quad\quad\quad E_* =$ B-PR$(\Gamma{\downarrow_{S_i}}, I \cap C)$
17: $\quad\quad\quad$ **else**
18: $\quad\quad\quad\quad \mathcal{L}ab = \mathcal{L}ab \cup \{(\mathbf{a}, \texttt{out}) \mid \mathbf{a} \in O\}$
19: $\quad\quad\quad\quad E_* =$ R-PREF$(\Gamma{\downarrow_{S_i \setminus O}}, I \cap C)$
20: $\quad\quad\quad$ **end if**
21: $\quad\quad\quad E'_p = E'_p \cup (\mathcal{L}ab \otimes E_*)$
22: $\quad\quad$ **end if**
23: $\quad$ **end for**
24: $\quad E_p := E'_p$
25: **end for**
26: **return** E_p

After the computation of $\mathcal{L}ab$ and U, Alg. 2 initialises the variable E_p to $\{\mathcal{L}ab\}$ at line 4. E_p stores the set of labellings that are incrementally constructed. Then the computation proceeds by considering the argumentation framework $\Gamma{\downarrow_U}$ (line 5).

At line 6 the strongly connected components of the argumentation framework are identified. We assume that an algorithm is available, denoted as SCCSSEQ, which receives as input an argumentation framework Γ and returns as output a sequence $(S_1, \ldots, S_n)$ including the strongly connected components of Γ in a topological order, i.e. if $\exists \mathbf{a} \in S_i, \mathbf{b} \in S_j$ such that $\mathbf{a} \rightarrow \mathbf{b}$ then $i \leq j$. This can be done in linear time under the number of attacks (Cormen et al. 2009, p. 617).

The incremental construction of the preferred labellings is performed by the outer loop (lines 7-25), which iteratively selects the i-th SCC. At the beginning of the iteration, E_p includes the partial labellings constructed for $S_1 \cup \ldots \cup S_{i-1}$.Then the inner loop is entered, which extends any labelling $\mathcal{L}ab \in E_p$ to S_i (lines 9-23) storing the labellings thus obtained in E'_p (initialised at line 8). At line 10 variable O is set to include arguments of S_i that are attacked by "outside" `in`-labelled arguments according to $\mathcal{L}ab$, and variable I is set to include arguments of S_i that are only attacked by "outside" `out`-labelled arguments. Formally, BOUNDCOND$(\Gamma, S_i, \mathcal{L}ab)$ returns (O, I) where

Algorithm 3 Determining the grounded labelling of an AF in a set C

GROUNDED(Γ, C)

1: **Input:** $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle, C \subseteq \mathcal{A}$
2: **Output:** $(\mathcal{L}ab, U) : U \subseteq \mathcal{A}, \mathcal{L}ab \in \mathfrak{L}_{\mathcal{A} \setminus U}$
3: $\mathcal{L}ab := \emptyset$
4: $U := \mathcal{A}$
5: **repeat**
6: $initialfound := \bot$
7: **for a** $\in C$ **do**
8: **if** $\{\mathbf{b} \in U \mid \mathbf{b} \to \mathbf{a}\} = \emptyset$ **then**
9: $initialfound := \top$
10: $\mathcal{L}ab := \mathcal{L}ab \cup \{(\mathbf{a}, \mathtt{in})\}$
11: $U := U \setminus \mathbf{a}$
12: $C := C \setminus \mathbf{a}$
13: **for b** $\in (U \cap \mathbf{a}^+)$ **do**
14: $\mathcal{L}ab := \mathcal{L}ab \cup \{(\mathbf{b}, \mathtt{out})\}$
15: $U := U \setminus \mathbf{b}$
16: $C := C \setminus \mathbf{b}$
17: **end for**
18: **end if**
19: **end for**
20: **until** $(initialfound)$
21: **return**$(\mathcal{L}ab, U)$

$O = \{\mathbf{a} \in S_i \mid \exists \mathbf{b} \in S \cap \mathbf{a}^- : \mathcal{L}ab(\mathbf{b}) = \mathtt{in}\}$ and $I = \{\mathbf{a} \in S_i \mid \forall\, \mathbf{b} \in S \cap \mathbf{a}^-, \mathcal{L}ab(\mathbf{b}) = \mathtt{out}\}$, with $S \equiv S_1 \cup \ldots \cup S_{i-1}$.

If $I = \emptyset$ then no argument of S_i can be labelled `in`, in particular each argument in O is labelled `out` and each arguments in $S_i \setminus O$ is labelled `undec`. Accordingly, $\mathcal{L}ab$ is extended with such a labelling (line 12) and included in E'_p (line 13). Then a new iteration of the loop is entered to process a new labelling $\mathcal{L}ab \in E_p$ (if any).

Otherwise, two cases are considered. If $O = \emptyset$ then there are no arguments to suppress from S_i, thus the base case of the recursion applies. This is a small improvement w.r.t. Def. 10 which allows to avoid an unnecessary recursive step. In the base case, the preferred labellings of $\Gamma{\downarrow_{S_i}}$ in $S_i \cap C$ are computed by means of the algorithm B-PR, and assigned to E_* (line 16). If $O \neq \emptyset$ then each argument in O is labelled `out`, and the R-PREF procedure is recursively invoked on $\Gamma{\downarrow_{S_i \setminus O}}$ and $I \cap C$ (line 19). In both cases, line 21 extends the labelling $\mathcal{L}ab$ to cover the whole component S_i — by combining it with all labelling of E_* — and update E'_p. Indeed, $\mathcal{L}ab \otimes E_*$ denotes the set $\{\mathcal{L}ab \cup \mathcal{L}ab^* \mid \mathcal{L}ab^* \in E_*\}$.

After the inner loop is exited, i.e. when all labellings of E_p have been considered, the obtained labellings E'_p are those covering $S_1, \ldots, S_i$, thus E_p is set to E'_p at line 24. After the outer loop is exited all strongly connected components have been processed, thus E_p is returned as the set of preferred labellings in C (line 26).

The following theorem shows that Algorithm 2 (and thus Algorithm 1) is correct

Theorem 1. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, Algorithm 1 returns $E_p = \mathcal{L}_{\mathcal{PR}}(\Gamma, C)$.*

A SAT-Based Approach for the Base Case

R-PREF (Alg. 2) relies on an external algorithm (B-PR) for enumerating the preferred labellings of Γ in a set of arguments C. In order to empirically prove the improvements obtained using Alg. 2, in this section we discuss a specific implementation of B-PR generalizing the approach of (Cerutti et al. 2014).

The proposed algorithm performs a search in the space of complete extensions in C, in order to maximise the set of `in` arguments (cf. Definitions 12 and 11). Each step of the search process requires to encode in a propositional formula the constraints corresponding to complete labellings of an AF in C, with opportune modifications due to the search process. Then, a SAT solver checks whether the formula is satisfiable, i.e. there exists a truth assignment of the variables such that the formula evaluates to $\top$, and if this is the case returns such an assignment.

As a first step to identify the encoding, here we provide the definition of *complete labelling of Γ in C* in a more algorithmic way than Definition 12.

Definition 13. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and $C \subseteq \mathcal{A}$ be a set of arguments. A total function $\mathcal{L}ab : \mathcal{A} \mapsto \{\mathtt{in}, \mathtt{out}, \mathtt{undec}\}$ is a* complete labelling of Γ in C *iff it satisfies the following conditions for any $\boldsymbol{a} \in C$:*

$\mathcal{L}^1_C$*:* $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{in} \Leftrightarrow \forall \boldsymbol{b} \in \boldsymbol{a}^- \mathcal{L}ab(\boldsymbol{b}) = \mathtt{out}$*;*
$\mathcal{L}^2_C$*:* $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{out} \Leftrightarrow \exists \boldsymbol{b} \in (\boldsymbol{a}^- \cap C) : \mathcal{L}ab(\boldsymbol{b}) = \mathtt{in}$*;*
$\mathcal{L}^3_C$*:* $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{undec} \Leftrightarrow \forall \boldsymbol{b} \in (\boldsymbol{a}^- \cap C), \mathcal{L}ab(\boldsymbol{b}) \neq \mathtt{in} \wedge \exists \boldsymbol{c} \in \boldsymbol{a}^- : \mathcal{L}ab(\boldsymbol{c}) = \mathtt{undec}$*;*

and the following conditions for any $\boldsymbol{a} \in (\mathcal{A} \setminus C)$:

$\mathcal{L}^1_{\mathcal{A} \setminus C}$*:* $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{out} \Leftrightarrow \exists \boldsymbol{b} \in (\boldsymbol{a}^- \cap C) : \mathcal{L}ab(\boldsymbol{b}) = \mathtt{in}$*;*
$\mathcal{L}^2_{\mathcal{A} \setminus C}$*:* $\mathcal{L}ab(\boldsymbol{a}) = \mathtt{undec} \Leftrightarrow \forall \boldsymbol{b} \in (\boldsymbol{a}^- \cap C), \mathcal{L}ab(\boldsymbol{b}) \neq \mathtt{in}$*.*

The following proposition shows that Definition 13 actually identifies the complete labellings in C as introduced in Definition 12.

Proposition 3. *Given an an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, $\mathcal{L}ab$ satisfies the conditions of Definition 13 if and only if there is a complete extension S of Γ in C such that $\mathcal{L}ab = \mathtt{Ext2Lab}(S)$.*

Sketch of proof. The proof generalizes (Cerutti et al. 2014, Proposition 2), to take into account the parameter C. □

The Complete Labelling Formula of Γ in C

Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, we are interested in identifying a boolean formula in conjunctive normal form (as required by the SAT solver), called *complete labelling formula of Γ in C* and denoted as $\Pi_{\Gamma, C}$, such that the satisfying assignments of the formula are in one-to-one correspondence with the complete labellings of Γ in C.

To this purpose, we have to introduce some notation. Let ϕ be a bijection $\phi : \{1, \ldots, |\mathcal{A}|\} \mapsto \mathcal{A}$ (the inverse map will be denoted as ϕ^{-1}): ϕ is an indexing of $\mathcal{A}$. Argument $\phi(i)$ will be sometimes referred to as argument i for brevity. For each argument i we define three boolean variables, I_i, O_i,

and U_i, with the intended meaning that I_i is $\top$ when argument i is labelled `in`, $\bot$ otherwise (and analogously O_i and U_i correspond to labels `out` and `undec`). Formally, given $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ we define the corresponding set of variables as $\mathcal{V}(\Gamma) \triangleq \cup_{i \in \phi^{-1}(\mathcal{A})} \{I_i, O_i, U_i\}$.

The following definition expresses the constraints of Definition 5 in terms of the variables $\mathcal{V}(\Gamma)$.

Definition 14. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ with $|\mathcal{A}| = k$ and a set $C \subseteq \mathcal{A}$, let $\phi : \{1, \ldots, k\} \mapsto \mathcal{A}$ be an indexing of $\mathcal{A}$. The ENC_{all}* encoding *defined on the variables in $\mathcal{V}(\Gamma)$ is given by the conjunction of the formulae listed below:*

$$\bigwedge_{i \in \phi^{-1}(C)} \Big((I_i \vee O_i \vee U_i) \wedge (\neg I_i \vee \neg O_i) \wedge (\neg I_i \vee \neg U_i) \wedge (\neg O_i \vee \neg U_i) \Big) \tag{1}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \left(I_i \vee \left(\bigvee_{\{j | \phi(j) \rightarrow \phi(i)\}} (\neg O_j) \right) \right) \tag{2}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \left(\bigwedge_{\{j | \phi(j) \rightarrow \phi(i)\}} \neg I_i \vee O_j \right) \tag{3}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \left(\bigwedge_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} \neg I_j \vee O_i \right) \tag{4}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \left(\neg O_i \vee \left(\bigvee_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} I_j \right) \right) \tag{5}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \Big(\bigwedge_{\{k | \phi(k) \rightarrow \phi(i)\}} \Big(U_i \vee \neg U_k \vee \Big(\bigvee_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} I_j \Big) \Big) \Big) \tag{6}$$

$$\bigwedge_{i \in \phi^{-1}(C)} \Big(\Big(\bigwedge_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} (\neg U_i \vee \neg I_j) \Big) \wedge \Big(\neg U_i \vee \Big(\bigvee_{\{k | \phi(k) \rightarrow \phi(i)\}} U_k \Big) \Big) \Big) \tag{7}$$

$$\bigwedge_{i \in \phi^{-1}(\mathcal{A} \setminus C)} \Big(\neg I_i \wedge (O_i \vee U_i) \wedge (\neg O_i \vee \neg U_i) \Big) \tag{8}$$

$$\bigwedge_{i \in \phi^{-1}(\mathcal{A} \setminus C)} \left(\bigwedge_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} \neg I_j \vee O_i \right) \tag{9}$$

$$\bigwedge_{i \in \phi^{-1}(\mathcal{A} \setminus C)} \left(\neg O_i \vee \left(\bigvee_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} I_j \right) \right) \tag{10}$$

$$\bigwedge_{i \in \phi^{-1}(\mathcal{A} \setminus C)} \left(U_i \vee \left(\bigvee_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} I_j \right) \right) \tag{11}$$

$$\bigwedge_{i \in \phi^{-1}(\mathcal{A} \setminus C)} \left(\bigwedge_{\{j \in \phi^{-1}(C) | \phi(j) \rightarrow \phi(i)\}} \neg U_i \vee \neg I_j \right) \tag{12}$$

Formulas (1) and (8) encode the fact that $\mathcal{L}ab$ is a total function. The other formulas are in direct correspondence with the conditions of Definition 13:

- formulas (2) and (3) encode $\mathcal{L}^1_C$;
- formulas (4) and (5) encode $\mathcal{L}^2_C$;
- formulas (6) and (7) encode $\mathcal{L}^3_C$;
- formulas (9) and (10) encode $\mathcal{L}^1_{\mathcal{A} \setminus C}$;
- formulas (11) and (12) encode $\mathcal{L}^2_{\mathcal{A} \setminus C}$.

The following proposition shows that ENC_{all} is a complete labelling formula of Γ in C, i.e. every satisfying assignment of the ENC_{all} encoding corresponds to a complete extension in C, and vice versa. For ease of notation, if a variable is not assigned to $\top$ then it is $\bot$.

Proposition 4. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and $C \subseteq \mathcal{A}$ be a set of arguments. If $\mathcal{L}ab$ is a complete labelling of Γ in C, then the assignment $\Phi_{\mathcal{V}(\Gamma)} \equiv \{(I_i, \top) \mid \mathcal{L}ab(\phi(i)) = \texttt{in}\} \cup \{(O_i, \top) \mid \mathcal{L}ab(\phi(i)) = \texttt{out}\} \cup \{(U_i, \top) \mid \mathcal{L}ab(\phi(i)) = \texttt{undec}\}$ satisfies the ENC_{all} encoding of Definition 14. Conversely, if $\Phi_{\mathcal{V}(\Gamma)}$ is a satisfying assignment of the ENC_{all} encoding, then the labelling $\mathcal{L}ab \equiv \{(\boldsymbol{a}, \texttt{in}) \mid I_{\phi^{-1}(\boldsymbol{a})} \in \Phi_{\mathcal{V}(\Gamma)}\} \cup \{(\boldsymbol{b}, \texttt{out}) \mid O_{\phi^{-1}(\boldsymbol{b})} \in \Phi_{\mathcal{V}(\Gamma)}\} \cup \{(\boldsymbol{c}, \texttt{undec}) \mid U_{\phi^{-1}(\boldsymbol{c})} \in \Phi_{\mathcal{V}(\Gamma)}\}$ is a complete labelling of Γ in C.*

Sketch of proof. The result derives from Proposition 3 by transforming logical implications in AND/OR. □

Several syntactically different encodings can be devised which, while being logically equivalent, can significantly affect the performance of the overall process (Cerutti et al. 2014). The following proposition shows the 20 non redundant encodings (i.e. it is not possible to drop out some clauses so as to obtain a simpler logically equivalent encoding) equivalent to ENC_{all}.

Proposition 5. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and $C \subseteq \mathcal{A}$ be a set of arguments. Referring to the formulae listed in Definition 14, let us define the following formulae (for the arguments in C):*

$C_1^a : (1) \wedge (2) \wedge (3) \wedge (4) \wedge (5)$
$C_1^b : (1) \wedge (4) \wedge (5) \wedge (6) \wedge (7)$
$C_1^c : (1) \wedge (2) \wedge (3) \wedge (6) \wedge (7)$

$C_2: (1) \wedge (3) \wedge (5) \wedge (7)$
$C_3: (1) \wedge (2) \wedge (4) \wedge (6)$

as well as the following ones (for the arguments in $\mathcal{A} \setminus C$):

$NC_1^a: (8) \wedge (9) \wedge (10)$
$NC_1^b: (8) \wedge (11) \wedge (12)$
$NC_2: (8) \wedge (10) \wedge (12)$
$NC_3: (8) \wedge (9) \wedge (11)$

Any encoding of the form $C \wedge NC$, where $C \in \{C_1^a, C_1^b, C_1^c, C_2, C_3\}$ and $NC \in \{NC_1^a, NC_1^b, NC_2, NC_3\}$, is equivalent to ENC_{all}, i.e. it has the same satisfying assignments of the variables in $\mathcal{V}(\Gamma)$.

Sketch of proof. The proof exploits the fact that $\mathcal{L}ab$ is a total function and shows that the conditions in $C \wedge NC$ entail those included in ENC_{all}. □

Corollary 1. *Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and $C \subseteq \mathcal{A}$ be a set of arguments. Any encoding of the form $C \wedge NC$ (see Proposition 5) is a complete labelling formula of Γ in C.*

The SAT_basepref Algorithm

We are now able to describe the proposed procedure to implement B-PR. The procedure, called SAT_basepref, resorts to five external functions: SS, I-ARGS, O-ARGS, U-ARGS and LAB. SS is a SAT solver able to prove unsatisfiability too: it accepts as input a CNF formula and returns a variable assignment satisfying the formula if it exists, ε otherwise. All the other functions accept as input a variable assignment concerning $\mathcal{V}(\Gamma)$: I-ARGS returns the corresponding set of arguments labelled as `in`, O-ARGS returns the corresponding set of arguments labelled as `out`, U-ARGS returns the corresponding set of arguments labelled as `undec`, while LAB returns the labelling corresponding to the input variable assignment. Moreover, INC_C denotes the clause $\bigvee_{i \in \phi^{-1}(C)} I_i$.

The SAT_basepref procedure (Alg. 4) initialises, at line 3, the variable E_p which stores the generated labellings. Then cnf is initialized to $\Pi_{\Gamma,C} \wedge INC_C$: $\Pi_{\Gamma,C}$ is a complete labelling formula of Γ in C, e.g. one of the equivalent encodings shown in Proposition 5, while INC_C restricts the search process to a non empty labelling (this is non restrictive due to the check of line 34). The search process is carried out by two nested **repeat-until** loops. Roughly, the inner loop (lines 8–24) resembles a depth-first search which, starting from a non-empty complete labelling in C, produces a sequence of complete labellings in C strictly ordered by set inclusion. When the sequence can no more be extended, its last element corresponds to a complete labelling in C maximizing the set of `in`-labelled arguments, namely to a preferred labelling. The outer loop (lines 5–33) drives the search: it ensures, through proper settings of the variables, that the inner loop is entered with different initial conditions, so that all preferred labellings in C are found.

The following theorem shows that Algorithm 4 is correct.

Theorem 2. *Given an AF $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle$ and a set $C \subseteq \mathcal{A}$, Algorithm 4 returns $\mathcal{L}_{\mathcal{PR}}(\Gamma, C)$.*

Algorithm 4 Enumerating the preferred labellings in a set C of a AF

SAT_basepref(Γ, C)

1: **Input:** $\Gamma = \langle \mathcal{A}, \mathcal{R} \rangle, C \subseteq \mathcal{A}$
2: **Output:** $E_p \in 2^{\mathfrak{L}(\Gamma)}$
3: $E_p := \emptyset$
4: $cnf := \Pi_{\Gamma,C} \wedge INC_C$
5: **repeat**
6: $\quad prefcand := \emptyset$
7: $\quad cnfdf := cnf$
8: $\quad$ **repeat**
9: $\quad\quad lastcompfound := SS(cnfdf)$
10: $\quad\quad$ **if** $lastcompfound != \varepsilon$ **then**
11: $\quad\quad\quad prefcand := lastcompfound$
12: $\quad\quad\quad$ **for a** $\in$ I-ARGS$(lastcompfound)$ **do**
13: $\quad\quad\quad\quad cnfdf := cnfdf \wedge I_{\phi^{-1}(\mathbf{a})}$
14: $\quad\quad\quad$ **end for**
15: $\quad\quad\quad$ **for a** $\in$ O-ARGS$(lastcompfound)$ **do**
16: $\quad\quad\quad\quad cnfdf := cnfdf \wedge O_{\phi^{-1}(\mathbf{a})}$
17: $\quad\quad\quad$ **end for**
18: $\quad\quad\quad remaining := \bot$
19: $\quad\quad\quad$ **for a** $\in C \cap$ U-ARGS$(lastcompfound)$ **do**
20: $\quad\quad\quad\quad remaining := remaining \vee I_{\phi^{-1}(\mathbf{a})}$
21: $\quad\quad\quad$ **end for**
22: $\quad\quad\quad cnfdf := cnfdf \wedge remaining$
23: $\quad\quad$ **end if**
24: $\quad$ **until** $(lastcompfound != \varepsilon)$
25: $\quad$ **if** $prefcand != \emptyset$ **then**
26: $\quad\quad E_p := E_p \cup \{\text{LAB}(prefcand)\}$
27: $\quad\quad oppsolution := \bot$
28: $\quad\quad$ **for a** $\in C \setminus$ I-ARGS$(prefcand)$ **do**
29: $\quad\quad\quad oppsolution := oppsolution \vee I_{\phi^{-1}(\mathbf{a})}$
30: $\quad\quad$ **end for**
31: $\quad\quad cnf := cnf \wedge oppsolution$
32: $\quad$ **end if**
33: **until** $(prefcand != \emptyset)$
34: **if** $E_p = \emptyset$ **then**
35: $\quad E_p = \{(\mathbf{a}, \texttt{undec}) \mid \mathbf{a} \in \mathcal{A}\}$
36: **end if**
37: **return** E_p

The Empirical Analysis

In this section, we present the results of a large experimental study examining the reduction of computational effort in enumerating preferred extensions due to the application of the proposed meta-algorithm (Alg. 1) instantiated with Alg. 4 as B-PR.

Experimental Setup

The experiments were performed on AMD Opteron™ 2.4 Ghz, 8 Gb of RAM and Linux operating system. As in the International Planning Competition (IPC) (Jiménez et al. 2012), a limit of 15 minutes was imposed to compute the preferred labellings for each AF. No limit was imposed on the RAM usage, but a run fails at saturation of the available memory. Moreover, we adopted the IPC speed score,

also borrowed from the planning community, which is defined as follows. For each AF, each system gets a score of $1/(1+\log_{10}(T/T^*))$, where T is its execution time and T^* the best execution time among the compared systems, or a score of 0 if it fails in that case. Runtimes below 0.01 sec get by default the maximal score of 1.

In order to evaluate PREF, we compared it with the implementation of the approach discussed in (Cerutti et al. 2014), which has been proved to be competitive w.r.t other state-of-the-art systems. In the following we refer to SCC-P as the prototype implementing the algorithm PREF with C_2 and NC_2, and SAT-P as the current version (rev. 128) of the software presented in (Cerutti et al. 2014) with C_2. We chose to compare SCC-P not with the implementation of Alg. 4 called as PREF$(\Gamma, \mathcal{A})$, but with SAT-P in order to give additional strength to this empirical evaluation. Both SCC-P and SAT-P are implemented in C++ and both adopt Glucose (Audemard and Simon 2009; 2012) as integrated SAT solver.

We designed three experiments aimed at addressing the following hypotheses:

$I1$: on Γ s.t. $|\text{SCC}_\Gamma| = 1$, SCC-P performs worse than SAT-P;

$I2$: there exists a value χ such that on Γ where $|\text{SCC}_\Gamma| > \chi$, SCC-P performs better that SAT-P;

$I3$: on Γ s.t. $|\text{SCC}_\Gamma| > \chi$, the greater $|\mathcal{E}_{\mathcal{PR}}(\Gamma)|$, the more SAT-P performs worse than SCC-P.

To check the first hypothesis, we conducted an experiment on 790 randomly generated AFs (Γ), s.t. $|\text{SCC}_\Gamma| = 1$, varying $\mathcal{A}$ between 25 and 250 with a step of 25.

Coming to the second hypothesis, we randomly generated 720 AFs varying $|\text{SCC}_\Gamma|$ between 5 and 45 with a step of 5. The size of the SCCs is determined by normal distributions with means between 20 and 40 with a step of 5, and with a fixed standard deviation of 5. We similarly varied the probability of having attacks between arguments among SCCs.

Finally, we addressed the third hypothesis conducting an experiment on 2800 AFs randomly generated as before, such that $50 \leq |\text{SCC}_\Gamma| \leq 80$ with a step of 5.

In the following we rely on the Wilcoxon Signed-Rank Test (WSRT) in order to identify significant subset of data.

Experiments on Γ s.t. $|\text{SCC}_\Gamma| = 1$

Figure 1 depicts the significant ($p < 0.05$) IPC values (normalised to 100) for SCC-P and SAT-P when $|\text{SCC}_\Gamma| = 1$ varying $|\mathcal{A}|$ and $|\mathcal{E}_{\mathcal{PR}}(\Gamma)|$. From this, we can observe that there is a significant statistical evidence supporting hypothesis $I1$. It is worth to mention that when $|\text{SCC}_\Gamma| = 1$, PREF (i) identifies the presence of a single SCC, (ii) calls the B-PR on the single computed SCC. Since B-PR is a variation (cf. Alg. 4) of the algorithm SAT-P, the great difference of performance shown in Fig. 1 suggests that refining the prototypical implementation we used in these experiments would lead to better results.

Experiments on Γ s.t. $5 \leq |\text{SCC}_\Gamma| \leq 45$

Figure 2 summarises the results obtained in the experiment aimed at proving hypothesis $I2$, about the existence of a number χ such that on Γ with $|\text{SCC}_\Gamma| > \chi$, SCC-P performs better than SAT-P.

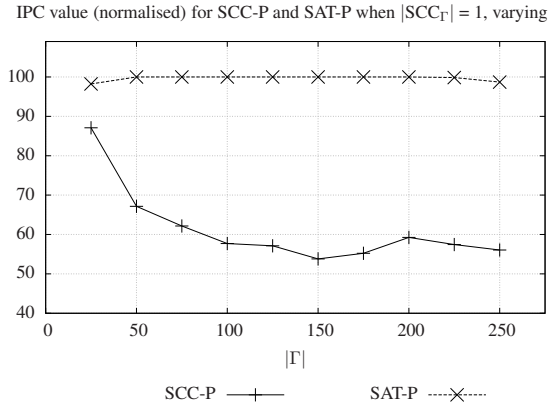


Figure 1: Significant ($p < 0.05$) IPC values (normalised) for SCC-P and SAT-P when $|\text{SCC}_\Gamma| = 1$ varying $|\mathcal{A}|$.

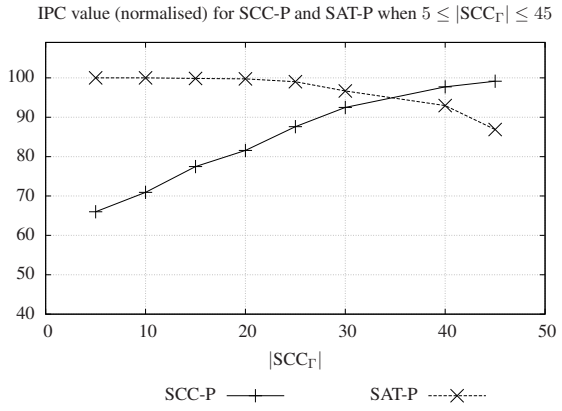


Figure 2: Significant ($p < 0.05$) IPC values (normalised) for SCC-P and SAT-P when $5 \leq |\text{SCC}_\Gamma| \leq 45$ varying $|\text{SCC}_\Gamma|$.

In particular, it turns out that:

- for $|\text{SCC}_\Gamma| = 30$, $Md(\text{SCC-P}) = 6.43$, $Md(\text{SAT-P}) = 5.63$, $z = -2.87$, $p < 0.01$;
- for $|\text{SCC}_\Gamma| = 35$, $Md(\text{SCC-P}) = 8.81$, $Md(\text{SAT-P}) = 8.53$, $z = -0.35$, $p = 0.73$;
- for $|\text{SCC}_\Gamma| = 40$, $Md(\text{SCC-P}) = 10.49$, $Md(\text{SAT-P}) = 12.53$, $z = -3.56$, $p < 0.01$;

where $Md(\cdot)$ indicates the median of execution times.

In other terms, when $|\text{SCC}_\Gamma| = 35$, the performances of SCC-P and SAT-P are statistically indistinguishable ($p = 0.73 > 0.05$). However, for $|\text{SCC}_\Gamma| > 35$, Fig. 2 shows that SCC-P performs significantly ($p < 0.05$) better than SAT-P.

Experiments on Γ s.t. $50 \leq |\text{SCC}_\Gamma| \leq 80$

Figure 3 depicts the medians of significant execution times of SCC-P and SAT-P varying the number of computed labellings. It illustrates a dependency of the execution times

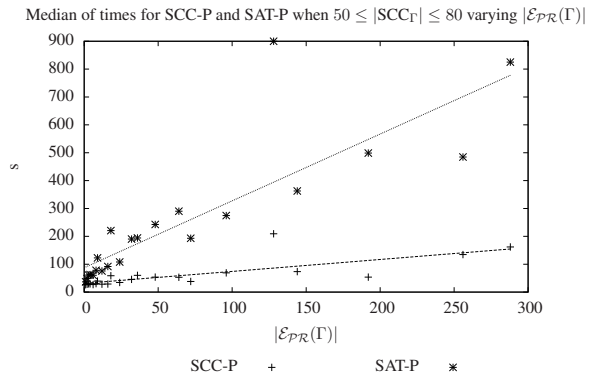


Figure 3: Significant median values for SCC-P and SAT-P when $50 \leq |\text{SCC}_\Gamma| \leq 80$ varying $|\mathcal{E}_{\mathcal{PR}}(\Gamma)|$, and their regression to the function $f(x) = a\ x + b$: SCC-P, $a = 0.43$, $b = 31.33$; SAT-P, $a = 2.40$, $b = 87.53$.

on the number of computed preferred labellings. Given the linear regression of these data, it turns out that the execution times of SAT-P grows $\approx 2 \cdot |\mathcal{E}_{\mathcal{PR}}(\Gamma)|$, while SCC-P grows $\approx \frac{1}{2} \cdot |\mathcal{E}_{\mathcal{PR}}(\Gamma)|$, thus supporting hypothesis $I3$.

Moreover, the third experiment provides additional statistically significant ($p < 0.05$) evidence in support of hypothesis $I2$, as Figure 3 summarises. Roughly speaking, computing the increment of performance as the average of the difference of execution times normalised on the maximum time between SCC-P and SAT-P, we can conclude that there is a significant increment of performances up to 56%.

Conclusions

In this paper we devised and evaluated a general parametric algorithm in order to decompose the preferred extensions enumeration problem into restricted subframeworks. To this purpose, several research challenges have been tackled.

First, an efficient algorithmic implementation of the SCC-recursive schema, originally introduced in a declarative form, has been developed, also leading to a labelling-based formulation. Let us notice that, recently, an approach has been proposed in (Liao, Lei, and Dai 2013) which partially exploits the SCC-recursive schema. In particular, while it borrows from (Baroni, Giacomin, and Guida 2005) the idea of decomposing the argumentation framework into its SCCs, it does not exploit the recursion step to achieve a deeper decomposition of the AF. Other "splitting" techniques have been proposed in literature. In particular, in the context of argumentation dynamics (Baumann et al. 2012) introduces a decomposition of an argumentation framework into two parts, while (Dvořák, Pichler, and Woltran 2012) exploits a tree-based decomposition. A general study of decomposability properties w.r.t. arbitrary partitions of an argumentation framework is also presented in (Baroni et al. 2012). Comparing different approaches for dividing an AF in subframeworks represents an interesting future work.

A second contribution of the present paper is the generalization of the approach in (Cerutti et al. 2014) to the computation of labellings in a restricted sub-framework, which is mandatory for applying the SCC-recursive schema. As acknowledged in (Cerutti et al. 2014), the relationship between argumentation semantics and the satisability problem has been already considered in the literature, e.g. in (Besnard and Doutre 2004; Dvořák et al. 2012; Arieli and Caminada 2013), where different encodings for labellings of an AF according to several semantics have been introduced. Generalizing these proposals in order to integrate them into the SCC-recursive schema is another challenge for future work.

Finally, an empirical investigation has been carried out to show a statistical evidence in favour of the reduction of computational effort by the exploitation of the SCC-recursive schema. In particular, as shown in the previous section, despite the fact that the current prototypical implementation needs further refinements (cf. the results of experiments on Γ s.t. $|\text{SCC}_\Gamma| = 1$), for Γ with $|\text{SCC}_\Gamma| > \chi = 35$ there is a statistical evidence showing how implementing the SCC-recursive schema reduces the computational effort of enumerating the preferred labellings. Moreover, as depicted in Fig. 3, the execution time of the SCC-recursive implementation is less sensible to the number of labellings, i.e. its execution time grows less than the non SCC-recursive implementation when the number of computed preferred labellings increases. This suggests to explore an hybrid approach where, at each recursion step, a choice is made on whether computing the preferred labellings with a direct call to B-PR or applying the SCC-recursive schema. Such choice can be based on an heuristic to estimate the number of SCCs and labellings, and can be easily implemented with an additional step after line 5 of Alg. 2.

Several other future works are envisaged. First, we want to strengthen the statistical evidence in favour of the SCC-recursive schema by exploiting and comparing different encodings of complete labellings derived from ENC_{all}, including the redundant ones. Moreover, Alg. 2 can be instantiated with a different B-PR algorithm, e.g. generalizing the algorithms presented in (Egly, Alice Gaggl, and Woltran 2010; Nofal, Atkinson, and Dunne 2014; Dvořák et al. 2014) for computing labellings in sub-frameworks.

Finally, we intend to apply the devised meta-algorithm to stable and CF2 semantics, which directly fit the SCC-recursive schema (Baroni, Giacomin, and Guida 2005), as well as to semi-stable and ideal semantics by exploiting the relationship with preferred semantics.

Acknowledgements

The authors would like to acknowledge the use of the University of Huddersfield Queensgate Grid in carrying out this work. The authors also thank the anonymous reviewers for their helpful comments.

References

Arieli, O., and Caminada, M. W. 2013. A QBF-based formalization of abstract argumentation semantics. *Journal of Applied Logic* 11(2):229–252.

Audemard, G., and Simon, L. 2009. Predicting Learnt Clauses Quality in Modern SAT Solvers. In *Proceedings*

of the 21st International Joint Conference on Artifical Intelligence (IJCAI 2009), 399–404.

Audemard, G., and Simon, L. 2012. GLUCOSE 2.1. http://www.labri.fr/perso/lsimon/glucose/.

Baroni, P., and Giacomin, M. 2004. A General Recursive Schema for Argumentation Semantics. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2004)*, 783–787.

Baroni, P.; Boella, G.; Cerutti, F.; Giacomin, M.; van der Torre, L.; and Villata, S. 2012. On Input/Output Argumentation Frameworks. In *Proceedings of the 4th International Conference on Computational Models of Arguments (COMMA 2012)*, 358–365.

Baroni, P.; Cerutti, F.; Dunne, P. E.; and Giacomin, M. 2013. Automata for Infinite Argumentation Structures. *Artificial Intelligence* 203(0):104–150.

Baroni, P.; Caminada, M.; and Giacomin, M. 2011. An introduction to argumentation semantics. *Knowledge Engineering Review* 26(4):365–410.

Baroni, P.; Giacomin, M.; and Guida, G. 2005. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence* 168(1-2):165–210.

Baumann, R.; Brewka, G.; Dvořák, W.; and Woltran, S. 2012. Parameterized Splitting: A Simple Modification-Based Approach. In Erdem, E.; Lee, J.; Lierler, Y.; and Pearce, D., eds., *Correct Reasoning*, volume 7265 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 57–71.

Besnard, P., and Doutre, S. 2004. Checking the acceptability of a set of arguments. In *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR 2004)*, 59–64.

Caminada, M. 2006. On the Issue of Reinstatement in Argumentation. In *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA 2006)*, 111–123.

Cerutti, F.; Dunne, P. E.; Giacomin, M.; and Vallati, M. 2013. A SAT-based Approach for Computing Extensions in Abstract Argumentation. In *Second International Workshop on Theory and Applications of Formal Argumentation (TAFA-13)*.

Cerutti, F.; Dunne, P. E.; Giacomin, M.; and Vallati, M. 2014. Computing Preferred Extensions in Abstract Argumentation: A SAT-Based Approach. In Black, E.; Modgil, S.; and Oren, N., eds., *TAFA 2013*, volume 8306 of *Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg. 176–193.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2009. *Introduction to Algorithms*. MIT Press.

Dung, P. M.; Mancarella, P.; and Toni, F. 2006. A dialectic procedure for sceptical, assumption-based argumentation. In *Prooceedings of the 1st International Conference on Computational Models of Arguments (COMMA 2006)*, 145–156.

Dung, P. M. 1995. On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming, and n-Person Games. *Artificial Intelligence* 77(2):321–357.

Dunne, P. E., and Wooldridge, M. 2009. Complexity of abstract argumentation. In Rahwan, I., and Simari, G., eds., *Argumentation in AI*. Springer-Verlag. chapter 5, 85–104.

Dvořák, W.; Gaggl, S. A.; Wallner, J.; and Woltran, S. 2011. Making Use of Advances in Answer-Set Programming for Abstract Argumentation Systems. In *Proceedings of the 19th International Conference on Applications of Declarative Programming and Knowledge Management (INAP 2011)*.

Dvořák, W.; Järvisalo, M.; Wallner, J. P.; and Woltran, S. 2012. Complexity-Sensitive Decision Procedures for Abstract Argumentation. In *Proceedings of 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*, 54–64.

Dvořák, W.; Järvisalo, M.; Wallner, J. P.; and Woltran, S. 2014. Complexity-sensitive decision procedures for abstract argumentation. *Artificial Intelligence* 206:53–78.

Dvořák, W.; Pichler, R.; and Woltran, S. 2012. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artificial Intelligence* 186:1–37.

Egly, U.; Alice Gaggl, S.; and Woltran, S. 2010. Answer-set programming encodings for argumentation frameworks. *Argument & Computation* 1(2):147–177.

Jiménez, S.; de la Rosa, T.; Fernández, S.; Fernández, F.; and Borrajo, D. 2012. A review of machine learning for automated planning. *Knowledge Engineering Review* 27(4):433–467.

Liao, B.; Lei, L.; and Dai, J. 2013. Computing Preferred Labellings by Exploiting SCCs and Most Sceptically Rejected Arguments. In *Second International Workshop on Theory and Applications of Formal Argumentation (TAFA-13)*.

Nofal, S.; Atkinson, K.; and Dunne, P. E. 2014. Algorithms for decision problems in argument systems under preferred semantics. *Artificial Intelligence* 207:23–51.

Nofal, S.; Dunne, P. E.; and Atkinson, K. 2012. On Preferred Extension Enumeration in Abstract Argumentation. In *Proceedings of 3rd International Conference on Computational Models of Arguments (COMMA 2012)*, 205–216.

On the Revision of Argumentation Systems: Minimal Change of Arguments Statuses

Sylvie Coste-Marquis **Sébastien Konieczny** **Jean-Guy Mailly** **Pierre Marquis**

CRIL, CNRS – Université d'Artois, Lens, France
{coste,konieczny,mailly,marquis}@cril.fr

Abstract

In this paper, we investigate the revision of argumentation systems à la Dung. We focus on revision as minimal change of the arguments status. Contrarily to most of the previous works on the topic, the addition of new arguments is not allowed in the revision process, so that the revised system has to be obtained by modifying the attack relation only. We introduce a language of revision formulae which is expressive enough for enabling the representation of complex conditions on the acceptability of arguments in the revised system. We show how AGM belief revision postulates can be translated to the case of argumentation systems. We provide a corresponding representation theorem in terms of minimal change of the arguments statuses. Several distance-based revision operators satisfying the postulates are also pointed out, along with some methods to build revised argumentation systems. We also discuss some computational aspects of those methods.

Introduction

In this paper, we investigate the revision issue for abstract argumentation systems à la Dung (Dung 1995). Argumentation systems are directed graphs, where nodes correspond to arguments and arcs to attacks between arguments. In such systems, the status (acceptance) of each argument depends on the chosen acceptability semantics (grounded, preferred, stable – among others).

In his book, Gärdenfors (1988) introduced abstractly belief change as the operation allowing to change the epistemic status of a piece of information with respect to the epistemic state of an agent. There are three possible statuses: accepted, rejected or undetermined. And revision, contraction and expansion are defined as the possible transitions between these statuses, as illustrated by Figure 1.

In order to instantiate Gärdenfors' general definition of belief change within Dung's argumentation theory, it is first necessary to define what are the available pieces of information and what these statuses mean. In Dung's argumentation theory, the basic pieces of information are the arguments of the system, and their statuses depend on the acceptability semantics under consideration. Thus, following Gärdenfors, it does not make sense to study the revision of argumentation systems directly on the attack graph, independently of any semantics. Stated otherwise, the revision of a given argumentation system under two different semantics may easily lead to two different results. For instance, in the case of the argumentation system AF given in Figure 2, we note that under the stable and preferred semantics, d belongs to every extension, whereas d does not belong to any extension in the case of the grounded semantics. So, revising AF in order to accept d does not need any change for the stable or the preferred semantics, but a change is required for the grounded semantics.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

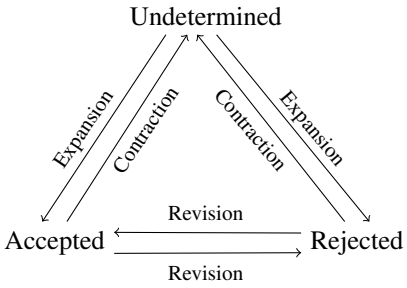


Figure 1: Gärdenfors' epistemic transitions

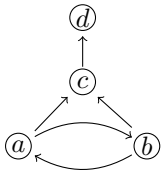


Figure 2: Revising AF may lead to different results.

In this paper, we focus on revision as minimal change of the arguments statuses. To be more precise, under a chosen semantics, given an argumentation system and a revision formula expressing how the statuses of some arguments has to be changed, we want to derive argumentation systems which satisfy the revision formula, and are such that the corresponding extensions are as close as possible to the extensions of the input system. In most of the previous works on the topic, the change in the input system is a

modification of the set of arguments. Contrarily to those works, we do not allow the addition of new arguments in the revision process, so that the revised system has to be obtained by modifying the attack relation, only. Especially, the revision formula does not indicate *why* the statuses of arguments have changed, which makes sense in cases when the agent learns an unjustified information from a trustworthy source. More details can be found in Section "Discussion".

Minimal change of the attack graph can be considered as a criterion for defining the revised systems, but the acceptance statuses of arguments is more fundamental information. Accordingly, ensuring a minimal change of these statuses is more important than (and not compatible with)[1] ensuring a minimal change of the attack graph.

To ensure such a minimal change of arguments statuses, we define revision as two step process: first, we revise extensions, without considering the attack relation. Then, from the output of this first step, we generate argumentation systems such that their extensions are the expected ones. This second step may use minimal change on the argumentation graph, as a second criterion.

The rest of the paper is organized as follows. After a short introduction to Dung's theory of abstract argumentation, we introduce a language of revision formulae which is expressive enough for enabling the representation of complex conditions on the acceptability of arguments in the revised system. Then we show how AGM belief revision postulates can be translated to the case of argumentation systems. We provide a corresponding representation theorem in terms of minimal change of the arguments statuses. Several distance-based revision operators satisfying the postulates are pointed out. Then we present methods for associating argumentation systems with the obtained sets of extensions. We discuss some computational aspects of the revision process. Proofs are omitted for space reasons.

Preliminaries

We start with a very short introduction to Dung's theory of argumentation (see (Dung 1995) for more details). A (finite) argumentation system (also referred to as an argumentation framework) is a pair $AF = \langle A, R\rangle$ where A is a (finite and non empty) set of so-called arguments and R is a binary relation over A (a subset of $A \times A$). In the following, A is supposed to contain at least two elements and the attack relation R is supposed to be irreflexive, i.e., self-contradicting arguments are rejected. AFs_A denotes the set of all such systems on the set of arguments A.

Arguments and attacks are considered in an abstract way as atomic concepts. Especially we do not assume any underlying logical setting in which arguments and attacks would be defined.

An argument $a \in A$ is acceptable with respect to a set of arguments $S \subseteq A$ whenever it is defended by S, i.e., for every $b \in A$ s.t. $(b, a) \in R$, there exists $c \in S$ such that $(c, b) \in R$. We say that a subset S of A is conflict-free if and only if for every $a, b \in S$, we have $(a, b) \notin R$. A subset S of A is admissible for AF if and only if S is conflict-free and acceptable with respect to S. "Solutions" of an argumentation systems are sets S of arguments accepted together. Some semantics σ (especially, the complete, the preferred, the stable, and the grounded semantics) can be considered for capturing formally this notion, and each of them gives rise to a specific notion of extension. For instance:

- S is a complete extension of AF if and only if it is an admissible set and every argument which is acceptable with respect to S belongs to S,
- S is a preferred extension of AF if and only if it is maximal (with respect to set inclusion) in the set of admissible sets for AF,
- S is a stable extension of AF if and only if S is conflict-free and $\forall a \in A \setminus S$, $\exists b \in S$ such that $(b, a) \in R$,
- S is the grounded extension of AF if and only if it is the smallest element (with respect to set inclusion) among the complete extensions.

$Ext_\sigma(AF)$ denotes the set of extensions of AF for the semantics σ. In this work, we focus on the skeptical policy to define the epistemic status of an argument. Thus, the epistemic status of any argument $a \in A$ with respect to the epistemic state represented by AF is given by: a is accepted if a belongs to every $\varepsilon \in Ext_\sigma(AF)$, a is refused if a does not belong to any $\varepsilon \in Ext_\sigma(AF)$, and a is undefined in the remaining case.

Moreover, we use the following notation about minimal elements of a set. For any pre-order $\leq$ over a set E, $<$ denotes the strict part of $\leq$ and $\simeq$ denotes the indifference relation associated with $\leq$. Given a set E and a pre-order $\leq$, the minimal elements of E w.r.t. $\leq$ are $\min(E, \leq) = \{e \in E | \nexists e' \in E, e' < e\}$.

On Revision Formulae

We want to define a revision setting for Dung's argumentation systems in which sophisticated revision formulae can be taken into account, and not only the fact that a given argument should be accepted or refused. To this end, we consider a logical language $\mathcal{L}_A$, where negation is used to denote the fact that a given argument should be refused, and formulae can be connected using conjunction and disjunction.

Definition 1. *Given $A = \{\alpha_1, \ldots, \alpha_k\}$ a set of arguments, $\mathcal{L}_A$ is the language generated by the following context-free grammar in BNF:*

$$\begin{array}{lll} arg & ::= & \alpha_1 | \ldots | \alpha_k \\ \Phi & ::= & arg | \neg\Phi | (\Phi \wedge \Phi) | (\Phi \vee \Phi) \end{array}$$

For instance, $\varphi_1 = (a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c)$ expresses that in the revised epistemic state, a must be accepted and b and c must be both accepted or both refused. The epistemic status of such a formula φ from $\mathcal{L}_A$ in an argumentation system $AF \in \mathsf{AFs}_A$ for a given semantics σ is given by:

Definition 2. *Let $\varepsilon \subseteq A$ and $\varphi \in \mathcal{L}_A$. The concept of* satisfaction *of φ by ε, noted $\varepsilon \mid\!\sim \varphi$, is defined inductively as follows:*

[1] See Section "Revision at the System Level" for details.

- *If* $\varphi = a \in A$*, then* $\varepsilon \vdash\!\sim \varphi$ *iff* $a \in \varepsilon$*,*
- *If* $\varphi = (\varphi_1 \wedge \varphi_2)$*,* $\varepsilon \vdash\!\sim \varphi$ *iff* $\varepsilon \vdash\!\sim \varphi_1$ *and* $\varepsilon \vdash\!\sim \varphi_2$*,*
- *If* $\varphi = (\varphi_1 \vee \varphi_2)$*,* $\varepsilon \vdash\!\sim \varphi$ *iff* $\varepsilon \vdash\!\sim \varphi_1$ *or* $\varepsilon \vdash\!\sim \varphi_2$*,*
- *If* $\varphi = \neg\psi$*,* $\varepsilon \vdash\!\sim \varphi$ *iff* $\varepsilon \not\vdash\!\sim \psi$*.*

Then for any AF *in* AFs_A*, and any semantics* σ*, we say that:*

- φ *is* accepted *w.r.t.* AF*, noted* $AF \vdash\!\sim_\sigma \varphi$*, if* $\varepsilon \vdash\!\sim \varphi$ *for every* $\varepsilon \in Ext_\sigma(AF)$*,*
- φ *is* refused *w.r.t.* AF*, noted* $AF \vdash\!\sim_\sigma \neg\varphi$*, if* $\varepsilon \vdash\!\sim \varphi$ *for no* $\varepsilon \in Ext_\sigma(AF)$*,*
- φ *is* undefined *w.r.t.* AF *in the remaining case.*

Inference $\vdash\!\sim_\sigma$ *can be extended to the case of a set* S *of argumentation systems by considering* $Ext_\sigma(S) = \bigcup_{AF \in S} Ext_\sigma(AF)$*.*

This language, used in next sections to define the revision of an argumentation framework, allows to change the status of an information φ to accepted (revise by φ) or rejected (revise by $\neg\varphi$), but not to undetermined. It is normal: Gärdenfors defines the change of status of a belief from accepted or rejected to undetermined as a *contraction*, as explained in the Introduction (see Fig. 1).

From now on we call *candidate*[2] any subset ε of A. Candidates can be interpreted as *interpretations* of revision formulae. Continuing the previous example, if $A = \{a, b, c\}$, φ_1 is satisfied by the candidates from $\{\{a\}, \{a, b, c\}\}$. Thus, for the grounded semantics, φ_1 is accepted w.r.t. AF_1 with $R_1 = \{(b, c), (c, b)\}$ but is refused w.r.t. AF_2 with $R_2 = \{(a, b), (b, a)\}$.

We define consistency in a classical way:

Definition 3. *Given a formula* φ*,* $\mathcal{A}_\varphi$ *denotes the set of candidates satisfying* φ*.* φ *is said to be* consistent *iff* $\mathcal{A}_\varphi \neq \emptyset$*.*

In the general case, $\mathcal{A}_\varphi$ is not the set of all σ-extensions of an AF in AFs_A. Consider for instance, $A = \{a, b, c\}$ and $\varphi_1 = (a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c)$. $\mathcal{A}_{\varphi_1} = \{\{a\}, \{a, b, c\}\}$, and there is no AF in AFs_A such that $Ext_\sigma(AF) = \mathcal{A}_{\varphi_1}$ for $\sigma =$ grounded, $\sigma =$ preferred or $\sigma =$ stable.

In this case, it is enough to choose two argumentation frameworks to cover the extensions $\{\{a\}, \{a, b, c\}\}$ (for instance, in the first AF a attacks b and c, and in the second AF the attack relation is empty). Note that in the general case, increasing the number of frameworks is not enough to capture the expected extensions. In order to characterize formulae that can be associated to a set of frameworks and a semantics, a concept of σ-representability can be defined as follows:

Definition 4. *A set* $\mathcal{C}$ *of candidates is* σ-representable *iff there exists a set* $\mathcal{S}$ *of argumentation systems in* AFs_A *such that* $C = Ext_\sigma(S)$*.*
A formula $\varphi \in \mathcal{L}_A$ *is* σ-representable *iff* $\mathcal{A}_\varphi$ *is* σ-representable*.*

When $A = \{a, b, c\}$, $\varphi_1 = (a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c)$ is σ-representable for $\sigma =$ grounded, $\sigma =$ preferred or $\sigma =$ stable since $\{\{a\}, \{a, b, c\}\} = Ext_\sigma(AF_3) \cup Ext_\sigma(AF_4)$ where $R_3 = \{(a, b), (a, c)\}$ and $R_4 = \emptyset$. Contrastingly, $\varphi_2 = \neg a \wedge \neg b \wedge \neg c$ is grounded-representable and preferred-representable but not stable-representable. $\varphi_3 = a \wedge \neg a$ is neither grounded-representable nor preferred-representable, but is stable-representable (consider AF_5 such that $R_5 = \{(a, b), (b, c), (c, a)\}$).

A form of consistency can be defined to take account for the semantics:

Definition 5. *Given* σ *a semantics, a formula* $\varphi \in \mathcal{L}_A$ *is* σ*-consistent iff* φ *is consistent and* φ *is* σ*-representable.*

From σ-consistency, we define a notion of model:

Definition 6. *Given a formula* $\varphi \in \mathcal{L}_A$ *and a semantics* σ*, the set of models of* φ *is defined by*

$$\mathcal{A}^\sigma_\varphi = \{\varepsilon \in \mathcal{A}_\varphi | \{\varepsilon\} \textit{ is } \sigma\textit{-representable}\}.$$

A last point about formulae is the definition of equivalence. Two formulae $\varphi, \psi \in \mathcal{L}_A$ are said σ-equivalent, noted $\varphi \equiv_\sigma \psi$, if and only if $\mathcal{A}^\sigma_\varphi = \mathcal{A}^\sigma_\psi$.

Revision at the Extension Level

In order to define revision operators, our approach follows a two-step process. Intuitively, the process first selects from models of φ those as close as possible to the σ-extensions of AF. This selection has to ensure the minimal change of arguments statuses. Then, the second step generates the argumentation systems such that the union of their σ-extensions precisely coincides with the selected candidates.

We define a revision operator on argumentation systems as a mapping associating a set of argumentation systems with the input argumentation system and the input revision formula:

Definition 7. *Given any set of arguments* A*, a* revision operator *on argumentation systems* $\star$ *is a mapping from* $\mathsf{AFs}_A \times \mathcal{L}_A$ *to* 2^{AFs_A}*.*

Clearly, the result of the revision of an argumentation system is not a unique argumentation system in the general case, but a set of argumentation systems. The reason is quite simple: there can be several possible results which have exactly the same maximum plausibility. So in this case there is no reason to select just one of them (we will return to this point later on). If this is problematic for a particular application, a selection function can be used as a tie-break rule for ensuring the unicity of the result (just like, for instance, the maxichoice selection function considered in AGM belief revision (Gärdenfors 1988)).

Of course, each mapping from $\mathsf{AFs}_A \times \mathcal{L}_A$ to 2^{AFs_A} is not a reasonable revision operator. For instance, the constant, yet trivial operator defined by $AF \star \varphi = \emptyset$ should be discarded.

In order to identify interesting revision operators, we have to identify the logical properties which guarantee a rational behaviour. Such an axiomatic approach is standard in

[2] That is to say "candidate to be an extension".

logic, and the AGM postulates (Alchourrón, Gärdenfors, and Makinson 1985; Katsuno and Mendelzon 1991) have been pointed out for characterizing valuable revision operators in a logical setting. As in (Qi, Liu, and Bell 2006), we can revisit these postulates in a set-theoretic framework, here suited to the argumentation case.

Let $\mathcal{S}$ be a set of argumentation systems AF in AFs_A. The counterpart of AGM postulates in the argumentation case is given by:

(AE1) $Ext_\sigma(AF \star \varphi) \subseteq \mathcal{A}^\sigma_\varphi$

(AE2) If $Ext_\sigma(AF) \cap \mathcal{A}^\sigma_\varphi \neq \emptyset$, then $Ext_\sigma(AF \star \varphi) = Ext_\sigma(AF) \cap \mathcal{A}^\sigma_\varphi$

(AE3) If φ is σ-consistent, then $Ext_\sigma(AF \star \varphi) \neq \emptyset$

(AE4) If $\varphi \equiv_\sigma \psi$, then $Ext_\sigma(AF \star \varphi) = Ext_\sigma(AF \star \psi)$

(AE5) $Ext_\sigma(AF \star \varphi) \cap \mathcal{A}^\sigma_\psi \subseteq Ext_\sigma(AF \star (\varphi \wedge \psi))$

(AE6) If $Ext_\sigma(AF \star \varphi) \cap \mathcal{A}^\sigma_\psi \neq \emptyset$, then $Ext_\sigma(AF \star (\varphi \wedge \psi)) \subseteq Ext_\sigma(AF \star \varphi) \cap \mathcal{A}^\sigma_\psi$

(AE1) states that the σ-extensions of the resulting set of argumentation systems must be among the models of φ. **(AE2)** demands that if there are σ-extensions of the input system satisfying φ, then the resulting σ-extensions must coincide with them. **(AE3)** requires the resulting set of σ-extensions to be non-empty as soon as φ is σ-consistent. **(AE4)** concerns the irrelevance of syntax: the revision by two formulae must be identical if the formulae are equivalent. The last two postulates **(AE5)** and **(AE6)** express a minimal change principle with respect to the arguments statuses: changes of the statuses of the arguments are expected to be minimal with respect to the input system. In particular, these postulates give the expected behaviour of the operator when an argumentation system is revised by a conjunction of formulae.

Interestingly, as in the logical case, we can derive a representation theorem which characterizes exactly the revision operators satisfying the postulates in a more constructive way. To this end, we first need to present a counter-part of the notion of faithful assignment (Katsuno and Mendelzon 1991) in the argumentation setting:

Definition 8. *A* faithful assignment *is a mapping associating any argumentation system $AF = \langle A, R\rangle$ (under a semantics σ) with a total pre-order $\leq^\sigma_{AF}$ on the set of candidates such that:*

1. *if $\varepsilon_1 \in Ext_\sigma(AF)$ and $\varepsilon_2 \in Ext_\sigma(AF)$, then $\varepsilon_1 \simeq^\sigma_{AF} \varepsilon_2$,*
2. *if $\varepsilon_1 \in Ext_\sigma(AF)$ and $\varepsilon_2 \notin Ext_\sigma(AF)$, then $\varepsilon_1 <^\sigma_{AF} \varepsilon_2$.*

The representation theorem can then be stated as follows:

Proposition 1. *Given a semantics σ, a revision operator $\star$ satisfies the rationality postulates **(AE1)** - **(AE6)** iff there exists a faithful assignment which maps every system $AF = \langle A, R\rangle$ to a total pre-order $\leq^\sigma_{AF}$ so that*

$$Ext_\sigma(AF \star \varphi) = \min(\mathcal{A}^\sigma_\varphi, \leq^\sigma_{AF}).$$

This theorem is important for defining operators satisfying the rationality postulates, as the ones presented in the next section.

Distance-Based Revision

Let us now present some (pseudo-)distance-based revision operators satisfying the rationality postulates **(AE1)** - **(AE6)**.

Let d be any pseudo-distance[3] on 2^A, for instance, the Hamming distance given by $d_H(\varepsilon_1, \varepsilon_2) = |(\varepsilon_1 \setminus \varepsilon_2) \cup (\varepsilon_2 \setminus \varepsilon_1)|$. Given $\varepsilon \in 2^A$ and $\mathcal{E} \subseteq 2^A$, d can be extended to a "distance" between ε and $\mathcal{E}$, by stating that $d(\varepsilon, \mathcal{E}) = \min_{\varepsilon' \in \mathcal{E}} d(\varepsilon, \varepsilon')$. For any argumentation system $AF \in \mathsf{AFs}_A$, this distance induces a total pre-order between candidates $\varepsilon_1, \varepsilon_2 \in 2^A$ given by

$$\varepsilon_1 \leq^{\sigma,d}_{AF} \varepsilon_2 \text{ iff } d(\varepsilon_1, Ext_\sigma(AF)) \leq d(\varepsilon_2, Ext_\sigma(AF)).$$

On this ground, revision operators can be defined by:

Definition 9. *Let σ be any given semantics. A* pseudo-distance-based revision operator $\star^d$ *is any revision operator for which there exists a pseudo-distance d on 2^A such that for every AF and every φ, we have $Ext_\sigma(AF \star^d \varphi) = \min(\mathcal{A}^\sigma_\varphi, \leq^{\sigma,d}_{AF})$.*

Proposition 2. *Let σ be any semantics. Any pseudo-distance-based revision operator $\star^d$ satisfies the rationality postulates **(AE1)** - **(AE6)**.*

Let us now define another family of pseudo-distance-based operators, which take advantage of labellings. Let us first recall that, instead of using extensions, the solutions of an argumentation system can be expressed using the concept of labelling (Caminada 2006).

Formally, a labelling is a mapping L associating a label in, $undec$ or out with every argument of the set A. The stronger notion of reinstatement labelling depends on the attack relation R: an argument a is labelled in iff every argument attacking a is out; an argument a is out iff there exists an argument in attacking a; an argument is $undec$ iff it is neither in nor out. These reinstatement labellings correspond to Dung's complete extensions in a bijective way. Thus, for any complete extension ε, the associated reinstatement labelling is such that every argument $a \in \varepsilon$ is in, every argument attacked by an argument in is out, every other argument is $undec$. Conversely, for every reinstatement labelling L, the corresponding extension $E(L)$ is the set of arguments labelled in by the labelling. All usual semantics can also be encoded with labellings.

We introduce some notation: L_φ is the set of labellings L such that $E(L) \in \mathcal{A}^\sigma_\varphi$. Given a set of labellings Lab, $E(Lab) = \{E(L) | L \in Lab\}$. Finally, given a system AF and a semantics σ, $Labs_\sigma(AF)$ denotes the set of labellings corresponding to the σ-extensions of AF.

Labellings, which bring richer information than extensions, can be used to define interesting pseudo-distance-based revision operators. Indeed, consider the following notion of edition pseudo-distance:

[3] A pseudo-distance d on a set S is defined as a binary relation on S which satisfies:

- $d(x, y) = d(y, x)$;
- $d(x, y) = 0$ iff $x = y$.

Definition 10. *Let m, n, o be three integers and let L_1 and L_2 be two labellings.*
An edition pseudo-distance $d_{(m,n,o)}$ *between labellings is defined as:*

$$d_{(m,n,o)}(L_1, L_2) = \sum_{a \in A} ad(L_1(a), L_2(a)),$$

where

- $ad(in, in) = ad(out, out) = ad(undec, undec) = 0$
- $ad(in, out) = ad(out, in) = m$
- $ad(in, undec) = ad(undec, in) = n$
- $ad(out, undec) = ad(undec, out) = o$

Proposition 3. *Let m, n, o be three integers. $d_{(m,n,o)}$ is a pseudo-distance.*

Interestingly, these edition pseudo-distances are not necessarily neutral or symmetric. We call neutral an edition pseudo-distance such that $ad(in, undec) + ad(undec, out) = ad(in, out)$ and symmetric a pseudo-distance such that $ad(in, undec) = ad(undec, out)$. Defining non-symmetric edition pseudo-distances is a way for instance to favor acceptance of arguments over rejection (see Example 1).

For any pseudo-distance $d_{\mathcal{L}}$ between labellings, we can define a pre-order $\leq_{AF}^{\sigma, d_{\mathcal{L}}}$ between labellings as we did it for candidates:

$L_1 \leq_{AF}^{\sigma, d_{\mathcal{L}}} L_2$ iff $d_{\mathcal{L}}(L_1, L_\sigma(AF)) \leq d_{\mathcal{L}}(L_2, L_\sigma(AF))$.

Definition 11. *Let σ be any given semantics. A* labelling-pseudo-distance-based revision operator $\star_{d^{\mathcal{L}}}$ *is any revision operator for which there exists a pseudo-distance $d^{\mathcal{L}} = d_{(m,n,o)}$ on 2^A such that for every AF and every φ, we have $Labs_\sigma(AF \star_{d^{\mathcal{L}}} \varphi) = \min(L_\varphi, \leq_{AF}^{\sigma, d_{\mathcal{L}}})$.*

The following example illustrates the impact of the chosen pseudo-distance on the revised system:

Example 1. *Let σ be the stable semantics. We revise the system AF_6 below by the formula $\varphi = (\neg d \wedge \neg e)$.*

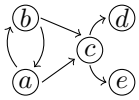


Figure 3: The system AF_6

$Ext_\sigma(AF_6) = \{\{a, d, e\}, \{b, d, e\}\}$, the associated stable labellings are $\{(a, in), (b, out), (c, out), (d, in), (e, in)\}$ and $\{(a, out), (b, in), (c, out), (d, in), (e, in)\}$. When we revise AF_6 by φ using the pseudo-distance-based operator induced by the pseudo-distance $d_{(1,9,10)}$ on labellings, the obtained result is a system with the following labellings: $\{(a, in), (b, out), (c, out), (d, out), (e, out)\}$ and $\{(a, out), (b, in), (c, out), (d, out), (e, out)\}$. When the pseudo-distance $d_{(9,1,10)}$ is used, we get $\{(a, in), (b, out), (c, out), (d, undec), (e, undec)\}$ and $\{(a, out), (b, in), (c, out), (d, undec), (e, undec)\}$ as labellings of the result systems.

If the second step of the process, i.e. the generation of the resulting argumentation systems, as expected, takes account for the labellings, the structure of the resulting graphs will be different: when the refused arguments are out, it means that there exists an attack from an accepted argument to a refused argument. When the arguments are $undec$, those attacks do not exist.
With the first pseudo-distance $d_{(1,9,10)}$, it is cheaper to change an argument from in to out than to $undec$. Such a pseudo-distance allows for choosing candidates which refuse arguments. Contrarily, the pseudo-distance $d_{(9,1,10)}$ allows for choosing candidates which accept more arguments.

Like operators based on extensions, pseudo-distance-based operators using labelling exhibit good logical properties:

Proposition 4. *Let σ be any semantics. Any labelling-pseudo-distance-based revision operator $\star_{d^{\mathcal{L}}}$ satisfies the rationality postulates **(AE1)** - **(AE6)**.*

Revision at the System Level

The operators defined in the previous section focus on the candidates that are as close as possible to the extensions of the input system. However, they do not indicate how to generate the corresponding argumentation systems, *i.e.*, the argumentation systems such that the union of their extensions coincides with the selected candidates. This task is the second step in the definition of the revision operator.

In order to achieve this task, we consider a mapping $\mathcal{AF}_\sigma$ from $2^{2^{\mathcal{A}}}$ to 2^{AFs_A}, called generation operator, that associates with any set $\mathcal{C}$ of candidates a set of argumentation systems such that $Ext_\sigma(\mathcal{AF}_\sigma(\mathcal{C})) = \mathcal{C}$.

An important point we would like to discuss is the fact that a revision operator $\star$ outputs a set of argumentation systems, and not a single argumentation system in the general case. Actually, this is a consequence of the expressiveness of the language of revision formulae we want to consider. In order to illustrate it, consider $A = \{a, b, c, d\}$, and AF_7 as represented in Figure 4.

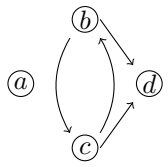
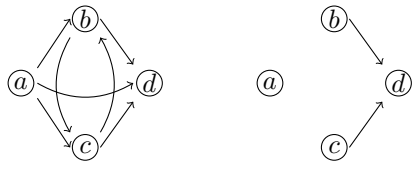


Figure 4: The system AF_7



AF_8 AF_9

Figure 5: Revision of AF_7

The extensions of AF_7 are the same ones for the stable and preferred semantics, $Ext_\sigma(AF_7) = \{\{a, b\}, \{a, c\}\}$. Let $\varphi_4 = (\neg b \vee c) \wedge (\neg c \vee b)$. Observe that b and c play symmetric roles, both in AF_7 and in φ_4. When computing the result of the revision with the revision operator based on Hamming distance between candidates, we obtain two candidates $\{a\}$ and $\{a, b, c\}$. We present in the following generation operators leading to the two corresponding argumentation systems AF_8 (corresponding to candidate $\{a\}$) and AF_9 (corresponding to candidate $\{a, b, c\}$). Clearly, choosing one of these systems would require to accept some arbitrariness given the symmetric roles of b and c.

Let us now show that for every semantics σ, generation operators $\mathcal{AF}_\sigma$ do exist.

Proposition 5. *Whatever the semantics σ, for every non-empty set $\mathcal{C}$ of candidates from 2^A, such that $\emptyset \notin \mathcal{C}$, there exists a finite set $\mathcal{S} \subseteq \mathsf{AFs}_A$ such that $\mathcal{C} = Ext_\sigma(S)$.*

For instance, for every candidate C, we can build the naive system $\langle A, R_C\rangle$ with $R_C = \{(x, y) \in A \times A | x \in C, y \notin C\}$.

So now we can define our revision operators:

Definition 12. *Given a semantics σ, a faithful assignment that matches every argumentation system to a total pre-order $\leq^\sigma_{AF}$, and a generation operator $\mathcal{AF}_\sigma$, the corresponding revision operator $\star$ is defined by:*

$$AF \star \varphi = \mathcal{AF}_\sigma(\min(\mathcal{A}^\sigma_\varphi, \leq^\sigma_{AF})).$$

One of the key results of the paper is that:

Proposition 6. *Every revision operator $\star$ defined following Definition 12 satisfies the postulates **(AE1)-(AE6)**.*

By construction, our revision operators are ensured to deal with minimality of change of arguments statuses, but not with minimality of change of the attack relation. Indeed, the rationality postulates ask for preserving as much as possible the statuses of arguments in the input system: doing so while ensuring that the revision formula is satisfied does not usually imply a minimal change of the attack relation, and vice-versa. As a matter of illustration, consider the argumentation systems AF_{10}, AF_{11}, AF_{12}, and AF_{13}.

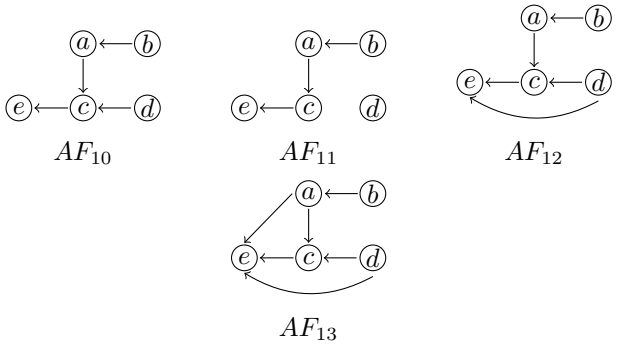


Figure 6: Minimal Change

Suppose that our goal is to reject e, that is to get a system so that e does not appear in any extension. So we consider the revision formula $\varphi_5 = \neg e$. A minimal change on the attack relation of AF_{10} leads to AF_{11} or AF_{12}: each of them differs with AF_{10} on a single attack. This contrasts with AF_{13} since the change on the attack relation required to go from AF_{10} to AF_{13} is strictly greater than the change on the attack relation required to go from AF_{10} to AF_{12}. Each of these four systems has a unique extension for the usual semantics[4]: $\{b, d, e\}$ for AF_{10}, $\{b, c, d\}$ for AF_{11}, and $\{b, d\}$ for AF_{12} and AF_{13}. Hence, the change on the statuses of arguments achieved when going from AF_{10} to AF_{12} or AF_{13} is strictly smaller than the change on the statuses of arguments achieved when going from AF_{10} to AF_{11}.

During the generation process, minimization can actually be considered in at least two ways: either minimizing change on the attack relation, or minimizing the number of output systems. In fact, these ways can be combined, either with a more important role to minimal change of the attack relation, or with a more important consideration for minimization of the cardinality.

Thus, a notion of minimal change on the attack relation can be defined through a notion of pseudo-distance dg on the attack relation. Such a pseudo-distance can be for instance the Hamming distance, given by $dg_H(AF_1, AF_2) = |(\mathcal{R}_1 \setminus \mathcal{R}_2) \cup (\mathcal{R}_2 \setminus \mathcal{R}_1)|$. The dg_H distance between two argumentation systems corresponds to the number of attacks that must be added or removed to make them identical. But we can consider more elaborated edition pseudo-distances such as those given in (Coste-Marquis et al. 2007). Each pseudo-distance dg induces a pre-order between argumentation systems, defined by $AF_1 \leq^{dg}_{AF} AF_2$ iff $dg(AF_1, AF) \leq dg(AF_2, AF)$.
We can easily extend this notion to a distance between a system AF and a set of systems AFs: $dg(AF, AFs) = \min_{AF_i \in AFs}(dg(AF, AF_i))$.

To give priority to minimal change on the attack relation, we define a generation operator that builds sets of argumentation systems which cover the candidates; then chooses the ones which minimize a function of the pseudo-distance dg; and finally retains the sets which are minimal in terms of cardinality.

Definition 13. *Given $\mathcal{C}$ a set of candidates, σ a semantics, dg a pseudo-distance between graphs and AF an argumentation system, $\mathcal{AF}^{dg,AF}_\sigma$ is defined as:*

$$\mathcal{AF}^{dg,AF}_\sigma(\mathcal{C}) = \bigcup\{AFs \in sets | card(AFs) \textit{ is minimal}\}$$

with

$$sets = \{AFs \quad | \quad Ext_\sigma(AFs) = \mathcal{C} \textit{ and } \sum_{AF_i \in AFs} dg(AF, AF_i) \textit{ is minimal}\}.$$

[4]For instance, for the complete, the preferred, the stable and the grounded semantics.

A second approach consists in giving priority to the minimality of the output cardinality. It builds first sets of systems that cover the set of candidates with a minimal number of systems, and then chooses the sets which minimize the change on the attack relation.

Definition 14. *Given $\mathcal{C}$ a set of candidates, σ a semantics, dg a pseudo-distance between graphs and AF an argumentation system, $\mathcal{AF}_\sigma^{card,AF}$ is defined by:*
$\mathcal{AF}_\sigma^{card,AF}(\mathcal{C}) = \bigcup\{AFs \in sets | \sum_{AF_i \in AFs} dg(AF, AF_i)$ *is minimal*$\}$
with

$$sets = \{AFs \quad | \quad Ext_\sigma(AFs) = \mathcal{C} \text{ and } card(AFs) \text{ is minimal}\}.$$

These two approaches are not equivalent:

Proposition 7. *The generation operators $\mathcal{AF}_\sigma^{dg,AF}$ and $\mathcal{AF}_\sigma^{card,AF}$ are distinct.*

Example 2. *Let us now give an example of revision with the approaches previously defined. The input system AF_{14} is given on Figure 7.*



Figure 7: The framework AF_{14}

Its unique stable extension is $\{a, b, c\}$. The revision formula is $\varphi_6 = (a \vee b) \wedge (\neg a \vee \neg b)$, the revision operators are $\star_{dg}$ such that

$$AF \star_{dg} \varphi = \mathcal{AF}_\sigma^{dg,AF}(\min(\mathcal{A}_\varphi^\sigma, \leq_{AF}^{\sigma,d_H}))$$

and $\star_{card}$ such that

$$AF \star_{card} \varphi = \mathcal{AF}_\sigma^{card,AF}(\min(\mathcal{A}_\varphi^\sigma, \leq_{AF}^{\sigma,d_H}))$$

both based on the Hamming distance on candidates and the Hamming distance on attack relations. Each one uses one of the previously defined generation operators.
Let us first compute the revised candidates. It is easy to find that $\{a, c\}$ and $\{b, c\}$ are the minimal models of φ_6 with respect to the Hamming distance and the stable extension of the input system.
Now we present the result for the two revision operators. When minimizing the change on attack relation, the generation step produces two argumentation systems, AF_{15} and AF_{16}, each one with a single difference from the input graph.



Figure 8: $AF_{14} \star_{dg} \varphi_6$

Contrastingly, the revision of AF_{14} with operator $\star_{card}$ gives as output a unique argumentation system AF_{17} with two differences with respect to the Hamming distance on the attack relation.

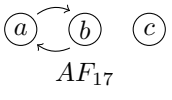


Figure 9: $AF_{14} \star_{card} \varphi_6$

While the two approaches exemplified here use the sum to aggregate the pseudo-distances, any aggregation function can be used instead. For instance, the $\min$, the $\max$, or any OWA (Ordered Weighted Average (Yager 1988)). These kind of aggregation function allows to combined distance and cardinality without giving priority to one of them. For instance, a specific ordered weighted average OWA_b is given by:

- $v(S) = (dg(AF_1, AF), \ldots, dg(AF_k, AF))$[5] with $S = \{AF_1, \ldots, AF_n\}$ and k the cardinal of the largest set (the vectors corresponding to smaller sets are normalized by adding the appropriate number of zeroes in front of the vector).
- $w_i = 2^{i-1}$
- $\text{OWA}_b(E) = \sum_{i=1}^k w_i v(E)[i]$

With the OWA_b function, a set of frameworks E_1 such that the vector of pseudo-distances is $v_1 = (1, 1, 4)$ is less preferred than a set E_2 with $v_2 = (1, 2, 3)$, because $\text{OWA}_b(E_1) = 19 > \text{OWA}_b(E_2) = 17$.
A set E_2, with three argumentation systems, can also be preferred to a set E_3 with only two systems, if those two systems are too far from the input systems. For instance, if the vector of pseudo-distances is $(1, 4)$, once normalized to $v_3 = (0, 1, 4)$, $\text{OWA}_b(E_3) = 18$, and so E_2 is still preferred in spite of its larger cardinality.

It is worthwhile to note that aggregation functions can also be used alone to define a generation operator. Given an aggregation function f, we define a pre-order $\leq_f$ such that $E_1 \leq_f E_2$ iff $f(E_1) \leq f(E_2)$. For any aggregation function on sets of argumentation systems, a generation operator is AF_σ^f given by

$$\mathcal{AF}_\sigma^f(\mathcal{C}) \in \min(\{E = \{AF_i\} | Ext_\sigma(E) = \mathcal{C}\}, \leq_f\}.$$

All proposed approaches cannot guarantee to produce a single set of argumentation systems. For instance, two sets may have the same cardinality and the same pseudo-distance from the input system with respect to the pseudo-distance dg.
The result can then be defined following one of the two options below:

- The result is defined as the union of all the sets. The reason is that these sets represent the uncertain result of the revision, so we keep all of them to avoid arbitrary choice. The main default of this method is that the size of the result is increased.
- A tie-break rule is used to select a single set of AFs. The agent is obliged to do an arbitrary choice. Note that it

[5] We suppose that the vector is sorted in increasing order.

does not prevent the revision operator from satisfying rationality postulates (because the postulates deal with extensions, not with attack relations).

Some Computational Aspects

A first interesting question concerns the size of the output of revision operators. The first step of the revision can lead to an exponential number of candidates, in terms of the number n of arguments in the input system. It is directly related to the revision formula. Given a pseudo-distance-based revision operator, the size of the output depends also on the generation operator which is used at the second step of the process. In the worst case, the number of argumentation systems which are generated is exponential in n:

Proposition 8. *If $\star$ is a revision operator based on the generation operator $\mathcal{AF}_{\sigma}^{dg,AF}$, then the size of $AF \star \varphi$ can be exponential in $|A|$.*

The complexity of the inference problem has also to be identified. Given a revision operator $\star$ for argumentation systems and a semantics σ, the *inference problem* from a revised argumentation system is the following decision problem:

- **Input:** An argumentation system AF on A, and two formulae $\varphi, \psi \in \mathcal{L}_A$.
- **Query:** Does $AF \star \varphi \mathrel{|\!\sim}_{\sigma} \psi$ hold?

Unsurprisingly, provided that σ ensures the existence of an extension for every AF, the inference problem from a revised argumentation system $AF \star \varphi$ is at least as hard as the inference problem from AF. In formal terms:

Proposition 9. *Let C be a complexity class which is closed under polynomial-time reductions. Suppose that $\star$ satisfies **(AE1)** to **(AE6)**, and that the semantics σ ensures the existence of an extension for every AF. If the inference problem from an argumentation system is C-hard, then the inference problem from a revised argumentation system is C-hard as well.*

Clearly enough, it can be the case that the inference problem from a revised argumentation system $AF \star \varphi$ is strictly harder than the inference problem from AF (unless $\mathsf{P} = \mathsf{NP}$). For instance, under the restriction when the queries ψ are restricted to arguments (or more generally, CNF formulae on A), it is easy to show that the inference problem from AF w.r.t. the grounded semantics can be solved in polynomial time. Contrastingly:

Proposition 10. *Suppose that $\star$ satisfies **(AE1)** and **(AE3)**. The inference problem from a revised argumentation system w.r.t. the grounded semantics is coNP-hard, even under the restriction when the queries ψ are restricted to CNF formulae on A.*

Our results show that the revision of argumentation systems is comparable to the revision of propositional formulae from a computational point of view. Especially, it may lead to harder computational problems: on the one hand, the revision of an argumentation system may require exponentially many systems for being represented (this is reminiscent to the non-compilability of some belief revision operators (Cadoli et al. 1999)); on the other hand, inference may also become harder (Nebel 1998).

Related Work

Some previous works have already considered the change issue for argumentation systems à la Dung. Thus, Boella, Kaci and Van der Torre, (2009a; 2009b) have studied abstraction and refinement principles. An abstraction is a reduction of the attack relation or of the set of arguments, whereas a refinement is the addition of attacks or arguments to the system. The authors focused on the study of semantics which ensure the existence of a unique extension (for instance, the grounded extension), and they formulated some principles of the form "if we do this particular change, then the extension of the result is like this". They identified some principles satisfied by the grounded semantics.

Cayrol, Dupin de Saint-Cyr and Lagasquie-Schiex (2010) studied the addition of an argument to an argumentation system. They stated some properties that can be satisfied when a change occurs in an argumentation system, and pointed out those which are satisfied (and under which conditions) when an argument (and the attacks concerning it) is added to the graph. With Bisquert, they did a similar study about the deletion of an argument (Bisquert et al. 2011).

In (Kontarinis et al. 2013), the authors studied the problem of minimal change to satisfy a goal based on arguments acceptance. The main difference between this work and our approach is that minimal change for them refers to the number of changes to perform on the attack relation, while we focus on arguments statuses.

Baumann (2012) also studied the minimal change problem in abstract argumentation. He reported some bounds on the number of modifications of the attack relation to make so as to enforce a given set of arguments. These bounds depend on the semantics and the type of change allowed.

Discussion

An abstract argumentation system à la Dung is defined by a set of arguments and an attack relation. All the works discussed in the previous section generate changes in an argumentation system by adding new arguments, while trying to minimize the modifications of the attack relation. Contrastingly, if we consider that the meaning of an argumentation system is given by the set of arguments that are accepted or rejected, then minimal change means to minimize the change on the (status of the) set of arguments.

Giving the priority on the minimization of the attack relation or on the minimization of the arguments status is really a question of considering the status of the arguments as first-class citizen of argumentation systems or only as a by-product of the graph. Whereas this last option received considerable attention these last years, there is no work except this one which concentrates on the first optic.

In addition, previous works also suppose that one can add as many arguments as one wants in order to modify the status of some arguments. In some cases this is perfectly sensible, but in other cases it is difficult to assume that such

arguments are available. Consider for instance the case of big society debates, where political parties, economists, and other specialists have already put forward all the arguments in favor of or against some decision (for example whether the state has to increase or decrease individual taxation). If a political leader wants to change the current decision, then he will need to be very brilliant in order to find an argument that has not been already pointed out by experts. More probably she will rather try to change the beliefs (or preferences) of the people on the fact that some arguments do or do not attack other ones.

Let us now mention some contexts where allowing only changes of the attack relation is very natural.

A first example of application for the revision process without adding new arguments is the reception of a unjustified but trustworthy information (which is a particular case of argument from authority). This scenario is frequent in applications of argumentation on social network debates (Gabbriellini and Torroni 2013). When an agent A initiates a debate about an argument α, if another agent B does not agree with A about α but considers that A is trustworthy, B has to revise her beliefs to accept α. In this case, agent B can change her beliefs even if agent A has not introduced a new argument in the debate. So, B has to reconsider the attacks between some arguments, but not the set of arguments itself.

A second example concerns applications of argumentation on public opinion, and is related to the society debates motivation we discussed above: suppose that an argumentation system represents the opinion of some groups of agents, where an attack between arguments exists if the majority of the group supports it. If a group leader wants to modify the statuses of arguments, then she can perform a revision of the input system. The resulting argumentation systems may help her to determine the attacks she has to focus on so as to modify the majoritarian opinion.

A third context concerns preference-based argumentation (see (Amgoud and Cayrol 2002)). In such argumentation systems some arguments attacks each other (in particular if the arguments are based on logical formulae and rebuttal, attack is symmetric), and the preference relation determines if an attack succeeds or not. So it is possible to modify the attack relation just by modifying the preferences of the agent. A similar case of revision can occur with value-based argumentation frameworks (Bench-Capon 2002): each argument is mapped to a value, and a value can be "stronger" than another. Comparison of values can lead an attack to fail. In this case, a change of values leads to a change of the (succeeding) attacks.

A final point we want to discuss is the fact that our revision methods output a set of argumentation systems instead of a single one. At the beginning of Section "Revision at the System Level", we advocated the fact that this is necessary to avoid arbitrary choices. We want to add that obtaining a set as result of a revision process is just usual in most belief revision settings. It is important to note that the canonical representation of AGM contraction/revision operators by use of relational partial-meet functions (Alchourrón, Gärdenfors, and Makinson 1985) defines the result of the process as a set of minimal theories. It turns out that the language used makes it possible to produce a single theory from this set using intersection (conjunction). But in languages where this is not possible it seems natural to keep a set as result. For instance (Fagin et al. 1986) defines *flocks* that are the set of logical databases which result of the revision of a single logical database. Flocks have also been used as sets of possible results for combination/merging operators (Baral, Kraus, and Minker 1991; Baral et al. 1992; Konieczny 2000).

Conclusion and Future Work

In this paper, we investigated the revision problem for abstract argumentation systems à la Dung. We focused on revision as minimal change of the arguments statuses. We introduced a language of revision formulae which is expressive enough for enabling the representation of complex conditions on the acceptability of arguments in the revised system. We showed how AGM belief revision postulates can be translated to the case of argumentation systems. We provided a corresponding representation theorem in terms of minimal change of the arguments status, and pointed out several pseudo-distance-based revision operators satisfying the postulates. We investigated some computational aspects of revision of argumentation systems.

We are currently encoding our revision operators by representing argumentation systems with logical constraints (in a similar way to (Besnard and Doutre 2004)), so as to be able to benefit from the power of constraint solvers to compute revised systems. At the time of writing this paper, the revision of the stable extensions of an AF by a logical constraint is encoded. Some future work is to define an encoding for other semantics, and to encode some generation operators. Informations about this work is available here: http://www.cril.fr/DynArgs/revision.html.

There are some things still to do concerning this topic. Let us quote some of them now.

Designing revision approaches which allow to combine the two kinds of revision operators of argumentation systems would be useful: the ones that allow only addition of new arguments and no change of the attack relation, and the ones that allow only changes of the attack relation and no addition of new arguments. Allowing both changes in a sensible way is an interesting question, especially in the presence of constraints (stating for instance that some arguments must remain accepted, or that some attacks cannot be changed).

Associating a minimal set of argumentation systems with a set of candidates is another important issue, not only for our revision purpose. It is related to the problem of realizability (Dunne et al. 2013), where the question is to find a (unique) argumentation system that corresponds to a set of candidates. This problem can also be studied in the case of labellings, and used for the generation of argumentation systems from a set of labellings, exploiting labelling-distance-based revision operators defined in this paper.

Finally, it is interesting to study if some restriction on the faithful assignment could allow to guarantee that the output of the revision operator is a single AF, in a similar way to (Delgrande and Peppas 2011).

Acknowledgments

We would like to thank the reviewers for their helpful comments.
This work benefited from the support of the project AMANDE ANR-13-BS02-0004 of the French National Research Agency (ANR).

References

Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change : Partial meet contraction and revision functions. *Journal of Symbolic Logic* 50:510–530.

Amgoud, L., and Cayrol, C. 2002. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence* 34(1-3):197–215.

Baral, C.; Kraus, S.; Minker, J.; and Subrahmanian, V. S. 1992. Combining knowledge bases consisting of first-order theories. *Computational Intelligence* 8(1):45–71.

Baral, C.; Kraus, S.; and Minker, J. 1991. Combining multiple knowledge bases. *IEEE Transactions on Knowledge and Data Engineering* 3(2):208–220.

Baumann, R. 2012. What does it take to enforce an argument? Minimal change in abstract argumentation. In *Proceedings of the European Conference on Artificial Intelligence (ECAI'12)*, 127–132.

Bench-Capon, T. J. M. 2002. Value-based argumentation frameworks. In *Proceedings of the 9th International Workshop on Non-Monotonic Reasoning (NMR'02)*, 443–454.

Besnard, P., and Doutre, S. 2004. Checking the acceptability of a set of arguments. In *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR'04)*, 59–64.

Bisquert, P.; Cayrol, C.; Dupin de Saint-Cyr, F.; and Lagasquie-Schiex, M.-C. 2011. Change in argumentation systems: Exploring the interest of removing an argument. In *Proceedings of the International Conference on Scalable Uncertainty Management (SUM'11)*, 275–288.

Boella, G.; Kaci, S.; and van der Torre, L. 2009a. Dynamics in argumentation with single extensions: Abstraction principles and the grounded extension. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09)*, 107–118.

Boella, G.; Kaci, S.; and van der Torre, L. 2009b. Dynamics in argumentation with single extensions: attack refinement and the grounded extension. In *Proceedings of the International Conference on Autonomous Agents and Multiagents Systems (AAMAS'09)*, 1213–1214.

Cadoli, M.; Donini, F. M.; Liberatore, P.; and Schaerf, M. 1999. The size of a revised knowledge base. *Artificial Intelligence* 115(1):25–64.

Caminada, M. 2006. On the issue of reinstatement in argumentation. In *Proceedings of the European Conference on Logics in Artificial Intelligence (JELIA'06)*, 111–123.

Cayrol, C.; Dupin de Saint-Cyr, F.; and Lagasquie-Schiex, M.-C. 2010. Change in abstract argumentation frameworks: Adding an argument. *Journal of Artificial Intelligence Research* 38:49–84.

Coste-Marquis, S.; Devred, C.; Konieczny, S.; Lagasquie-Schiex, M.-C.; and Marquis, P. 2007. On the merging of Dung's argumentation systems. *Artificial Intelligence Journal* 171:730–753.

Delgrande, J. P., and Peppas, P. 2011. Revising horn theories. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, 839–844.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n-person games. *Artificial Intelligence* 77(2):321–357.

Dunne, P. E.; Dvořák, W.; Linsbichler, T.; and Woltran, S. 2013. Characteristics of multiple viewpoints in abstract argumentation. In *Proceedings of the 4th Workshop on Dynamics of Knowledge and Belief (DKB-2013)*, 16–30.

Fagin, R.; Kuper, G. M.; Ullman, J. D.; and Vardi, M. Y. 1986. Updating logical databases. *Advances in Computing Research* 3:1–18.

Gabbriellini, S., and Torroni, P. 2013. MS Dialogues: Persuading and getting persuaded, A model of social network debates that reconciles arguments and trust. In *Proceedings of the Tenth International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2013)*.

Gärdenfors, P. 1988. *Knowledge In Flux*. Cambridge, UK: Cambridge University Press.

Katsuno, H., and Mendelzon, A. O. 1991. Propositional knowledge base revision and minimal change. *Artificial Intelligence* 52:263–294.

Konieczny, S. 2000. On the difference between merging knowledge bases and combining them. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'00)*, 135–144.

Kontarinis, D.; Bonzon, E.; Maudet, N.; Perotti, A.; van der Torre, L.; and Villata, S. 2013. Rewriting rules for the computation of goal-oriented changes in an argumentation system. In *Proceedings of the 14th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA XIV)*.

Nebel, B. 1998. How hard is it to revise a belief base? In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 3: Belief Change. Kluwer. 77–145.

Qi, G.; Liu, W.; and Bell, D. A. 2006. Knowledge base revision in description logics. In *Proceedings of the European Conference on Logics in Artificial Intelligence (JELIA'06)*, 386–398.

Yager, R. R. 1988. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics* 18:183–190.

A Dynamic Logic Framework for Abstract Argumentation

Sylvie Doutre
IRIT - Université de Toulouse

Andreas Herzig
IRIT - Université de Toulouse

Laurent Perrussel
IRIT - Université de Toulouse

Abstract

We provide a logical analysis of abstract argumentation frameworks and their dynamics. Following previous work, we express attack relation and argument status by means of propositional variables and define acceptability criteria by formulas of propositional logic. We here study the dynamics of argumentation frameworks in terms of basic operations on these propositional variables, viz. change of their truth values. We describe these operations in a uniform way within a well-known variant of Propositional Dynamic Logic PDL: the Dynamic Logic of Propositional Assignments, DL-PA. The atomic programs of DL-PA are assignments of propositional variables to truth values, and complex programs can be built by means of the connectives of sequential and nondeterministic composition and test. We start by showing that in DL-PA, the construction of extensions can be performed by a DL-PA program that is parametrized by the definition of acceptance. We then mainly focus on how the acceptance of one or more arguments can be enforced and show that this can be achieved by changing the truth values of the propositional variables describing the attack relation in a minimal way.

Introduction

Argumentation is a reasoning model based on the construction and on the evaluation of arguments. The seminal approach by Dung (Dung 1995) represents an argumentation framework $\mathcal{AF}$ as a set of abstract arguments, the structure and origin of which are left unspecified, along with an attack relationship between arguments. This paper builds on this framework. Dung and his followers have defined semantics for the evaluation of the acceptability of arguments (see (Baroni and Giacomin 2009) for a comprehensive overview). We focus in this paper on extension-based semantics, that define collectively acceptable sets of arguments, called extensions.

Dung's $\mathcal{AF}$ has already been represented in various logics, notably in propositional logic, starting with (Besnard and Doutre 2004). There, $\mathcal{AF}$ is described by means of a boolean formula in a logical language whose propositional variables represent the attacks (the *attack variables*). Furthermore, extensions of the $\mathcal{AF}$ under a given semantics σ can also be described by means of boolean formulas constraining valuations to correspond to the extensions under the semantics. This is done in an extension of the language of attack variables by variables representing argument acceptance.

Based on such a logical representation, several authors have recently investigated the dynamics of the $\mathcal{AF}$, such as (Baumann 2012; Booth et al. 2013; Bisquert et al. 2013; Coste-Marquis et al. 2013). They start by distinguishing several kinds of modification of the $\mathcal{AF}$, such as the addition or the removal of attacks, or the enforcement of the acceptability of an argument a (e.g. such that a is part of at least one extension). All these papers build on previous work in belief change, either referring to AGM theory (Alchourrón, Gärdenfors, and Makinson 1985), such as (Booth et al. 2013; Coste-Marquis et al. 2013), or to KM theory (Katsuno and Mendelzon 1992), such as (Bisquert et al. 2013). They express the modification as a logical formula describing some *goal*, i.e., a property that $\mathcal{AF}$ should satisfy: the task is to revise/update $\mathcal{AF}$ so that this formula is true.

The above papers do not provide a single framework encompassing at the same time $\mathcal{AF}$, the logical definition of the enforcement constraint and the change operations: there is usually one language for representing $\mathcal{AF}$ and another language for representing constraints, plus some definitions in the metalanguage connecting them. This has motivated us to provide a general, unified logical framework for the representation and the update of argumentation frameworks. We make use of a flexible yet simple logic: Dynamic Logic of Propositional Assignments, abbreviated DL-PA (Balbiani, Herzig, and Troquard 2013). DL-PA is a simple instantiation of Propositional Dynamic Logic PDL (Harel 1984; Harel, Kozen, and Tiuryn 2000) whose atomic programs are assignments of propositional variables to either true or false. Complex programs are built then from atomic programs by the standard PDL program operators of sequential composition, nondeterministic composition, and test. We here moreover add a less frequently considered PDL program operator, namely the converse operator. The language of DL-PA has formulas of the form $\langle\pi\rangle\varphi$ and $[\pi]\varphi$, where π is a program and φ is a formula. The former expresses that φ is true after *some* possible execution of π, and the latter expresses that φ is true after *every* possible execution of π. It is shown in (Balbiani, Herzig, and Troquard 2013) that every DL-PA formula can be reduced to an equivalent propositional formula. The reduction extends to the converse operator in a straight-

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

forward manner and provides a syntactical representation of the modified belief base.

We start by showing that the construction of extensions under a given semantics can be performed by a DL-PA program that is parametrized by the formula describing the semantics. Then we consider modifications of the attack relation and/or of the extensions. Modifications of the extensions are enforced by changing the attack relation only (addition or removal of attacks between the existing arguments). This can be achieved by changing the truth values of the attack variables. More precisely, to every input formula A describing the desired modification we associate a DL-PA program π_A implementing the update by A. We can then check whether a formula C is true in all (resp. in some) extensions of (the argumentation framework resulting from) the update of $\mathcal{AF}$ by the goal A.

The paper is organized as follows. In the next section we recall the definitions of an argumentation framework and of various semantics as well as their encoding in propositional logic. We then introduce DL-PA and show how to use it to construct extensions, before applying it to the modification of the attack relation and of the extensions. After that, we discuss several ways to extend our framework in order to capture other kinds of modifications. The last section discusses related work and concludes.[1]

Representing Argumentation Frameworks

In the present section we set the stage for our paper: we recall the definition of argumentation frameworks and provide a logical language to reason about them.

Argumentation Frameworks An abstract argumentation framework as defined by Dung in (Dung 1995) is a pair $\mathcal{AF} = (\mathcal{A}, R)$ where $\mathcal{A}$ is a finite set of abstract arguments and $R \subseteq \mathcal{A} \times \mathcal{A}$ is a binary relation on $\mathcal{A}$, called the attack relation: $(a, b) \in R$ means that a attacks b. So an argumentation framework is a directed graph with arguments as vertices and attacks as edges.

An argumentation framework $\mathcal{AF}$ can be represented in propositional logic as follows. First, a set of *attack variables* is associated to the set of arguments $\mathcal{A}$:

$$\mathtt{ATT}_{\mathcal{A}} = \{\mathsf{Att}_{a,b} \ : \ (a, b) \in \mathcal{A} \times \mathcal{A}\}$$

Let $\mathcal{L}_{\mathsf{Att}}$ be the set of all formulas that are built from attack variables. The theory of $\mathcal{AF} = (\mathcal{A}, R)$ is the boolean formula

$$\mathsf{Th}_{\mathcal{AF}} = \left(\bigwedge_{(a,b)\in R} \mathsf{Att}_{a,b}\right) \wedge \left(\bigwedge_{(a,b)\notin R} \neg\mathsf{Att}_{a,b}\right)$$

Example 1. *Consider the set of arguments $\mathcal{A}_1 = \{a, b\}$ and the two argumentation frameworks $\mathcal{AF}_1 = (\mathcal{A}_1, R_1)$ and $\mathcal{AF}_2 = (\mathcal{A}_1, R_2)$, with $R_1 = \{(a, b)\}$ and $R_2 = \{(a, b), (b, a)\}$. They are depicted in Figure 1.*

$\mathcal{AF}_1$ and $\mathcal{AF}_2$ are described by the theories

$$\mathsf{Th}_{\mathcal{AF}_1} = \mathsf{Att}_{a,b} \wedge \neg\mathsf{Att}_{b,a} \wedge \neg\mathsf{Att}_{a,a} \wedge \neg\mathsf{Att}_{b,b}$$
$$\mathsf{Th}_{\mathcal{AF}_2} = \mathsf{Att}_{a,b} \wedge \ \mathsf{Att}_{b,a} \wedge \neg\mathsf{Att}_{a,a} \wedge \neg\mathsf{Att}_{b,b}$$

[1] A longer version of the paper including proofs is available at: http://www.irit.fr/~Laurent.Perrussel/kr14long.pdf



Figure 1: $\mathcal{AF}_1$ (on the left) and $\mathcal{AF}_2$ (on the right)

Argumentation Semantics Many semantics have been defined for acceptability. Some of them are extension-based, some others are labelling-based. In this paper we only consider the first: they prevail in the literature, and moreover equivalent extension-based formulations of labelling-based semantics are available (Baroni and Giacomin 2009).

For a given $\mathcal{AF}$, a semantics identifies sets of acceptable arguments, called extensions. There may be none, one or several such extensions. In order to characterize extensions we associate to $\mathcal{A}$ a second set of propositional variables:

$$\mathtt{IN}_{\mathcal{A}} = \{\mathsf{In}_a \ : \ a \in \mathcal{A}\}$$

where In_a stands for "argument a is in the extension".

As proposed in (Besnard and Doutre 2004) and as extensively discussed in (Gabbay 2011), several semantics can be captured in propositional logic. Among such semantics, we consider at first the well-known *stable*, *admissible* and *complete* semantics.[2]

We write the propositional formulas capturing these semantics in a language that is built from the set of propositional variables $\mathbb{P} = \mathtt{ATT}_{\mathcal{A}} \cup \mathtt{IN}_{\mathcal{A}}$ by means of the boolean connectives in the usual way. Let $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ be our language of Boolean formulas.

A *stable* extension of $\mathcal{AF} = (\mathcal{A}, R)$ is a set $S \subseteq \mathcal{A}$ such that it does not exist two arguments a and b in S such that $(a, b) \in R$ (that is, S is *conflict-free*), and for each argument $b \notin S$, there exists $a \in S$ such that $(a, b) \in R$ (any argument outside the extension is attacked by the extension). This can be rephrased in propositional logic as follows (Besnard and Doutre 2004):

$$\mathsf{Stable}_{\mathcal{A}} = \bigwedge_{a\in\mathcal{A}} \left(\mathsf{In}_a \leftrightarrow \neg \bigvee_{b\in\mathcal{A}} (\mathsf{In}_b \wedge \mathsf{Att}_{b,a})\right)$$

An *admissible* set of $\mathcal{AF} = (\mathcal{A}, R)$ is a conflict-free set $S \subseteq \mathcal{A}$ that *defends* all its elements: for all $a \in S$, if there exists b such that $(b, a) \in R$, then there is some argument $c \in S$ such that $(c, b) \in R$.

$$\mathsf{Adm}_{\mathcal{A}} = \bigwedge_{a\in\mathcal{A}} \left(\mathsf{In}_a \rightarrow \bigwedge_{b\in\mathcal{A}} \left(\mathsf{Att}_{b,a} \rightarrow \left(\neg\mathsf{In}_b \wedge \bigvee_{c\in\mathcal{A}} (\mathsf{In}_c \wedge \mathsf{Att}_{c,b})\right)\right)\right)$$

A *complete* extension of $\mathcal{AF} = (\mathcal{A}, R)$ is an admissible set $S \subseteq \mathcal{A}$ that contains all the arguments it defends, that is: if an argument a is such that, for all b such that $(b, a) \in R$,

[2] Admissibility is usually considered to be a building brick of other, stronger semantics, but we consider it here as a semantics on its own, the construction of admissible sets following the same process as the construction of extensions.

there exists some $c \in S$ such that $(c, b) \in R$, then $a \in S$.

$$\mathsf{Complete}_{\mathcal{A}} = \bigwedge_{a \in \mathcal{A}} \Big(\big(\mathsf{In}_a \rightarrow \neg \bigvee_{b \in \mathcal{A}} (\mathsf{In}_b \wedge \mathsf{Att}_{b,a}) \big) \wedge \big(\mathsf{In}_a \leftrightarrow \bigwedge_{b \in \mathcal{A}} \big(\mathsf{Att}_{b,a} \rightarrow \bigvee_{c \in \mathcal{A}} (\mathsf{In}_c \wedge \mathsf{Att}_{c,b}) \big) \big) \Big)$$

The *justification state* of an argument (Baroni and Giacomin 2009) depends on the extensions it belongs to; basically, an argument is *credulously* justified (resp. *skeptically* justified) under a given semantics, if it belongs to at least one of (resp. all) the extensions under the semantics.

Extensions and Valuations Argumentation frameworks and semantics being encoded in propositional logic, computing the extensions under the semantics consists in finding the models of the formulas representing the argumentation system and the semantics. A *valuation* is a subset of the set of propositional variables $\mathbb{P} = \mathtt{ATT}_{\mathcal{A}} \cup \mathtt{IN}_{\mathcal{A}}$. The set of all valuations is $2^{\mathbb{P}}$. We use v, v_1, v_2, etc. to denote valuations. A given valuation determines the truth value of the boolean formulas of the language $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ in the usual way. The set of valuations where A is true is called *the set of models of* A or *the set of* A*-valuations* and is noted $\|A\|$.

The following proposition, adapted from (Besnard and Doutre 2004), connects extensions and valuations.

Proposition 1. *Let $\mathcal{AF} = (\mathcal{A}, R)$ be an argumentation framework and let $E \subseteq \mathcal{A}$. Consider the valuation $v_E = \{\mathsf{Att}_{a,b} : (a, b) \in R\} \cup \{\mathsf{In}_a : a \in E\}$.*

1. *E is a* stable *extension of $\mathcal{AF}$ if and only if v_E is a model of $\mathsf{Th}_{\mathcal{AF}} \wedge \mathsf{Stable}_{\mathcal{A}}$.*
2. *E is an* admissible *set of $\mathcal{AF}$ if and only if v_E is a model of $\mathsf{Th}_{\mathcal{AF}} \wedge \mathsf{Adm}_{\mathcal{A}}$.*
3. *E is an* complete *extension of $\mathcal{AF}$ if and only if v_E is a model of $\mathsf{Th}_{\mathcal{AF}} \wedge \mathsf{Complete}_{\mathcal{A}}$.*

Given a definition of a semantics σ (either stable, admissible, or complete), we denote the associated propositional formula describing the semantics by $\mathsf{Semantics}^{\sigma}$.

Corollary 1. *Let $\mathcal{AF} = (\mathcal{A}, R)$ be an argumentation framework. Let σ be a semantics. Let $C \in \mathcal{L}_{\mathsf{Att},\mathsf{In}}$ be a formula describing some property of $\mathcal{A}$. All extensions of $\mathcal{AF}$ have property C if and only if $\mathsf{Th}_{\mathcal{AF}} \wedge \mathsf{Semantics}^{\sigma}_{\mathcal{A}} \rightarrow C$ is valid.*

Example 2. *Let us consider again Example 1. $\mathcal{AF}_1$ has a single stable extension $\{a\}$, two admissible extensions $\{\}$ and $\{a\}$, and one complete extension $\{a\}$; that is:*

$$\mathsf{Th}_{\mathcal{AF}_1} \wedge \mathsf{Stable}_{\mathcal{A}_1} \rightarrow \mathsf{In}_a \wedge \neg \mathsf{In}_b$$
$$\mathsf{Th}_{\mathcal{AF}_1} \wedge \mathsf{Adm}_{\mathcal{A}_1} \rightarrow \neg \mathsf{In}_b$$
$$\mathsf{Th}_{\mathcal{AF}_1} \wedge \mathsf{Complete}_{\mathcal{A}_1} \rightarrow \mathsf{In}_a \wedge \neg \mathsf{In}_b$$

DL-PA: Dynamic Logic of Propositional Assignments

We have just seen that looking for the extensions of an argumentation framework $\mathcal{AF}$ with respect to some semantics σ amounts to looking for the models of the formulas representing $\mathcal{AF}$ and σ. We now consider a way to build these models in a logic which allows to perform operations on valuations: Dynamic Logic of Propositional Assignments DL-PA (Herzig et al. 2011; Balbiani, Herzig, and Troquard 2013). DL-PA is based on Propositional Dynamic Logic PDL (Harel 1984; Harel, Kozen, and Tiuryn 2000). Just as PDL, it has operators of sequential and nondeterministic composition of programs, test and iteration (the Kleene star), which enable reasoning about programs. The atomic programs of DL-PA are assignments of truth values to propositional variables, such as assigning true or false to p, respectively written $p{\leftarrow}\top$ and $p{\leftarrow}\bot$. The formulas of DL-PA can express how valuations are modified by programs so that after this modification stage, some property holds. In our context, they express how to build the valuations representing the extensions of $\mathcal{AF}$ w.r.t. σ. Before showing this we define syntax and semantics of DL-PA and state complexity results.

We here base our work on the star-free version of DL-PA of (Herzig et al. 2011), that we extend by the operator of converse execution of programs. We keep on calling that logic DL-PA. because the converse operator can be linearly eliminated, as we shall see.

Language The language of star-free DL-PA is defined by the following grammar:

$$\begin{array}{lll} \pi & ::= & p{\leftarrow}\top \mid p{\leftarrow}\bot \mid \varphi? \mid \pi;\pi \mid \pi \cup \pi \mid \pi^{-} \\ \varphi & ::= & p \mid \top \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi \end{array}$$

where p ranges over $\mathbb{P}$.

Key formulas are: $[\pi]\varphi$, read "after every execution of the program π formula φ holds", and $\langle\pi\rangle\varphi$, read "after some execution of the program π formula φ holds". The formula $[\pi]\varphi$ abbreviates $\neg\langle\pi\rangle\neg\varphi$.

The *atomic programs* $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ respectively make p true and make p false. The operators of sequential composition (";"), nondeterministic composition ("$\cup$") and test ("(.)?") are familiar from PDL. The operator "$(.)^{-}$" is the converse operator: formula $[\pi^{-}]\varphi$ stands for "before every execution of the program π formula φ was true" and $\langle\pi^{-}\rangle\varphi$ stands for "before some execution of the program π formula φ was true".

The *length* of a formula φ, noted $|\varphi|$, is the number of symbols used to write down φ, without "$\langle$", "$\rangle$", and parentheses. For example, $|\neg(q \vee \neg r)| = 1{+}(1{+}1{+}2) = 5$ and $|\langle q{\leftarrow}\top\rangle(q \vee r)| = 3{+}3 = 6$. The length of a program π, noted $|\pi|$, is defined in the same way. For example, $|p{\leftarrow}\bot; p?| = 3{+}1{+}2 = 6$.

Conjunction ($\wedge$), implication ($\rightarrow$) and equivalence ($\leftrightarrow$) are considered with their usual meaning. We also make use the exclusive disjunction $\varphi \oplus \psi$ which abbreviates $(\varphi \wedge \neg\psi) \vee (\neg\varphi \wedge \psi)$. Furthermore, several program abbreviations are familiar from PDL. First, skip abbreviates $\top?$ ("nothing happens"). Second, for $n \geq 0$, we define inductively π^{n} and $\pi^{\leq n}$ as follows:

$$\pi^{n} = \begin{cases} \mathsf{skip} & \text{if } n = 0 \\ \pi;\pi^{n-1} & \text{if } n \geq 1 \end{cases}$$

$$\pi^{\leq n} = \begin{cases} \mathsf{skip} & \text{if } n = 0 \\ (\mathsf{skip} \cup \pi);\pi^{\leq n-1} & \text{if } n \geq 1 \end{cases}$$

$$
\begin{aligned}
\|p{\leftarrow}\top\| &= \{(v_1, v_2) \ : \ v_2 = v_1 \cup \{p\}\} \\
\|p{\leftarrow}\bot\| &= \{(v_1, v_2) \ : \ v_2 = v_1 \setminus \{p\}\} \\
\|\varphi?\| &= \{(v, v) \ : \ v \in \|\varphi\|\} \\
\|\pi; \pi'\| &= \|\pi\| \circ \|\pi'\| \\
\|\pi \cup \pi'\| &= \|\pi\| \cup \|\pi'\| \\
\|\pi^-\| &= \|\pi\|^{-1} \\
\|p\| &= \{v \ : \ p \in v\} \\
\|\top\| &= 2^{\mathbb{P}} \\
\|\bot\| &= \{\} \\
\|\neg\varphi\| &= 2^{\mathbb{P}} \setminus \|\varphi\| \\
\|\varphi \vee \psi\| &= \|\varphi\| \cup \|\psi\| \\
\|\langle\pi\rangle\varphi\| &= \{v \ : \ \text{there is } v' \text{ s.t. } (v, v') \in \|\pi\| \text{ and } v' \in \|\varphi\|\}
\end{aligned}
$$

Table 1: Interpretation of the DL-PA connectives

Finally, we introduce assignments of literals to variables by way of the following abbreviations:

$$
\begin{aligned}
p{\leftarrow}q &= (q?; p{\leftarrow}\top) \cup (\neg q?; p{\leftarrow}\bot) \\
p{\leftarrow}\neg q &= (q?; p{\leftarrow}\bot) \cup (\neg q?; p{\leftarrow}\top)
\end{aligned}
$$

The former assigns to p the truth value of q, while the latter assigns to p the truth value of $\neg q$. So $p{\leftarrow}p$ does nothing and $p{\leftarrow}\neg p$ flips the truth value of p. Note that both abbreviations have constant length (viz. 14).

Semantics Models of DL-PA are nothing but models of classical propositional logic, i.e., subsets of the set of propositional variables $\mathbb{P}$, alias valuations. We sometimes write $v(p) = 1$ when $p \in v$ and $v(p) = 0$ when $p \notin v$. DL-PA programs are interpreted by means of a (unique) *relation between valuations*. For instance, suppose $\mathbb{P} = \{p\}$ and consider the atomic program $\pi = p{\leftarrow}\top$, and the two valuations $v_1 = \{\}$ and $v_2 = \{p\}$. The execution of π relates v_1 to v_2, and v_2 to itself. In other words, π is interpreted as $\|\pi\| = \{(v_1, v_2), (v_2, v_2)\}$.

Atomic programs of the form $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ are interpreted as update operations on valuations, and complex programs are interpreted just as in PDL by mutual recursion. Table 1 gives the interpretation of the DL-PA connectives.

Two formulas φ_1 and φ_2 are *formula equivalent* if $\|\varphi_1\| = \|\varphi_2\|$. Two programs π_1 and π_2 are *program equivalent* if $\|\pi_1\| = \|\pi_2\|$. In that case we write $\pi_1 \equiv \pi_2$. For example, the program equivalence $\pi; \mathsf{skip} \equiv \pi$ holds.

An *expression* is a formula or a program. When we say that two expressions are equivalent we mean program equivalence if we are talking about programs, and formula equivalence otherwise. Equivalence is preserved under replacement of a sub-expression by an equivalent expression (Balbiani, Herzig, and Troquard 2013).

Eliminating the Modal Operators DL-PA formulas being interpreted through classical propositional logic valuations, modal operators can be eliminated. In order to do this, we first eliminate the converse operator. Table 2 lists the relevant program equivalences (aka reduction axioms). Iterating their application we can eliminate the converse operator with only linear increase in program size. For example, for assignments of variables to literals we have:

$$
\begin{aligned}
(\varphi?)^- &\equiv \varphi? \\
(\pi_1; \pi_2)^- &\equiv \pi_2^-; \pi_1^- \\
(\pi_1 \cup \pi_2)^- &\equiv \pi_1^- \cup \pi_2^- \\
p{\leftarrow}\top^- &\equiv p?; (\mathsf{skip} \cup p{\leftarrow}\bot) \\
&\equiv p? \cup (p?; p{\leftarrow}\bot) \\
p{\leftarrow}\bot^- &\equiv \neg p?; (\mathsf{skip} \cup p{\leftarrow}\top) \\
&\equiv \neg p? \cup (\neg p?; p{\leftarrow}\top)
\end{aligned}
$$

Table 2: Reduction axioms for the converse operator

$$
p{\leftarrow}q^- \equiv \begin{cases} p{\leftarrow}p & \text{if } p = q \\ p{\leftrightarrow}q?; (p{\leftarrow}\top \cup p{\leftarrow}\bot) & \text{otherwise} \end{cases}
$$

$$
p{\leftarrow}\neg q^- \equiv \begin{cases} p{\leftarrow}\neg p & \text{if } p = q \\ p{\oplus}q?; (p{\leftarrow}\top \cup p{\leftarrow}\bot) & \text{otherwise} \end{cases}
$$

Proposition 2. *For every* DL-PA *formula φ there is an equivalent* DL-PA *formula φ' such that φ' is without $(.)^-$ and such that the length of φ' is linear in the length of φ.*

Next, we remind the following key result: for converse-free DL-PA, every formula is equivalent to a boolean formula (Herzig et al. 2011; Balbiani, Herzig, and Troquard 2013). With Proposition 2 we obtain the following result for our present language with converse.

Theorem 1. *For every* DL-PA *formula there is an equivalent boolean formula.*

For example, for different propositional variables r and p, the formula $\langle p{\leftarrow}\bot\rangle(p \vee r)$ is successively equivalent to $\langle p{\leftarrow}\bot\rangle p \vee \langle p{\leftarrow}\bot\rangle r$, to $\bot \vee r$, and to r.

Note that the boolean formula might be exponentially longer than the original formula.

Validity and Complexity Results A formula φ is DL-PA *valid* if it is equivalent to $\top$, i.e., if $\|\varphi\| = 2^{\mathbb{P}}$. It is DL-PA *satisfiable* if it is not a formula equivalent to $\bot$, i.e., if $\|\varphi\| \neq \{\}$. For example, the formulas $\langle p{\leftarrow}\top\rangle\top$ and $\langle p{\leftarrow}\top\rangle\varphi \leftrightarrow \neg\langle p{\leftarrow}\top\rangle\neg\varphi$ are DL-PA valid.

Due to Proposition 2, the complexity results of (Herzig et al. 2011) transfer to our extension with the converse operator: both model checking and satisfiability checking are PSPACE complete for our language.

Constructing Extensions with DL-PA

Let us first show how the extensions of an argumentation framework can be constructed by means of DL-PA programs. Remember that such programs modify valuations: our aim is to define a program which sets the truth values of the variables representing the acceptance of an argument so that after the execution of that program, the formula characterizing the semantics holds. We first present a few definitions that will help us to define that program.

Sorting Variables of $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ Formulas For every $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ formula A we consider two sets: the set of attack variables occurring in A, noted $\mathtt{ATT}(A)$, and the set of accept variables occurring in A, noted $\mathtt{IN}(A)$. For example, $\mathtt{ATT}(\neg\mathsf{Att}_{a,b} \vee \mathsf{In}_a) = \{\mathsf{Att}_{a,b}\}$ and $\mathtt{IN}(\neg\mathsf{Att}_{a,b} \vee \mathsf{In}_a) = \{\mathsf{In}_a\}$. It is the modifications of $\mathtt{ATT}(A)$ that will be minimized in order to construct the extensions, while the modifications of the set $\mathtt{IN}(A)$ will not be minimized: given the truth values of the attack variables, the semantics determines the truth values for the accept variables (or rather, the possible combinations of accept variables, because there may be several extensions).

Evaluating the Difference between Valuations The symmetric difference between two valuations $v_1, v_2 \subseteq \mathbb{P}$ is the number of variables whose truth values differ, formally defined as

$$v_1 \dot{-} v_2 = (v_1 \setminus v_2) \cup (v_2 \setminus v_1)$$

So $v_1 \dot{-} v_2$ is the set of all those p such that $v_1(p) \neq v_2(p)$. The *Hamming distance* between two valuations $v_1, v_2 \subseteq \mathbb{P}$ is the cardinality of their symmetric difference: $v_1 \dot{-} v_2$ is the *number* of all those p such that $v_1(p) \neq v_2(p)$.

Example 3. *Suppose two valuations: $\{p, q\}$ and $\{q, r, s\}$. Their symmetric difference is $\{p, r, s\}$ and their Hamming distance is* 3.

Changing Variables through DL-PA Programs Let us consider the two following DL-PA programs that can change the truth value of one or more propositional variables in the finite subset $P = \{p_1, \ldots, p_n\}$ of the set of propositional variables $\mathbb{P} = \mathtt{ATT}_{\mathcal{A}} \cup \mathtt{IN}_{\mathcal{A}}$.

$$\mathsf{vary}(P) = (p_1 \leftarrow \top \cup p_1 \leftarrow \bot); \cdots; (p_n \leftarrow \top \cup p_n \leftarrow \bot)$$
$$\mathsf{flip1}(P) = p_1 \leftarrow \neg p_1 \cup \cdots \cup p_n \leftarrow \neg p_n$$

The intuitive meaning of program $\mathsf{vary}(P)$ is "the program is composed of a sequence of n steps and at each step i, the value of variable p_i is nondeterministically set to either to true or false"; one may also say that $\mathsf{vary}(P)$ forgets the values of the variables in P. Second, program $\mathsf{flip1}(P)$ should be read as "one variable from P is randomly chosen and is nondeterministically set to either true or false". Observe that the programs $\mathsf{vary}(P)$ and $(\mathsf{flip1}(P))^{\leq n}$ are equivalent.

Note that the length of each program is linear in the cardinality of P.

The following lemma characterises the behaviour of the two programs in semantic terms.

Lemma 1. *The following hold:*

1. $(v_1, v_2) \in \|\mathsf{vary}(P)\|$ *iff* $v_1 \dot{-} v_2 \subseteq P$
2. $(v_1, v_2) \in \|\mathsf{flip1}(P)\|$ *iff* $v_1 \dot{-} v_2 = \{p_k\}$ *for some* $p_k \in P$

Given an $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ formula A, $\mathbb{P}_A$ is defined as the set of variables from $\mathbb{P}$ occurring in A. The first item of Lemma 1 hence tells us that the program $\mathsf{vary}(\mathbb{P}_A); A?$ accesses all relevant A-valuations (where 'relevant' means that only those A-valuations are accessed which keep constant the truth values of variables not occurring in A).

Example 4. *Consider the formula $A = \mathsf{In}_a \wedge \neg\mathsf{In}_b$. The set of variables of A is $\mathbb{P}_A = \{\mathsf{In}_a, \mathsf{In}_b\}$. The models of A can be built by, first, varying the truth values of the variables of the formula ($\mathsf{vary}(\mathbb{P}_A)$), and second, by testing which of the resulting valuations satisfy the formula, and by keeping them only ($A?$). $\mathsf{vary}(\mathbb{P}_A)$ is here:*

$$(\mathsf{In}_a \leftarrow \top \cup \mathsf{In}_a \leftarrow \bot); (\mathsf{In}_b \leftarrow \top \cup \mathsf{In}_b \leftarrow \bot)$$

Its execution relates each valuation for $\mathbb{P}_A$, i.e., each of $\{\}$, $\{\mathsf{In}_a\}$, $\{\mathsf{In}_b\}$, and $\{\mathsf{In}_a, \mathsf{In}_b\}$, to every other, including itself. Among the couples of valuations making up that relation, there is only one relating the model of A (that is, $\{\mathsf{In}_a\}$) to itself. So the execution of the test $A?$ only keeps this pair in the relation. So the pair $(\{\mathsf{In}_a\}, \{\mathsf{In}_a\})$ is a possible execution of the program $\mathsf{vary}(\mathbb{P}_A); A?$.

Our example illustrates that the satisfiability of a boolean formula A can be expressed in DL-PA as the existence of valuation that is accessible by the program $\mathsf{vary}(\mathbb{P}_A); A?$, i.e., by means of the formula

$$\langle \mathsf{vary}(\mathbb{P}_A); A? \rangle \top$$

The formula says that there is a way of changing the truth value of some variables of A so that A is true afterwards.

Lemma 2. *Let A be a propositional formula. A is (propositionnally) satisfiable iff $\langle \mathsf{vary}(\mathbb{P}_A); A? \rangle \top$ is DL-PA valid.*

Defining Extensions in DL-PA Let us now show how a DL-PA program can express in a concise and natural way the construction of extensions of an argumentation framework $\mathcal{AF}$ under a semantics σ. The program $\mathsf{makeExt}^{\sigma}_{\mathcal{A}}$ below constructs the valuations representing extensions w.r.t. a semantics σ, of argumentation frameworks over the set of arguments $\mathcal{A}$. Remember that the set of attack variables is $\mathtt{ATT}_{\mathcal{A}}$, that the set of accept variables is $\mathtt{IN}_{\mathcal{A}}$, and that σ is either stable, complete, or admissible acceptability semantics. We define the DL-PA program $\mathsf{makeExt}^{\sigma}_{\mathcal{A}}$ as follows:

$$\mathsf{makeExt}^{\sigma}_{\mathcal{A}} = \mathsf{vary}(\mathtt{IN}_{\mathcal{A}}); \mathsf{Semantics}^{\sigma}_{\mathcal{A}}?$$

The following result says that if v_1 completely describes the attack relation then the set of accept variables of the valuations v_2 such that $(v_1, v_2) \in \|\mathsf{makeExt}^{\sigma}_{\mathcal{A}}\|$ characterises the extensions of $\mathcal{AF}$ w.r.t. semantics σ. It is a consequence of Corollary 1 and Lemma 2.

Lemma 3. *Let $\mathcal{AF}$ be an argumentation framework. Let σ be either the stable, complete or admissible semantics. Let v_1 be a model of $\mathsf{Th}_{\mathcal{AF}}$. Then $(v_1, v_2) \in \|\mathsf{makeExt}^{\sigma}_{\mathcal{A}}\|$ iff v_2 is a model of $\mathsf{Th}_{\mathcal{AF}} \wedge \mathsf{Semantics}^{\sigma}_{\mathcal{A}}$.*

Lemma 3 entails that we can construct all the extensions of a given argumentation system w.r.t. some semantics by means of a DL-PA program.

Proposition 3. *Let $\mathcal{AF} = (\mathcal{A}, R)$ be an argumentation framework and let C be a boolean formula. The boolean formula $\mathsf{Th}_{\mathcal{AF}} \wedge \mathsf{Semantics}^{\sigma}_{\mathcal{A}} \rightarrow C$ is propositionally valid if and only if the DL-PA formula $\mathsf{Th}_{\mathcal{AF}} \rightarrow [\mathsf{makeExt}^{\sigma}_{\mathcal{A}}]C$ is valid. Moreover, the equivalence*

$$\mathsf{Th}_{\mathcal{AF}} \wedge \mathsf{Semantics}^{\sigma}_{\mathcal{A}} \leftrightarrow \langle (\mathsf{makeExt}^{\sigma}_{\mathcal{A}})^{-} \rangle \mathsf{Th}_{\mathcal{AF}}$$

is DL-PA valid.

According to Proposition 3, given a description of the attack relation, program $\mathsf{makeExt}^{\sigma}_{\mathcal{A}}$ can construct valuations where the formula describing the semantics is true, and so in a way such that only acceptance variables are changed, while the attack variables do not change. Note that the equivalence takes care of situations where there is no extension. Let the $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ formula A be a goal, i.e., a boolean formula that should be satisfied by the extensions of some $\mathcal{AF}$. By means of the program $\mathsf{makeExt}^{\sigma}_{\mathcal{A}}$ one can check what has been called *σ-consistency* in (Coste-Marquis et al. 2013): whether there exists an argumentation framework $\mathcal{AF}$ such that some or every extension w.r.t. σ satisfies A. Adopting the standard terms for characterizing a justification state, we distinguish the notions of credulous and skeptical consistency.

Definition 1 (σ-consistency). *Let $\mathcal{AF} = (\mathcal{A}, R)$ be an argumentation framework, let σ be either the stable, admissible, or complete semantics. The $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ formula A is* σ-credulously-consistent for $\mathcal{A}$ *iff* $\langle\mathsf{makeExt}^{\sigma}_{\mathcal{A}}\rangle A$ *is satisfiable; A is* σ-skeptically-consistent for $\mathcal{A}$ *iff* $[\mathsf{makeExt}^{\sigma}_{\mathcal{A}}]A$ *is satisfiable.*

Example 5. *Consider the set of arguments $\mathcal{A}_2 = \{a, b\}$ and the goal $A = \neg\mathsf{In}_a \wedge \neg\mathsf{In}_b \wedge \neg\mathsf{Att}_{a,a}$. Then $\langle\mathsf{makeExt}^{\sigma}_{\mathcal{A}}\rangle A$ is unsatisfiable; in particular, it is false in every valuation for $\mathsf{Th}_{\mathcal{AF}_2}$ of Example 1. Therefore the formula A is stable-credulous inconsistent for $\mathcal{A}_2$.*

To sum it up, we have shown how extensions can be constructed through programs modifying propositional variables. In the next section we show how to modify an argumentation framework and its extensions.

Modifying an Argumentation Framework

We start by discussing several kinds of modifications of an argumentation framework $\mathcal{AF}$ and then focus on the most involved family of operations: modifications enforcing a goal either in all extensions or in some extension of $\mathcal{AF}$.

Generally speaking, when we want to incorporate a new piece of information A into a belief base B then we want to build a new belief base that retains as much as possible the initial beliefs of B while entailing A. Neither AGM theory (Alchourrón, Gärdenfors, and Makinson 1985) nor KM (Katsuno and Mendelzon 1992) theory provide a single, concrete belief change operation: they rather constrain the set of 'reasonable' belief change operations by means of a set of postulates. Several concrete belief change operations satisfying the AGM or KM postulates have been defined in the literature. Among the most prominent are Winslett's 'possible models approach' (PMA) (Winslett 1988; 1990), Forbus's update operation (Forbus 1989), Winslett's standard semantics (WSS) (Winslett 1995), and Dalal's revision operation (Dalal 1988); see (Herzig and Rifi 1999; Lang 2007) for an overview. According to Katsuno and Mendelzon's distinction between update and revision operations (Katsuno and Mendelzon 1992), the first three are update operations, usually written $\diamond$, while Dalal's is a revision operation, usually written $*$.

Kinds of Change Several kinds of modifications of a given argumentation framework $\mathcal{AF} = (\mathcal{A}, R)$ have been distinguished (Cayrol, Bannay, and Lagasquie-Schiex 2010):

1. One may modify the set of arguments $\mathcal{A}$ by adding or removing an argument.
2. One may modify the attack relation R by adding or removing an edge between two arguments.
3. One may wish to modify the extensions of $\mathcal{AF}$.

The first kind of operation, viz. adding an argument to the set of arguments $\mathcal{A}$ or removing it, requires more linguistic resources than our language $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ provides; we will sketch a way of implementing it in DL-PA in an augmented language in the end of the paper.

Addition and removal of an attack edge between two arguments $a, b \in \mathcal{A}$ are rather simple operations: we just add or subtract (a, b) from R. In our logical representation, these operations correspond to making propositional variables true or false: the addition of (a, b) to R can be implemented by the update of $\mathsf{Th}_{\mathcal{AF}}$ by $\mathsf{Att}_{a,b}$, and the removal of (a, b) from R can be implemented by the update of $\mathsf{Th}_{\mathcal{AF}}$ by $\neg\mathsf{Att}_{a,b}$.[3] In DL-PA, the update of $\mathsf{Th}_{\mathcal{AF}}$ by $\mathsf{Att}_{a,b}$ can be implemented by the program $\mathsf{Att}_{a,b}\leftarrow\top$, and the update by $\neg\mathsf{Att}_{a,b}$ by $\mathsf{Att}_{a,b}\leftarrow\bot$. Indeed, we have:

$$\mathsf{Th}_{\mathcal{AF}} \diamond \mathsf{Att}_{a,b} = \langle \mathsf{Att}_{a,b}\leftarrow\top^{-}\rangle \mathsf{Th}_{\mathcal{AF}}$$
$$\mathsf{Th}_{\mathcal{AF}} \diamond \neg\mathsf{Att}_{a,b} = \langle \mathsf{Att}_{a,b}\leftarrow\bot^{-}\rangle \mathsf{Th}_{\mathcal{AF}}$$

where $\diamond$ is any of the above belief change operations. Once $\mathsf{Th}_{\mathcal{AF}}$ has been updated by some $\mathsf{Att}_{a,b}$ or $\neg\mathsf{Att}_{a,b}$, the extensions of the resulting framework can be obtained by conjoining the result with $\mathsf{Semantics}^{\sigma}_{\mathcal{A}}$.

Adding an argument to extensions or removing it is more involved because it has to be achieved indirectly, by changing the underlying attack relation of $\mathcal{AF}$ (or by changing the set of arguments, but as we said, we disregard this option for the time being). Moreover, when an argumentation framework has several extensions, one may wish to change the justification status of a, so that it be skeptically justified (that is, added in all the extensions), or credulously justified (added in at least one extension), or credulously justified but not skeptically justified (added in at least one extension, but removed from at least one), or even not credulously justified (removed from all extensions).

While the distinction between update and revision was irrelevant for the modification of the attack relation, it gets important when it comes to the modification of extensions: which operation should we apply? We dedicate the next section to the problem of modifying an extension.

[3] We note that the question whether this is an update or a revision does not really play a role here because we are in a simple situation: as far as the language $\mathcal{L}_{\mathsf{Att}}$ is concerned, the formula $\mathsf{Th}_{\mathcal{AF}}$ is complete (it has a unique model), and the update of complete theories by literals leads to the same result for all the belief change operations that we have mentioned, WSS, PMA, Forbus, and Dalal, be they update or revision operations.

The Modification of Extensions In order to discuss which change operation is the most appropriate, let us consider again the above argumentation framework $\mathcal{AF}_2 = (\mathcal{A}_2, R_2)$ and the stable semantics. $\mathcal{AF}_2$ has two stable extensions $E_a = \{a\}$ and $E_b = \{b\}$. We have seen that the associated boolean formula $\mathsf{Th}_{\mathcal{AF}_2} \wedge \mathsf{Stable}_{\mathcal{A}_2}$ has two $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ models:

$$v_a = \{\mathsf{Att}_{a,b}, \mathsf{Att}_{b,a}, \mathsf{In}_a\}$$
$$v_b = \{\mathsf{Att}_{a,b}, \mathsf{Att}_{b,a}, \mathsf{In}_b\}$$

Suppose we wish to modify $\mathcal{AF}_2$ such that a is in none of its stable extensions. What we have to do is to adapt the attack relation of $\mathcal{AF}_2$ in a way that is minimal and that guarantees that a does not occur in any of its extensions, while guaranteeing that a is rejected. We view minimality as minimality of the number of modifications of the attack relation; in contrast, the current extensions may be modified in a non-minimal way. This policy agrees with (Coste-Marquis et al. 2013), where the minimization of the changes on the attack relation is considered to be secondary.

What we do is to modify the models of $\mathsf{Th}_{\mathcal{AF}_2} \wedge \mathsf{Stable}_{\mathcal{A}_2}$ one by one so that In_a holds: we minimally modify the attack variables such that the result is stable. As we can see, *enforcing* a constraint is similar to update and we will hereafter commit to this reading.

To sum up, enforcing A consists in minimally updating the attack relation of an initial argumentation framework $\mathcal{AF} = (\mathcal{A}, R)$ with respect to some goal A in a way such that A holds in all of the extensions of the resulting $\mathcal{AF}' = (\mathcal{A}, R')$. Another perspective is however possible, too, where enforcing consists in minimally modifying the attack relation such that A holds in *some* of the extensions of the resulting $\mathcal{AF}' = (\mathcal{A}, R')$. This operation seems to be closer in spirit to a revision operation. So 'enforcement' can be understood in two different ways, leading to two different definitions. In both cases, there is a key difference with standard revision and update operations: in classical belief change, output is limited to one belief set, while here, several $\mathcal{AF}'$ may be produced by the enforcement operations. Let $\mathcal{AF} \diamond^{\mathsf{Cred},\sigma} A$ denote the *credulous* enforcement of property A in $\mathcal{AF}$ w.r.t. semantics σ, and let $\mathcal{AF} \diamond^{\mathsf{Skep},\sigma} A$ denote the *skeptical* enforcement of property A in $\mathcal{AF}$. We consider that both $\diamond^{\mathsf{Cred},\sigma}$ and $\diamond^{\mathsf{Skep},\sigma}$ are operations mapping an argumentation framework and an $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ formula to a set of argumentation frameworks.

In line with the definitions of classical revision and update operations, we now define some postulates that 'reasonable' operations $\diamond$ should satisfy. They are mainly inspired by (Coste-Marquis et al. 2013).

We have shown above that both attack relations and σ-consistency can be captured by DL-PA formulas. As moreover the goal A is an $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ formula, too, our postulates can be formulated in terms of DL-PA formulas.

Definition 2 (enforcement postulates). *Let $\diamond$ be an operation mapping an argumentation framework and an $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ formula to a set of argumentation frameworks. Let σ be either the stable, admissible, or complete semantics. The operation $\diamond$ is a* credulous enforcement *operation iff it satisfies the following postulates:*

E1.C *For all $\mathcal{AF}' \in \mathcal{AF} \diamond^{\sigma} A$, $\models \mathsf{Th}_{\mathcal{AF}'} \rightarrow \langle \mathsf{makeExt}^{\sigma}_{\mathcal{A}} \rangle A$.*

E2.C *If $\models \mathsf{Th}_{\mathcal{AF}} \wedge \mathsf{Semantics}^{\sigma}_{\mathcal{A}} \rightarrow \langle \mathsf{makeExt}^{\sigma}_{\mathcal{A}} \rangle A$ then $\mathcal{AF} \diamond^{\sigma} A = \{\mathcal{AF}\}$.*

E3 *If $\models A_1 \leftrightarrow A_2$ then for every $\mathcal{AF}_1 \in \mathcal{AF} \diamond^{\sigma} A_1$ there exists $\mathcal{AF}_2 \in \mathcal{AF} \diamond^{\sigma} A_2$ such that $\models \mathsf{Th}_{\mathcal{AF}_1} \leftrightarrow \mathsf{Th}_{\mathcal{AF}_2}$.*

The operation $\diamond$ is said to be a skeptical enforcement *operation iff it satisfies postulate **E3** plus the following:*

E1.S *For all $\mathcal{AF}' \in \mathcal{AF} \diamond^{\sigma} A$, $\models \mathsf{Th}_{\mathcal{AF}'} \rightarrow [\mathsf{makeExt}^{\sigma}_{\mathcal{A}}] A$.*

E2.S *If $\models \mathsf{Th}_{\mathcal{AF}} \wedge \mathsf{Semantics}^{\sigma}_{\mathcal{A}} \rightarrow [\mathsf{makeExt}^{\sigma}_{\mathcal{A}}] A$ then $\mathcal{AF} \diamond^{\sigma} A = \{\mathcal{AF}\}$.*

The postulates **E1.C** and **E1.S** say that success in required for credulous and skeptical enforcement. **E2** represents a minimal change principle: it states that if A already holds then $\mathcal{AF}$ is unchanged. Postulate **E3** is the postulate of syntax independence: enforcement should be based on the content of a goal and not on its syntax.

Additional postulates may be formulated; see (Bisquert et al. 2013; Coste-Marquis et al. 2013) for more details. A key difference is that we do not consider postulates based on the expansion operation. The main reason is that this operation is actually useless: first, if an attack has to be changed then expansion cannot be used because, as we have seen above, $\mathsf{Th}_{\mathcal{AF}}$ is complete for $\mathcal{L}_{\mathsf{Att}}$. Now consider that an argument has to be enforced in a credulous way. We face two cases: (i) either the argument is already credulously acceptable and there is no reason to change the argumentation framework, or (ii) the argument is not credulously acceptable. Then as all possible extensions are considered, some attacks must be changed so that new extensions can be constructed (and A will then hold). The same reasoning can be made for skeptical acceptance.

Let us now show how DL-PA helps us to reason about enforcement.

Expressing Extension Modification in DL-PA

The aim of this section is to show how modification programs can be defined. These programs should satisfy the enforcement postulates that we have defined. Second, they should only minimally change the attack relations of an initial argumentation framework. As previously mentioned, enforcement is close to update, and we will root our programs in the update operations that guarantee minimal change. One of the most popular is Forbus's update operation (Forbus 1989), where minimal change involves counting how many variables have been changed. Our solution is strongly connected to it.

The Hamming Distance Predicate Let us define the DL-PA formula $\mathsf{h}(A, P, \geq m)$, where A is an $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ formula, P is a set of propositional variables, and $m \geq 0$ is an integer:

$$\mathsf{h}(A, P, \geq m) = \begin{cases} \top & \text{if } m = 0 \\ \neg\langle(\mathsf{flip1}(P))^{\leq m-1}\rangle A & \text{if } m \geq 1 \end{cases}$$

We call $\mathsf{h}(A, P, \geq m)$ the *Hamming distance predicate w.r.t. the set of variables P*: it is true at a valuation v exactly when the A-valuations v' that are closest to v in the sense of the Hamming distance differ in at least m variables from

v, where the computation of the distance only considers variables from the set P, while the other variables in $\mathbb{P} \setminus P$ keep their value.

Proposition 4. *Let v a valuation, A a boolean formula, P some set of propositional variables, and $m \geq 0$. Then*

1. *$v \in \|\mathsf{h}(A, P, \geq m)\|$ iff the A-valuations that are closest to v w.r.t. P have Hamming distance at least m, i.e., iff $\mathsf{card}(v \dot{-} v') \geq m$ for every $v' \in \|A\|$ s.t. $v \dot{-} v' \subseteq P$.*
2. *$(v, v') \in \|\mathsf{h}(A, P, \geq m)?; (\mathsf{flip1}(P))^m; A?\|$ iff v' is one of the A-valuations that is closest to v w.r.t. P, i.e., iff $v' \in \|A\|$ and $\mathsf{card}(v \dot{-} v') = m$ for every $v' \in \|A\|$ s.t. $v \dot{-} v' \subseteq P$.*

It follows from the first item of Proposition 4 that when P equals $\mathbb{P}_A$ then $v \in \|\mathsf{h}(A, \mathbb{P}_A, \geq m)\|$ iff the A-valuations that are closest to v have Hamming distance at least m, i.e., iff $\mathsf{card}(v \dot{-} v') \geq m$ for every $v' \in \|A\|$. This is used in Forbus's udpate operation.

Forbus's Update Operation Forbus's update operation is based on minimization of the Hamming distance. First, the Forbus update of a valuation v by A is the set of those A-valuations whose Hamming distance to v is minimal. Second, the Forbus update of a belief base B by A collects the Forbus updates of all B-valuations by A.

The following DL-PA program performs Forbus's update operation:

$$\mathsf{forbus}(A) = \bigcup_{m \leq \mathsf{card}(\mathbb{P}_A)} \Big(\mathsf{h}(A, \mathbb{P}_A, \geq m)?; (\mathsf{flip1}(\mathbb{P}_A))^m\Big); A?$$

The program nondeterministically chooses an integer m, checks if the Hamming distance to A-valuations is at least m and flips m of the variables of A. Finally, the test A? only succeeds for A-valuations.

Proposition 5. *The formula C is true after the Forbus update of B by A if and only if $B \rightarrow [\mathsf{forbus}(A)]C$ is* DL-PA *valid.*

Argumentation Framework Update The update of argumentation frameworks has some specificities: first, we are going to modify only the attack variables while leaving the accept variables unchanged; second, the target formula is not going to be a boolean formula, but a formula saying that A will be the case *after building extensions*. Let us define the program $\mathsf{credEnf}(A)$ modifying $\mathcal{AF}$ w.r.t. some semantics σ such that the boolean formula $A \in \mathcal{L}_{\mathsf{Att},\mathsf{In}}$ becomes true in some σ-extensions:

$$\mathsf{credEnf}(A) = \bigcup_{m \leq \mathsf{card}(\mathtt{ATT}_{\mathcal{A}})} \Big(\mathsf{h}(\langle \mathsf{makeExt}^{\sigma}_{\mathcal{A}} \rangle A, \mathtt{ATT}_{\mathcal{A}}, \geq m)?; (\mathsf{flip1}(\mathtt{ATT}_{\mathcal{A}}))^m\Big); \langle \mathsf{makeExt}^{\sigma}_{\mathcal{A}} \rangle A?$$

The following program enforces a constraint in a skeptical way.

$$\mathsf{skepEnf}(A) = \bigcup_{m \leq \mathsf{card}(\mathtt{ATT}_{\mathcal{A}})} \Big(\mathsf{h}([\mathsf{makeExt}^{\sigma}_{\mathcal{A}}] A, \mathtt{ATT}_{\mathcal{A}}, \geq m)?; (\mathsf{flip1}(\mathtt{ATT}_{\mathcal{A}}))^m\Big); [\mathsf{makeExt}^{\sigma}_{\mathcal{A}}] A?$$

The length of these two programs is polynomial in the cardinality of $\mathcal{A}$. (The cardinality of the set $\mathtt{ATT}_{\mathcal{A}}$ is quadratic in that of $\mathcal{A}$ and the length of $(\mathsf{flip1}(\mathtt{ATT}_{\mathcal{A}}))^m$ is quadratic in that of $\mathcal{A}$.)

To check whether C is true in all extensions of $\mathcal{AF}$ modified by A (for $A, C \in \mathcal{L}_{\mathsf{Att},\mathsf{In}}$) can then be done by checking whether the DL-PA formula $\mathsf{Th}_{\mathcal{AF}} \rightarrow [\mathsf{skepEnf}(A)]C$ is valid. In particular we have the following proposition which states that if there is a chance to get A by modifying the attack variables then the enforcement will succeed:

Proposition 6.

$$\models [\mathsf{credEnf}(A)]\langle \mathsf{makeExt}^{\sigma}_{\mathcal{A}} \rangle A$$
$$\models [\mathsf{skepEnf}(A)][\mathsf{makeExt}^{\sigma}_{\mathcal{A}}] A$$

The second key property is that an argumentation framework is unchanged if the goal already holds.

Proposition 7. *For every goal A that is credulously justified, the credulous update program does not change anything:*

$$\models (\mathsf{Th}_{\mathcal{AF}} \wedge \langle \mathsf{makeExt}^{\sigma}_{\mathcal{A}} \rangle A \wedge C) \rightarrow [\mathsf{credEnf}(A)]C$$

For every goal A that is skeptically justified, the skeptical update program does not change anything:

$$\models (\mathsf{Th}_{\mathcal{AF}} \wedge [\mathsf{makeExt}^{\sigma}_{\mathcal{A}}] A \wedge C) \rightarrow [\mathsf{skepEnf}(A)]C$$

The modified argumentation frameworks can be extracted from the formulas $\langle \mathsf{credEnf}(A)^{-} \rangle \mathsf{Th}_{\mathcal{AF}}$ and $\langle \mathsf{skepEnf}(A)^{-} \rangle \mathsf{Th}_{\mathcal{AF}}$ representing it in DL-PA (or rather, their reduction) by forgetting the accept variables, as proposed in (Coste-Marquis et al. 2013). This operation can be implemented in our framework by the program $\mathsf{vary}(\mathtt{IN}_{\mathcal{A}})$.

Definition 3 ($\diamond^{\sigma,\mathsf{enf}}$)**.** *Let σ be either the stable, admissible, or complete semantics. Let* enf *be either the* skepEnf *or the* credEnf *program. Let $\diamond^{\sigma,\mathsf{enf}}$ be an operation mapping an argumentation framework and an $\mathcal{L}_{\mathsf{Att},\mathsf{In}}$ formula to a set of argumentation frameworks. The update of $\mathcal{AF}$ by A under σ and* enf *is*

$$\mathcal{AF} \diamond^{\sigma,\mathsf{enf}} A = \{(\mathcal{A}, R_v) \ : \ v \in \|\langle (\mathsf{enf}(A))^{-} \rangle \mathsf{Th}_{\mathcal{AF}}\|\}$$

where R_v is the attack relation extracted from v, defined as $R_v = \{(a, b) \ : \ \mathsf{Att}_{a,b} \in v\}$.

The two preceding propositions guarantee that our enforcement operations satisfy the postulates.

Theorem 2. *Operation $\diamond^{\sigma,\mathsf{credEnf}}$ satisfies **E1.C** and **E2.C**. Operation $\diamond^{\sigma,\mathsf{skepEnf}}$ satisfies **E1.S** and **E2.S**. Both operations $\diamond^{\sigma,\mathsf{credEnf}}$ and $\diamond^{\sigma,\mathsf{skepEnf}}$ satisfy **E3**.*

This result is actually not a surprise, given that our tool for enforcement is a variant of Forbus's update operation.

Example 6. *Let us take up the argumentation framework $\mathcal{AF}_2$ of Example 1. Remember that $\mathcal{AF}_2 = (\mathcal{A}_1, R_2)$, with $\mathcal{A}_1 = \{a, b\}$ and $R_2 = \{(a, b), (b, a)\}$ and that $\mathsf{Th}_{\mathcal{AF}_2} = \mathsf{Att}_{a,b} \wedge \mathsf{Att}_{b,a}$. Let us consider the stable semantics and suppose our goal is to enforce that a is always acceptable (skeptical enforcement). We disregard self-attacks for the sake of simplicity. The nondeterministic part $\bigcup_{m \leq \mathsf{card}(\mathtt{ATT}_{\mathcal{A}})}(\ldots)$ of the program $\mathsf{SkepEnf}(\mathsf{In}_{\mathsf{a}})$ changes one variable from*

$\mathsf{Th}_{\mathcal{AF}_2}$, *either* $\mathsf{Att}_{a,b}$ *or* $\mathsf{Att}_{b,a}$. *This corresponds to two candidate extensions: one where a only attacks b and one where b only attacks a. Only the former case gives valuations where a is always acceptable. Hence:*

$$\mathcal{AF} \diamond^{\mathsf{Stable,skepEnf}} \mathsf{In}_a = \{(\mathcal{A}_1, \{(a, b)\})\}$$

Going Further

We have illustrated how DL-PA offers a fruitful framework for representing argumentation framework and reasoning about them. We now sketch several ways of extending our account.

Other Argumentation Semantics Our exposition focused on the stable, admissible, and complete semantics. This can however be generalized to *every* semantics Σ whose extensions can be characterized by a propositional formula $\Sigma_{\mathcal{A}}$ built from the set of attack variables $\mathtt{ATT}_{\mathcal{A}}$ and the set of accept variables $\mathtt{IN}_{\mathcal{A}}$. In the previous sections we have given DL-PA programs modifying valuations (that represent argumentation frameworks) in a minimal way and producing (representations of) extensions. Therefore all minimization-based semantics can be handled in an elegant way, as well as all maximization-based semantics.

For instance, the preferred semantics (that captures the maximal, w.r.t. set-inclusion, admissible sets) can be handled as follows:

$$\mathsf{Semantics}^{pref}_{\mathcal{A}} =$$

$$\mathsf{Adm}_{\mathcal{A}} \wedge [\mathsf{MakeTrueSome}(\mathtt{IN}_{\mathcal{A}}); \mathsf{h}(\mathsf{Adm}_{\mathcal{A}}, \geq 1)?]\neg\mathsf{Adm}_{\mathcal{A}}$$

where $\mathsf{MakeTrueSome}(P) = (\neg p_1?; (p_1 \leftarrow \bot \cup p_1 \leftarrow \top)); \ldots; (\neg p_n?; (p_n \leftarrow \bot \cup p_n \leftarrow \top))$ makes true some of the elements of P. $[\mathsf{MakeTrueSome}(\mathtt{IN}_{\mathcal{A}})]$ evaluates all possible combinations of changes in the arguments out of the admissible set. $[\mathsf{MakeTrueSome}(\mathtt{IN}_{\mathcal{A}}); \mathsf{h}(\mathsf{Adm}_{\mathcal{A}}, \geq 1)?]\neg\mathsf{Adm}_{\mathcal{A}}$ tries to enforce all non-empty subsets of arguments currently out of the admissible set. Maximality of the admissible set is thus guaranteed. The resulting encoding is polynomial time.

Other Update Semantics The modification programs that we have defined in the present paper basically applies Forbus's update operation to the attack variables. It is possible to use other such operations, such as Winslett's Possible Models Approach (PMA) (Winslett 1988), or revision operations such as Dalal's (Dalal 1988).

Other Kinds of Change Up to now we did not cover addition and removal of some argument from an argumentation framework. Let us sketch how this could be done in a fairly straightforward way in the framework of DL-PA. First of all, we have to add a further ingredient to argumentation frameworks: let us consider triples $\mathcal{AF} = (\mathcal{A}_0, \mathcal{A}, R)$ where $\mathcal{A} \subseteq R \times R$ as before, and moreover $\mathcal{A}_0$ is a (possibly infinite) set such that $\mathcal{A} \subseteq \mathcal{A}_0$. We think of $\mathcal{A}_0$ as the background set of all possible arguments, while $\mathcal{A}$ is the set of all arguments that are currently under consideration. We have to modify the logical language $\mathcal{L}_{\mathsf{Att,In}}$ accordingly. First, we suppose that there are propositional variables $\mathsf{Att}_{a,b}$ and In_a for every a, b in the background set $\mathcal{A}_0$. Second, we add propositional variables Cons_a, one per argument $a \in \mathcal{A}_0$, where Cons_a reads "a is considered". Let us denote the resulting language by $\mathcal{L}_{\mathsf{Att,In,Cons}}$. Then the theory of $\mathcal{AF} = (\mathcal{A}_0, \mathcal{A}, R)$ is the boolean formula

$$\mathsf{Th}_{\mathcal{AF}} = \left(\bigwedge_{(a,b)\in R} \mathsf{Att}_{a,b}\right) \wedge \left(\bigwedge_{(a,b)\notin R} \neg\mathsf{Att}_{a,b}\right) \wedge \left(\bigwedge_{a\in\mathcal{A}} \mathsf{Cons}_a\right)$$

So $\mathsf{Th}_{\mathcal{AF}}$ does not say anything about the arguments that are currently not under consideration: the models of $\mathsf{Th}_{\mathcal{AF}}$ have arbitrary truth values for $\mathsf{Att}_{a,b}$, $\mathsf{Att}_{b,a}$, In_a, and Cons_a as soon as $a \notin \mathcal{A}$.

The semantic definitions have to be adapted, too, and should only quantify over arguments in $\mathcal{A}$, and not over those in $\mathcal{A}_0$. For stable semantics we get:

$$\mathsf{Stable}_{\mathcal{A}} = \bigwedge_{a\in\mathcal{A}} \left(\mathsf{Cons}_a \rightarrow \left(\mathsf{In}_a \leftrightarrow \neg \bigvee_{b\in\mathcal{A}} (\mathsf{In}_b \wedge \mathsf{Att}_{b,a})\right)\right)$$

In this setting, the mere addition or deletion of an argument a can be achieved straightforwardly, viz. by changing the status of a from 'disregarded' to 'considered' by means of the assignments $\mathsf{Cons}_a \leftarrow \top$ and $\mathsf{Cons}_a \leftarrow \bot$. Subsequently, the attack relation can be modified as sketched above: one has to first delete all the $\mathsf{Att}_{a,b}$ and $\mathsf{Att}_{b,a}$ such that $b \in \mathcal{A}$, and then add the attack relation as desired.

Conclusion

The main result of this paper is the encoding of argumentation frameworks and their dynamics in DL-PA. More precisely, our contribution is threefold.

First, as long as argument acceptability can be expressed in propositional logic, finding acceptable arguments and enforcing acceptability can be done in DL-PA. Other logical frameworks allow capturing and computing argument acceptability (see (Charwat et al. 2013) for an overview), but few of them allow capturing and computing acceptability change as well.

Second, as DL-PA formulas can be rewritten as propositional logic formulas, the result of the modification of an argumentation framework is described by a propositional formula from the models of which one may retrieve the modified argumentation frameworks. Our proposal is hence more 'operational' than those of (Bisquert et al. 2013; Coste-Marquis et al. 2013) because we use a formal logic encompassing the representation of change operations. Moreover, we consider not only credulous acceptability changes, as most of the current approaches do (Baumann 2012), but skeptical acceptability change as well.

Third, our framework takes advantage of the complexity results for DL-PA: both model checking and satisfiability checking are in PSPACE. A closer look at the formulas expressing the modifications shows that the alternation of quantifications is bounded, which typically leads to complexity bounds at the second level of the polynomial hierarchy.

The proposed DL-PA encodings may be related to QBF encodings for argumentation (Arieli and Caminada 2013). As QBF has the same complexity as star-free DL-PA, all we do in DL-PA must be polynomially encodable into QBF.

However, the availability of assignment programs makes a difference: update programs such as $\mathsf{forbus}(A)$ and extension construction programs such as $\mathsf{makeExt}^{\sigma}_{\mathcal{A}}$ capture things in a more general, flexible, and natural way than a QBF encoding.

To conclude, the richness of our framework makes it expandable to other kinds of changes, other update semantics, and other argumentation semantics beyond those that are detailed in the present paper. We plan to investigate this research avenue in future work.

Acknowledgements

This work benefited from the support of the AMANDE project (ANR-13-BS02-0004) of the French National Research Agency (ANR). We would like to thank Gabriele Kern-Isberner for triggering our work when DL-PA was presented at the Dagstuhl seminar "Belief Change and Argumentation in Multi-Agent Scenarios" in June 2013.

References

Alchourrón, C.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *J. of Symbolic Logic* 50:510–530.

Arieli, O., and Caminada, M. W. 2013. A QBF-based formalization of abstract argumentation semantics. *Journal of Applied Logic* 11(2):229–252.

Balbiani, P.; Herzig, A.; and Troquard, N. 2013. Dynamic logic of propositional assignments: a well-behaved variant of PDL. In *Logic in Computer Science (LICS)*. IEEE.

Baroni, P., and Giacomin, M. 2009. Semantics of abstract argument systems. In Simari, G., and Rahwan, I., eds., *Argumentation in Artificial Intelligence*. Springer US. 25–44.

Baumann, R. 2012. What does it take to enforce an argument? minimal change in abstract argumentation. In Raedt, L. D.; Bessière, C.; Dubois, D.; Doherty, P.; Frasconi, P.; Heintz, F.; and Lucas, P. J. F., eds., *ECAI 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, 127–132. IOS Press.

Besnard, P., and Doutre, S. 2004. Checking the acceptability of a set of arguments. In *10th Int. Workshop on Non-Monotonic Reasoning (NMR'2004)*, 59–64.

Bisquert, P.; Cayrol, C.; Bannay, F.; and Lagasquie-Schiex, M.-C. 2013. Enforcement in Argumentation is a kind of Update. In Liu, W.; Subrahmanian, V.; and Wijsen, J., eds., *International Conference on Scalable Uncertainty Management (SUM), Washington DC, USA*, number 8078 in LNAI, 30–43. Springer Verlag.

Booth, R.; Kaci, S.; Rienstra, T.; and van der Torre, L. 2013. A logical theory about dynamics in abstract argumentation. In Liu, W.; Subrahmanian, V. S.; and Wijsen, J., eds., *SUM*, volume 8078 of *Lecture Notes in Computer Science*, 148–161. Springer.

Cayrol, C.; Bannay, F.; and Lagasquie-Schiex, M.-C. 2010. Change in Abstract Argumentation Frameworks: Adding an Argument. *Journal of Artificial Intelligence Research* 38:49–84.

Charwat, G.; Dvorák, W.; Gaggl, S. A.; Wallner, J. P.; and Woltran, S. 2013. Implementing abstract argumentation – a survey. Technical report, DBAI-TR-2013-82.

Coste-Marquis, S.; Konieczny, S.; Mailly, J.-G.; and Marquis, P. 2013. On the revision of argumentation systems: Minimal change of arguments status. In *TAFA'13*.

Dalal, M. 1988. Investigations into a theory of knowledge base revision: preliminary report. In *Proc. 7th Conf. on Artificial Intelligence (AAAI'88)*, 475–479.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.* 77(2):321–357.

Forbus, K. D. 1989. Introducing actions into qualitative simulation. In Sridharan, N. S., ed., *Proc. 11th Int. Joint Conf. on Artificial Intelligence (IJCAI'89)*, 1273–1278. Morgan Kaufmann Publishers.

Gabbay, D. M. 2011. Dung's argumentation is essentially equivalent to classical propositional logic with the peirce–quine dagger. *Logica Universalis* 5(2):255–318.

Harel, D.; Kozen, D.; and Tiuryn, J. 2000. *Dynamic Logic*. MIT Press.

Harel, D. 1984. Dynamic logic. In Gabbay, D. M., and Günthner, F., eds., *Handbook of Philosophical Logic*, volume II. D. Reidel, Dordrecht. 497–604.

Herzig, A., and Rifi, O. 1999. Propositional belief base update and minimal change. *Artificial Intelligence Journal* 115(1):107–138.

Herzig, A.; Lorini, E.; Moisan, F.; and Troquard, N. 2011. A dynamic logic of normative systems. In Walsh, T., ed., *International Joint Conference on Artificial Intelligence (IJCAI)*, 228–233. Barcelona: IJCAI/AAAI. Erratum at http://www.irit.fr/~Andreas.Herzig/P/Ijcai11.html.

Katsuno, H., and Mendelzon, A. O. 1992. On the difference between updating a knowledge base and revising it. In Gärdenfors, P., ed., *Belief revision*. Cambridge University Press. 183–203. (preliminary version in Allen, J.A., Fikes, R., and Sandewall, E., eds., Principles of Knowledge Representation and Reasoning: Proc. 2nd Int. Conf., pages 387–394. Morgan Kaufmann Publishers, 1991).

Lang, J. 2007. Belief update revisited. In *Proc. of the 10th International Joint Conference on Artificial Intelligence (IJCAI'07)*, 2517–2522.

Winslett, M.-A. 1988. Reasoning about action using a possible models approach. In *Proc. 7th Conf. on Artificial Intelligence (AAAI'88)*, 89–93.

Winslett, M.-A. 1990. *Updating Logical Databases*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Winslett, M.-A. 1995. Updating logical databases. In Gabbay, D. M.; Hogger, C. J.; and Robinson, J. A., eds., *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4. Oxford University Press. 133–174.

Characteristics of Multiple Viewpoints in Abstract Argumentation*

Paul E. Dunne
University of Liverpool
Department of Computer Science, UK
P.E.Dunne@liverpool.ac.uk

Wolfgang Dvořák
University of Vienna
Faculty of Computer Science, Austria
wolfgang.dvorak@univie.ac.at

Thomas Linsbichler
Vienna University of Technology
Institute of Information Systems, Austria
linsbich@dbai.tuwien.ac.at

Stefan Woltran
Vienna University of Technology
Institute of Information Systems, Austria
woltran@dbai.tuwien.ac.at

Abstract

The study of extension-based semantics within the seminal abstract argumentation model of Dung has largely focused on definitional, algorithmic and complexity issues. In contrast, matters relating to comparisons of representational limits, in particular, the extent to which given collections of extensions are expressible within the formalism, have been under-developed. As such, little is known concerning conditions under which a candidate set of subsets of arguments are "realistic" in the sense that they correspond to the extensions of some argumentation framework AF for a semantics of interest. In this paper we present a formal basis for examining extension-based semantics in terms of the sets of extensions that these may express within a single AF. We provide a number of characterization theorems which guarantee the existence of AFs whose set of extensions satisfy specific conditions and derive preliminary complexity results for decision problems that require such characterizations.

1 Introduction

The last 15 years have seen an enormous effort to design, compare, and implement different semantics for Dung's abstract argumentation frameworks (Dung 1995), AFs for short. As a result of this extensive study argumentation is now a significant topic within AI research (Bench-Capon and Dunne 2007; Rahwan and Simari 2009). Surprisingly, systematic comparison of AF capabilities respecting *multiple* extensions, and thus their power in modelling multiple viewpoints within a single framework has, so far, been neglected. Nevertheless, an understanding of which extensions can, in principle, coexist when a framework is evaluated with respect to a semantics of interest not only clarifies the "strength" of that semantics but also is a crucial issue in several applications.

In this work, we address this issue by studying the *signatures* of argumentation semantics. A semantics σ maps each argumentation framework F to a set of extensions (i.e. a set of sets of arguments). The signature of σ is defined by

$$\Sigma_\sigma = \{\sigma(F) \mid F \text{ is an AF}\},$$

and gives the collection of all possible sets of extensions an AF can possess under semantics σ. We shall focus on several important semantics namely naive, preferred, semi-stable, stage, stable, and complete semantics (Dung 1995; Verheij 1996; Caminada, Carnielli, and Dunne 2012) and aim at finding simple criteria to decide whether a set $\mathbb{S}$ is contained in Σ_σ. For instance, we will show that each $\mathbb{S} \in \Sigma_{pref}$ satisfies the condition that for each pair of distinct sets A and B from $\mathbb{S}$ there is at least one $a \in A$ and $b \in B$ such that a, b do not occur together in any set in $\mathbb{S}$. Thus, for instance, $\mathbb{S} = \{\{a, b\}, \{c, d\}, \{a, c\}, \{b, d\}\}$ is part of the signature Σ_{pref} while neither $\mathbb{S} \cup \{\{a, d\}\}$ nor $\mathbb{S} \cup \{\{b, c\}\}$ are. This fact can be exploited in a search procedure for enumerating preferred extensions: assume for a given AF F, the three extensions from $\mathbb{S}$ have already been calculated as preferred extensions of F. The procedure can now restrict the search space to find further extensions of F (if they exist) to sets with at least one argument different from a, b, c, and d.

The problem we study here is also essential in many other aspects.

First, our results are important for constructing AFs. Indeed, knowing whether a set $\mathbb{S}$ is contained in Σ_σ is a necessary condition which should be checked before actually looking for an AF F which *realizes* $\mathbb{S}$ under σ, i.e. $\sigma(F) = \mathbb{S}$. This is of high importance when dynamic aspects of argumentation are considered (Falappa et al. 2011). As an example, suppose a framework F possesses as its σ-extensions a set $\mathbb{S}$ and one asks for an adaptation of the framework F such that its σ-extensions are given by $\mathbb{S} \cup \{E\}$, i.e. one extension is to be added. The addition of E to $\mathbb{S}$ may, for instance, be desired by some agent on the grounds that E contains some subset of arguments which it wishes to be collectively accepted by other agents: no extension in $\mathbb{S}$, however, provides support for the subset of interest to be considered justifiable. Furthermore the agent wishing to add E is reluctant to jeopardize the chance of this happening if the modified AF is such that some existing element of $\mathbb{S}$ ceases to be an extension: such an outcome being likely to prejudice other agents against agreeing to changes which admit E. Before considering the adapted framework's structure, it is obviously crucial to know whether an appropriate framework exists at all, i.e. whether $S \cup \{E\} \in \Sigma_\sigma$. In a recent paper on revision of AFs (Coste-Marquis et al. 2013), the authors circumvent this issue by allowing revision to result in a set of AFs such that

*This research has been supported by the Austrian Science Fund (FWF) through projects I1102 and P25521.
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

the union of their extensions yields the desired outcome. Our results provide exact conditions under which their approach admits a single AF as an outcome of a given revision.

Second, our work adds to the comparison of semantics (see, e.g., (Baroni and Giacomin 2007)) by means of different properties. So far such properties have largely focused on aspects of *single* extensions $S \in \mathbb{S}$ rather than on *sets* of such. An obvious exception being incomparability (the sets in $\mathbb{S}$ are not proper subsets of each other; this property is also known as I-maximality); as we will see, however, all of the standard semantics impose additional (yet distinct) requirements on $\mathbb{S}$ in order for containment in the signature to hold. Furthermore, our results add to the growing body of work considering *generic* treatments of argumentation semantics, that is with respect to shared properties rather than from the perspective of distinguishing features. Previous such examinations of abstract argumentation semantics include the "principle-based evaluation" methods proposed by Baroni and Giacomin (2007) and the concept of "rationality postulates" from (Caminada and Amgoud 2007). For example, we show that most semantics σ are closed under intersection of extensions (more formally, for all AFs F_1, F_2, there exists an AF F such that $\sigma(F) = \sigma(F_1) \cap \sigma(F_2)$, whenever $\sigma(F_1) \cap \sigma(F_2) \neq \emptyset$).

Related work includes studies on enforcing (Baumann 2012; Baumann and Brewka 2010) certain outcomes, where the task is to modify AFs in such a way that desired arguments become acceptable. However, the issue of multiple extensions is not covered. In fact, the work which is closest to our investigations are studies of intertranslatability issues (Dvořák and Spanring 2012; Dvořák and Woltran 2011), where signatures of semantics are put in relation to each other. More precisely, if there is an exact translation such that θ-extensions of the transformed AF coincide with the σ-extensions of the original AF, then θ is at least as expressive as σ, that is $\Sigma_\sigma \subseteq \Sigma_\theta$ in our terms. These results, however, tell us little about the *contents* of Σ_σ and Σ_θ.

To summarize, the main contributions of our work are:

• We first identify necessary conditions any set of extensions under a given semantics σ satisfies, i.e. we identify sets of extensions Σ^+_σ with $\Sigma_\sigma \subseteq \Sigma^+_\sigma$. This not only informs the exact characteristics for the signature of σ, but also determines those sets of extensions that are impossible to be jointly expressed with one AF.

• Then, we provide sufficient conditions for a set of extensions to be realizable under a given semantics σ. For any such set $\mathbb{S}$, we present constructions of canonical frameworks having $\mathbb{S}$ as their σ-extensions. In other words we identify sets of extensions Σ^-_σ with $\Sigma^-_\sigma \subseteq \Sigma_\sigma$. For all semantics σ under consideration (with the exception of complete extensions) our results hold with $\Sigma^+_\sigma = \Sigma^-_\sigma$. We, thus, obtain exact characterizations of the signatures Σ_σ.

• We apply our results to study the aforementioned property of closure under intersection (of extension-sets). Moreover, we identify limits of disagreement the different semantics face. While the capabilities of semantics differ in cases involving expressing more than two extensions, it turns out that the maximal number of extensions (over n arguments) that can be captured is equal for many semantics.

• One particular application of our results is the problem of recasting, i.e. to decide if the σ-extensions of a given AF can be expressed via another semantics θ. We give some preliminary complexity results of the recasting problem, which go up to Π^P_2-completeness. In contrast, we show that the problem of checking if $\mathbb{S} \in \Sigma_\sigma$ for a given $\mathbb{S}$ is decidable in polynomial time for all exactly characterized semantics.

2 Preliminaries

We assume a countably infinite domain $\mathfrak{A}$ of arguments. An *argumentation framework* (AF) is a pair $F = (A, R)$ where $A \subseteq \mathfrak{A}$ is non-empty and finite, and $R \subseteq A \times A$ is the attack relation. The collection of all AFs is given as $AF_{\mathfrak{A}}$. We write $a \mapsto_R b$ for $(a, b) \in R$ and $S \mapsto_R a$ (resp. $a \mapsto_R S$) if $\exists s \in S$ such that $s \mapsto_R a$ (resp. $a \mapsto_R s$). We drop subscript R in $\mapsto_R$ if there is no ambiguity. For $S \subseteq A$, the *range* of S (wrt. R), denoted S^+_R, is the set $S \cup \{b \mid S \mapsto_R b\}$.

Given $F = (A, R)$, an argument $a \in A$ is *defended* (in F) by a set $S \subseteq A$ if for each $b \in A$, such that $b \mapsto_R a$, also $S \mapsto_R b$. A set T of arguments is defended (in F) by S if each $a \in T$ is defended by S (in F). The following result is in the spirit of Dung's fundamental lemma and is used later.

Lemma 1. *Given an AF $F = (A, R)$ and two sets of arguments $S, T \subseteq A$. If S defends itself in F and T defends itself in F, then $S \cup T$ defends itself in F.*

Given an AF $F = (A, R)$, a set $S \subseteq A$ is *conflict-free* (in F), if there are no arguments $a, b \in S$, such that $(a, b) \in R$. We denote the set of all conflict-free sets in F as $cf(F)$. $S \in cf(F)$ is called *admissible* (in F) if S defends itself. We denote the set of admissible sets in F as $adm(F)$.

The semantics we focus in this work are the naive, stable, preferred, complete, stage, and semi-stable extensions. Given $F = (A, R)$ they are defined as subsets of $cf(F)$:

- $S \in naive(F)$, if there is no $T \in cf(F)$ with $T \supset S$
- $S \in stb(F)$, if $S \mapsto a$ for all $a \in A \setminus S$
- $S \in pref(F)$, if $S \in adm(F)$ and $\nexists T \in adm(F)$ s.t. $T \supset S$
- $S \in com(F)$, if $S \in adm(F)$ and $a \in S$ for all $a \in A$ defended by S
- $S \in stage(F)$, if $\nexists T \in cf(F)$ with $T^+_R \supset S^+_R$
- $S \in sem(F)$, if $S \in adm(F)$ and $\nexists T \in adm(F)$ s.t. $T^+_R \supset S^+_R$

The objects of our interest are the *signatures* of a semantics.

Definition 1. The signature Σ_σ of a semantics σ is defined as

$$\Sigma_\sigma = \{\sigma(F) \mid F \in AF_{\mathfrak{A}}\}.$$

For characterizing the signatures of the semantics of our interest we will make frequent use of the following concepts.

Definition 2. Given $\mathbb{S} \subseteq 2^{\mathfrak{A}}$, $Arg_{\mathbb{S}}$ denotes $\bigcup_{S \in \mathbb{S}} S$ and $Pairs_{\mathbb{S}}$ denotes $\{(a, b) \mid \exists S \in \mathbb{S} : \{a, b\} \subseteq S\}$. $\mathbb{S}$ is called an *extension-set* (over $\mathfrak{A}$) if $Arg_{\mathbb{S}}$ is finite. While $|\mathbb{S}|$ denotes the number of extensions in $\mathbb{S}$, $\|\mathbb{S}\|$ stands for $|Arg_{\mathbb{S}}|$.

Observe that for any $a \in Arg_{\mathbb{S}}$, $(a, a) \in Pairs_{\mathbb{S}}$ holds for all extension-sets $\mathbb{S}$; also note that for all considered semantics σ each element $\mathbb{S} \in \Sigma_\sigma$ is an extension-set.

3 Properties of Argumentation Semantics

In this section, we provide necessary conditions for an extension-set $\mathbb{S}$ to be in the signature Σ_σ. To this end, we have to find common properties for $\sigma(F)$ which hold for any AF F. We do so by abstracting away from the syntactical structure of a given framework.

Some properties are rather easy to see: *cf*(F) satisfies the property that for any $S \in$ *cf*(F), also $S' \in$ *cf*(F) for any $S' \subseteq S$. Likewise, for all $\sigma \in \{naive, stb, stage, pref, sem\}$ it obviously holds that $\sigma(F)$ is incomparable for any AF. We define these two properties formally.

Definition 3. Let $\mathbb{S} \subseteq 2^{\mathfrak{A}}$. The *downward-closure*, *dcl*$(\mathbb{S})$, of $\mathbb{S}$ is given by $\{S' \subseteq S \mid S \in \mathbb{S}\}$. We call $\mathbb{S}$ *downward-closed* if $\mathbb{S} =$ *dcl*$(\mathbb{S})$ and *incomparable* if for each $S, S' \in \mathbb{S}$, $S \subseteq S'$ implies $S = S'$.

However, all considered semantics σ enjoy additional properties. Just consider the incomparable extension-set $\mathbb{S} = \{\{a, b\}, \{a, c\}, \{b, c\}\}$. Now suppose there exists an AF F with $\sigma(F) = \mathbb{S}$. Then F must not contain attacks between a and b, a and c, and respectively b and c. But then $\sigma(F)$ typically contains $\{a, b, c\}$.

Stable, Stage, Naive and Conflict-Free Sets

There are several ways to define the required property which excludes sets like $\mathbb{S}$ from above. To characterize stable, stage and naive extensions, we start with a rather strong restriction making use of the concept of *Pairs*$_\mathbb{S}$ (in Section 6, we will discuss an alternative yet equivalent definition in terms of a ternary operator on $\mathbb{S}$). For the other semantics, we will use a strictly weaker criterion.

Definition 4. An extension-set $\mathbb{S} \subseteq 2^{\mathfrak{A}}$ is *tight* if for all $S \in \mathbb{S}$ and $a \in$ *Arg*$_\mathbb{S}$ it holds that if $S \cup \{a\} \notin \mathbb{S}$ then there exists an $s \in S$ such that $(a, s) \notin$ *Pairs*$_\mathbb{S}$.

For incomparable $\mathbb{S}$, the definition of $\mathbb{S}$ being tight reduces to check whether for all $S \in \mathbb{S}$ and $a \in$ *Arg*$_\mathbb{S} \setminus S$ there is an $s \in S$ such that $(a, s) \notin$ *Pairs*$_\mathbb{S}$. The idea behind the notion of being tight is that if an argument a does not occur in some extension S there must be a reason for that. The most simple reason one can think of is that there is a conflict between a and some $s \in S$, i.e. a and s do not occur jointly in any extension-set of $\mathbb{S}$ or, in other words, $(a, s) \notin$ *Pairs*$_\mathbb{S}$. Indeed, the set $\mathbb{S} = \{\{a, b\}, \{a, c\}, \{b, c\}\}$ from above is not tight, since there is no reason to, for instance, exclude c from extension $\{a, b\}$ ((a, c) and (b, c) are both contained in *Pairs*$_\mathbb{S}$). On the other hand, the set $\{\{a, b\}, \{a, c\}, \{b, d\}, \{c, d\}\}$ is easily checked to be tight.

Before stating our first results, we examine certain properties of tight sets which we exploit later.

Lemma 2. *For a tight extension-set* $\mathbb{S} \subseteq 2^{\mathfrak{A}}$ *it holds that*

1. *the* $\subseteq$*-maximal elements in* $\mathbb{S}$ *form a tight set, and*
2. *if* $\mathbb{S}$ *is incomparable then each* $\mathbb{S}' \subseteq \mathbb{S}$ *is tight.*

Proof. Consider some tight extension-set $\mathbb{S}$. (1) For each $S \in \mathbb{S}$ and $a \in$ *Arg*$_\mathbb{S}$ it holds that $S \cup \{a\} \notin \mathbb{S}$ if $S \cup \{a\} \notin \mathbb{S}'$, where $\mathbb{S}'$ is the set of $\subseteq$-maximal elements in $\mathbb{S}$. Hence, as *Pairs*$_\mathbb{S} =$ *Pairs*$_{\mathbb{S}'}$ and $\mathbb{S}' \subseteq \mathbb{S}$, $\mathbb{S}'$ is tight too. (2) Let $\mathbb{S}$ be incomparable, and consider some $\mathbb{S}' \subseteq \mathbb{S}$. Then

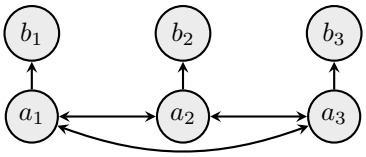


Figure 1: Argumentation framework F used in Example 1.

Pairs$_{\mathbb{S}'} \subseteq$ *Pairs*$_\mathbb{S}$ and, as $\mathbb{S}$ is incomparable, $S \cup \{a\} \notin \mathbb{S}$ iff $S \cup \{a\} \notin \mathbb{S}'$ for all $S \in \mathbb{S}$, $a \in$ *Arg*$_\mathbb{S}$. Thus $\mathbb{S}'$ is tight. □

Proposition 1. *For each* AF $F = (A, R)$*,*

1. *cf*(F) *is non-empty, downward-closed and tight;*
2. *naive*(F) *is non-empty, incomparable and its downward-closure is tight;*
3. *stage*(F) *is non-empty, incomparable and tight;*
4. *stb*(F) *is incomparable and tight.*

Proof. The properties of being non-empty and incomparable are clear. Likewise, it is easy to see that *cf*$(F) =$ *dcl*(*cf*(F)), i.e. *cf*(F) is downward-closed. To show that *cf*(F) is tight, let $S \in$ *cf*(F) and $a \in$ *Arg*$_{cf(F)}$, such that $S \cup \{a\} \notin$ *cf*(F). It follows that $S \neq \emptyset$ (otherwise, $\{a\} \notin$ *cf*(F) and since we know *cf*$(F) =$ *dcl*(*cf*(F)), this would yield $a \notin$ *Arg*$_{cf(F)}$). Moreover there exists an argument $s \in S$ such that $s \mapsto a$ or $a \mapsto s$. Then $\{a, s\} \notin$ *cf*(F) and since *cf*$(F) =$ *dcl*(*cf*(F)), $\{a, s\} \not\subseteq T$ for any $T \in$ *cf*(F). It follows that $(a, s) \notin$ *Pairs*$_{cf(F)}$. Since this applies to all S, a with $S \in$ *cf*(F) and $a \notin S$, tightness of *cf*(F) follows. Next, observe that *dcl*(*naive*(F)) = *cf*(F), by definition of the semantics. Thus *dcl*(*naive*(F)) is tight, and by Lemma 2, *naive*(F) is tight. With the same lemma, we get that every $\mathbb{S} \subseteq$ *naive*(F) is tight. By the well-known fact that *stb*$(F) \subseteq$ *stage*$(F) \subseteq$ *naive*(F), the result follows. □

Example 1. For the AF F in Figure 1, we have $\mathbb{S} =$ *stb*$(F) =$ *stage*$(F) = \{\{a_1, b_2, b_3\}, \{a_2, b_1, b_3\}, \{a_3, b_1, b_2\}\}$ and *naive*$(F) = \mathbb{S} \cup \{\{b_1, b_2, b_3\}\}$. $\mathbb{S}$ is indeed tight. Take, for instance, $E = \{a_1, b_2, b_3\}$; then for each argument t not in E (i.e. b_1, a_2, a_3) there is an argument $s \in E$ such $(s, t) \notin$ *Pairs*$_\mathbb{S}$. In this particular case a_1 plays this role for each t. The other two extensions behave in a symmetric way. However, *dcl*$(\mathbb{S})$ is not tight. In fact, $\{b_2, b_3\} \in$ *dcl*$(\mathbb{S})$ and now for b_1, $\{b_1, b_2, b_3\} \notin$ *dcl*$(\mathbb{S})$ but (b_1, b_2) and (b_1, b_3) are contained in *Pairs*$_{dcl(\mathbb{S})} =$ *Pairs*$_\mathbb{S}$. Proposition 1 now gives evidence that an AF G with *naive*$(G) = \mathbb{S}$ cannot exist.

Preferred, Semi-Stable and Admissible Sets

The following property checks whether each pair of extensions in an extension-set $\mathbb{S}$ has at least one mutual conflict in case its union is not contained in $\mathbb{S}$.

Definition 5. A set $\mathbb{S} \subseteq 2^{\mathfrak{A}}$ is called *adm-closed* if for each $A, B \in \mathbb{S}$ the following holds: if $(a, b) \in$ *Pairs*$_\mathbb{S}$ for each $a, b \in A \cup B$, then also $A \cup B \in \mathbb{S}$.

Note that for $a, b \in A$ (likewise $a, b \in B$), $(a, b) \in$ *Pairs*$_\mathbb{S}$ holds by definition. Thus the property of adm-closed is determined by arguments $a \in A \setminus B$, $b \in B \setminus A$, for $A, B \in \mathbb{S}$.

Lemma 3. *Every tight extension-set is also adm-closed.*

Proof. Consider some tight extension-set $\mathbb{S}$ and let $A, B \in \mathbb{S}$. We need to show that in case $(a, b) \in \mathit{Pairs}_{\mathbb{S}}$ for each $a, b \in A \cup B$, then also $A \cup B \in \mathbb{S}$. Let $B = \{b_1, \ldots, b_n\}$. As $\mathbb{S}$ is tight and $(a, b_1) \in \mathit{Pairs}_{\mathbb{S}}$ for all $a \in A$, $A \cup \{b_1\} = A_1 \in \mathbb{S}$ must hold. We can do this for each $b_i \in B$ and get $A_{n-1} \cup \{b_n\} = (A \cup B) \in \mathbb{S}$. □

Observe that for incomparable $\mathbb{S}$ the property of being adm-closed reduces to check for each $A, B \in \mathbb{S}$ $(A \neq B)$, whether there exist $a, b \in A \cup B$ such that $(a, b) \notin \mathit{Pairs}_{\mathbb{S}}$.

We can give results analogous to those of Lemma 2.

Lemma 4. *For an adm-closed extension-set $\mathbb{S} \subseteq 2^{\mathfrak{A}}$,*

1. *the $\subseteq$-maximal elements in $\mathbb{S}$ form an adm-closed set,*
2. *if $\mathbb{S}$ is incomparable then each $\mathbb{S}' \subseteq \mathbb{S}$ is adm-closed, and*
3. *$\mathbb{S} \cup \{\emptyset\}$ is adm-closed.*

Proof. (1) basically follows from the fact that $\mathit{Pairs}_{\mathbb{S}} = \mathit{Pairs}_{\mathbb{S}'}$ where $\mathbb{S}'$ is the set of $\subseteq$-maximal elements in $\mathbb{S}$. For (2) recall that for incomparable $\mathbb{S}$, adm-closed reduces to check for each $A, B \in \mathbb{S}$ with $A \neq B$, whether there exist $a, b \in A \cup B$ such that $(a, b) \notin \mathit{Pairs}_{\mathbb{S}}$. It is easy to see that this property still holds for $\mathbb{S}' \subseteq \mathbb{S}$ (since then, $\mathit{Pairs}'_{\mathbb{S}} \subseteq \mathit{Pairs}_{\mathbb{S}}$). (3) holds, since $S \cup \emptyset \in \mathbb{S}$ for each $S \in \mathbb{S}$ ensures that $\mathbb{S} \cup \{\emptyset\}$ is adm-closed by definition. □

The following proposition finally shows the role the property of being adm-closed plays in terms of admissible, preferred and semi-stable semantics.

Proposition 2. *For each* AF $F = (A, R)$,

1. *$\mathit{adm}(F)$ is adm-closed and contains $\emptyset$;*
2. *$\mathit{pref}(F)$ is non-empty, incomparable and adm-closed;*
3. *$\mathit{sem}(F)$ is non-empty, incomparable and adm-closed.*

Proof. ad 1) By definition, $\emptyset$ is always admissible. We show that $\mathit{adm}(F)$ is adm-closed. Towards a contradiction, assume $B, C \in \mathit{adm}(F)$ such that for all $b, c \in B \cup C$, $(b, c) \in \mathit{Pairs}_{\mathit{adm}(F)}$, but $B \cup C \notin \mathit{adm}(F)$. From Lemma 1 we know that $B \cup C$ defends itself in F. So for $B \cup C \notin \mathit{adm}(F)$ there must be a conflict in $B \cup C$, i.e. there is an attack $(b, c) \in R$ such that $\{b, c\} \subseteq B \cup C$. But then, for all $D \in \mathit{adm}(F)$, $\{b, c\} \not\subseteq D$. Hence, $(b, c) \notin \mathit{Pairs}_{\mathit{adm}(F)}$, a contradiction.

ad 2+3) By definition, both semantics $\sigma \in \{\mathit{pref}, \mathit{sem}\}$ yield at least one extension and for any $S, S' \in \sigma(F)$, $S \subseteq S'$ implies $S = S'$. By Lemma 4 and definition of preferred extensions, the fact that $\mathit{pref}(F)$ is adm-closed follows from the already shown property of $\mathit{adm}(F)$ being adm-closed. Since $\mathit{sem}(F) \subseteq \mathit{pref}(F)$ holds for all AFs F, Lemma 4 immediately shows that $\mathit{sem}(F)$ is adm-closed, as well. □

Example 2. Consider the framework F in Figure 2 and let $A = \{a, b\}$, $B = \{a, c, e\}$, $C = \{b, d, e\}$, and $\mathbb{S} = \{A, B, C\}$. We have $\mathit{pref}(F) = \mathit{sem}(F) = \mathbb{S}$. $\mathbb{S}$ is adm-closed, since for each pair of extensions, there exists a pair of arguments not contained in $\mathit{Pairs}_{\mathbb{S}}$: $b, c \in A \cup B$ and $(b, c) \notin \mathit{Pairs}_{\mathbb{S}}$; $a, d \in A \cup C$ and $(a, d) \notin \mathit{Pairs}_{\mathbb{S}}$; $c, d \in B \cup C$ and $(c, d) \notin \mathit{Pairs}_{\mathbb{S}}$. However, we also observe

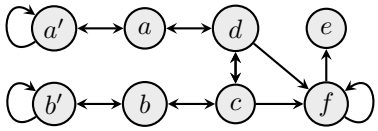


Figure 2: Argumentation framework F used in Example 2.

that $\mathbb{S}$ is not tight, since $A \cup \{e\} \notin \mathbb{S}$ but both (a, e) and (b, e) are contained in $\mathit{Pairs}_{\mathbb{S}}$. This shows that the reverse of Lemma 3 does not hold.

Complete Sets

Finally, we turn to the complete semantics. Compared to being adm-closed, we need a further relaxation.

Definition 6. Given an extension-set $\mathbb{S} \subseteq 2^{\mathfrak{A}}$ and $E \subseteq \mathfrak{A}$. we define the completion-sets $\mathbb{C}_{\mathbb{S}}(E)$ of E in $\mathbb{S}$ as the set of $\subseteq$-minimal sets $S \in \mathbb{S}$ with $E \subseteq S$.

In words, completion sets just give the "next" (in terms of supersets of A) elements contained in $\mathbb{S}$.

Definition 7. A set $\mathbb{S} \subseteq 2^{\mathfrak{A}}$ is called *com-closed* if for each $\mathbb{T} \subseteq \mathbb{S}$ the following holds: if $(a, b) \in \mathit{Pairs}_{\mathbb{S}}$ for each $a, b \in \mathit{Arg}_{\mathbb{T}} = \bigcup_{T \in \mathbb{T}} T$, then there exists a unique completion-set $S \in \mathbb{C}_{\mathbb{S}}(\mathit{Arg}_{\mathbb{T}})$. For a com-closed extension-set $\mathbb{S} \subseteq 2^{\mathfrak{A}}$ and $E \subseteq \mathfrak{A}$, we denote the unique element of $\mathbb{C}_{\mathbb{S}}(E)$ by $C_{\mathbb{S}}(E)$.

Lemma 5. *Each adm-closed extension-set is com-closed.*

Proof. Consider an adm-closed extension-set $\mathbb{S}$ and $\mathbb{T} \subseteq \mathbb{S}$. Then $(a, b) \in \mathit{Pairs}_{\mathbb{S}}$ for each $a, b \in \mathit{Arg}_{\mathbb{T}}$ implies $\mathit{Arg}_{\mathbb{T}} \in \mathbb{S}$, i.e. $\mathbb{C}_{\mathbb{S}}(\mathit{Arg}_{\mathbb{T}}) = \{\mathit{Arg}_{\mathbb{T}}\}$. □

In case of incomparable sets, the notions of adm-closed and com-closed coincide. In anticipation of the following result, this coincidence reflects the fact that all preferred extensions are complete.

Proposition 3. *For each* AF F, *$\mathit{com}(F)$ is a non-empty, com-closed extension-set with $(\bigcap_{S \in \mathit{com}(F)} S) \in \mathit{com}(F)$.*

Proof. First note that there is always at least one complete extension, namely the grounded extension. Moreover the grounded extension is the unique $\subseteq$-minimal complete extension and hence $(\bigcap_{S \in \mathit{com}(F)} S) \in \mathit{com}(F)$. Finally consider a set of complete extensions $\mathbb{E} \subseteq \mathit{com}(F)$ such that $(a, b) \in \mathit{Pairs}_{\mathit{com}(F)}$ for each $a, b \in \mathit{Arg}_{\mathbb{E}}$. By Lemma 1, $\mathit{Arg}_{\mathbb{E}}$ is an admissible set and thus can be extended to a unique complete extension $E' \supseteq \mathit{Arg}_{\mathbb{E}}$ by iteratively adding all defended arguments. Therefore $\mathit{com}(F)$ is com-closed. □

Example 3. Consider the AF $F = (\{a, a', b, b', c\}, \{(x, x'), (x', x), (x', x'), (x', c) \mid x \in \{a, b\}\})$. We have $\mathit{com}(F) = \{\emptyset, \{a\}, \{b\}, \{a, b, c\}\}$, which is com-closed, in particular, as $\mathbb{C}_{\mathit{com}(F)}(\{a\} \cup \{b\}) = \{\{a, b, c\}\}$. Observe that since $(a, b) \in \mathit{Pairs}_{\mathit{com}(F)}$, but $\{a, b\} \notin \mathit{com}(F)$, $\mathit{com}(F)$ is not adm-closed. This also shows that the reverse direction of Lemma 5 does not hold.

4 Realizability

In the previous section we have given necessary characteristics for the extension-sets $\mathbb{S} \in \Sigma_\sigma$, where $\sigma \in \{cf, adm, naive, stb, stage, pref, sem, com\}$. Except for the complete semantics, we will now show that these characteristics are also sufficient.

To this end, we need the concept of *realizability*: an extension-set $\mathbb{S} \subseteq 2^{\mathfrak{A}}$ is σ-realizable if there is an AF $F \in AF_{\mathfrak{A}}$, such that $\sigma(F) = \mathbb{S}$. We start with the following concept of a canonical argumentation framework, which will underlie all subsequent results on realizability.

Definition 8. Given an extension-set $\mathbb{S}$, we define the canonical argumentation framework for $\mathbb{S}$ as

$$F^{cf}_{\mathbb{S}} = \big(Arg_{\mathbb{S}}, (Arg_{\mathbb{S}} \times Arg_{\mathbb{S}}) \setminus Pairs_{\mathbb{S}}\big).$$

The idea behind the framework is simple: we draw a relation between two arguments iff they do *not* occur jointly in any set $S \in \mathbb{S}$. Thus, for any $\mathbb{S}$, $F^{cf}_{\mathbb{S}}$ is symmetric and has no self-attacking arguments (recall that for each $a \in Arg_{\mathbb{S}}$, $(a, a) \in Pairs_{\mathbb{S}}$). As an example, consider $\mathbb{T} = \{\{a_1, b_2, b_3\}, \{a_2, b_1, b_3\}, \{a_3, b_1, b_2\}, \{b_1, b_2, b_3\}\}$. $F^{cf}_{\mathbb{T}}$ has the same structure as the AF from Figure 1, but with all attacks being symmetric. We obtain $stb(F^{cf}_{\mathbb{T}}) = naive(F^{cf}_{\mathbb{T}}) = \mathbb{T}$. When we consider $\mathbb{S} = \{\{a_1, b_2, b_3\}, \{a_2, b_1, b_3\}, \{a_3, b_1, b_2\}\}$, i.e. $\mathbb{S} = \mathbb{T} \setminus \{\{b_1, b_2, b_3\}\}$, we obtain the same framework $F^{cf}_{\mathbb{S}} = F^{cf}_{\mathbb{T}}$, since $Pairs_{\mathbb{S}} = Pairs_{\mathbb{T}}$. In terms of naive semantics, this is not problematic, since $\mathbb{S}$ (as discussed in Example 1) cannot be realized via naive semantics at all. However, this observation readily suggests that realizing $\mathbb{S}$ with, say, stable semantics, requires additional concepts.

Stable, Stage, Naive and Conflict-Free Sets

Proposition 4. *(1) For each downward-closed and tight extension-set $\mathbb{S} \neq \emptyset$, $cf(F^{cf}_{\mathbb{S}}) = \mathbb{S}$. (2) For each incomparable extension-set $\mathbb{S} \neq \emptyset$, where $dcl(\mathbb{S})$ is tight, $naive(F^{cf}_{\mathbb{S}}) = \mathbb{S}$.*

Proof. (1) Let $F^{cf}_{\mathbb{S}} = (Arg_{\mathbb{S}}, R_{\mathbb{S}})$. $cf(F^{cf}_{\mathbb{S}}) \subseteq \mathbb{S}$: Observe that for each $E \in cf(F^{cf}_{\mathbb{S}})$, $(a, b) \in Pairs_{\mathbb{S}}$ for all $a, b \in E$ by construction of $R_{\mathbb{S}}$. Now suppose there exists $E' \in cf(F^{cf}_{\mathbb{S}})$ such that $E' \notin \mathbb{S}$. Wlog let E' be $\subseteq$-minimal with this property. Then $E' = S \cup \{c\}$ for some $S \in \mathbb{S}$. As $\mathbb{S}$ is tight there is an $s \in S$ such that $(s, c) \notin Pairs_{\mathbb{S}}$, a contradiction to the above observation. $cf(F^{cf}_{\mathbb{S}}) \supseteq \mathbb{S}$: Follows immediately by the fact that $\forall (a, b) \in Pairs_{\mathbb{S}} : (a, b) \notin R_{\mathbb{S}}$.

(2) First, note that $F^{cf}_{dcl(\mathbb{S})} = F^{cf}_{\mathbb{S}}$, since $Arg_{dcl(\mathbb{S})} = Arg_{\mathbb{S}}$ and $Pairs_{dcl(\mathbb{S})} = Pairs_{\mathbb{S}}$. $cf(F^{cf}_{\mathbb{S}}) = dcl(\mathbb{S})$ follows from (1). By construction of $dcl(\mathbb{S})$, the $\subseteq$-maximal sets in $dcl(\mathbb{S})$ are given by $\mathbb{S}$ and as naive sets are just $\subseteq$-maximal conflict-free, $naive(F^{cf}_{\mathbb{S}}) = \mathbb{S}$ follows. □

We proceed with stable and stage semantics. Stable semantics are the only semantics that can realize $\mathbb{S} = \emptyset$. Note that $\mathbb{S} = \emptyset$ is easily *stb*-realizable, for instance with the framework $(\{a\}, \{(a, a)\})$. In Proposition 1 the only difference between stable and stage semantics was the case $\mathbb{S} = \emptyset$.

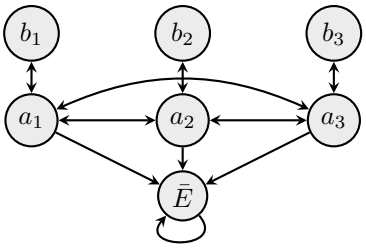


Figure 3: Excluding $\{b_1, b_2, b_3\}$ from $stb(F^{cf}_{\mathbb{S}})$.

The next result shows that this indeed is the only difference between the signatures for stable and stage semantics.

The idea of the construction below is to suitably extend the canonical framework from Definition 8 such that undesired sets are excluded. Coming back to our example with $\mathbb{S} = \{\{a_1, b_2, b_3\}, \{a_2, b_1, b_3\}, \{a_3, b_1, b_2\}\}$, recall that $F^{cf}_{\mathbb{S}}$ had one such undesired naive (and stable) extension, $E = \{b_1, b_2, b_3\}$. To get rid of it we add a new argument which is attacked by all other sets from $\mathbb{S}$ but not by E, see Figure 3 for illustration. The next definition generalizes this idea which is due to (Dvořák and Spanring 2012).

Definition 9. Given an extension-set $\mathbb{S}$ and its canonical framework $F^{cf}_{\mathbb{S}} = (A^{cf}_{\mathbb{S}}, R^{cf}_{\mathbb{S}})$, let $\mathbb{X} = stb(F^{cf}_{\mathbb{S}}) \setminus \mathbb{S}$. We define $F^{st}_{\mathbb{S}} = (A^{cf}_{\mathbb{S}} \cup \{\bar{E} \mid E \in \mathbb{X}\}, R^{cf}_{\mathbb{S}} \cup \{(\bar{E}, \bar{E}), (a, \bar{E}) \mid E \in \mathbb{X}, a \in Arg_{\mathbb{S}} \setminus E\})$.

Proposition 5. *For each non-empty, incomparable and tight extension-set $\mathbb{S}$, $stb(F^{st}_{\mathbb{S}}) = stage(F^{st}_{\mathbb{S}}) = \mathbb{S}$.*

Proof. Since $\mathbb{S}$ is non-empty, showing existence of an AF F with $stb(F) = \mathbb{S}$ is sufficient (for each F with $stb(F) \neq \emptyset$, $stb(F) = stage(F)$ holds).

Moreover, for each incomparable and tight extension-set $\mathbb{S}$, it holds that $\mathbb{S} \subseteq stb(F^{cf}_{\mathbb{S}})$: Towards a contradiction, let $S \in \mathbb{S}$ such that $S \notin stb(F^{cf}_{\mathbb{S}})$. By construction of $F^{cf}_{\mathbb{S}}$, $S \in cf(F^{cf}_{\mathbb{S}})$. Thus $\exists a \in Arg_{\mathbb{S}} : S \not\mapsto a$. But then by construction, $(a, s) \in Pairs_{\mathbb{S}}$, for all $s \in S$; this contradicts $\mathbb{S}$ being tight.

Now let $\mathbb{X} = stb(F^{cf}_{\mathbb{S}}) \setminus \mathbb{S}$ and consider $F^{st}_{\mathbb{S}}$. We show that $stb(F^{st}_{\mathbb{S}}) = \mathbb{S}$. $stb(F^{st}_{\mathbb{S}}) \subseteq \mathbb{S}$: Let $E \in stb(F^{st}_{\mathbb{S}})$. Then also $E \in stb(F^{cf}_{\mathbb{S}}) = \mathbb{X} \cup \mathbb{S}$. But if $E \in \mathbb{X}$, by Definition 9, $E \not\mapsto \bar{E}$ and also $\bar{E} \notin E$, thus $E \notin stb(F^{st}_{\mathbb{S}})$. Hence, $E \in \mathbb{S}$. $stb(F^{st}_{\mathbb{S}}) \supseteq \mathbb{S}$: Let $E \in \mathbb{S}$. We already know that $E \in stb(F^{cf}_{\mathbb{S}})$. Now consider $F^{st}_{\mathbb{S}}$. E still attacks all arguments in $Arg_{\mathbb{S}} \setminus E$. Now consider an arbitrary argument $\bar{E}'$ for $E' \in \mathbb{X}$. $\bar{E}'$ is attacked by all arguments $a \in Arg_{\mathbb{S}} \setminus E'$ and as E and E' are both stable in $F^{cf}_{\mathbb{S}}$ (and thus incomparable) at least one of these arguments must be contained in E. Hence $E \in stb(F^{st}_{\mathbb{S}})$. □

Preferred, Semi-Stable and Admissible Sets

Towards a suitable canonical AF for admissibility-based semantics we introduce the following technical concept.

Definition 10. Given an extension-set $\mathbb{S}$, the *defense-formula* $\mathcal{D}^{\mathbb{S}}_a$ of an argument $a \in Arg_{\mathbb{S}}$ in $\mathbb{S}$ is defined as $\bigvee_{S \in \mathbb{S} \text{ s.t. } a \in S} \bigwedge_{s \in S \setminus \{a\}} s$. $\mathcal{D}^{\mathbb{S}}_a$ given as (a logically equivalent) CNF is called *CNF-defense-formula* $\mathcal{CD}^{\mathbb{S}}_a$ of a in $\mathbb{S}$.

Intuitively, $\mathcal{D}^{\mathbb{S}}_a$ describes the conditions for the argument a being in an extension. Each disjunct represents a set of arguments which jointly allows a to "join" an extension, i.e. represents a collection of arguments defending a. Further note that $\mathcal{D}^{\mathbb{S}}_a = \top$ if and only if $\{a\} \in \mathbb{S}$.

Example 4. Consider $\mathbb{T} = \{\emptyset, \{a\}, \{b, c\}, \{a, c, d\}\}$. Then $\mathcal{D}^{\mathbb{T}}_a = \top, \mathcal{D}^{\mathbb{T}}_b = c, \mathcal{D}^{\mathbb{T}}_c = b \vee (a \wedge d)$ and $\mathcal{D}^{\mathbb{T}}_d = a \wedge c$. The corresponding CNF-defense-formulas[1] are $\mathcal{CD}^{\mathbb{T}}_a = \{\}$, $\mathcal{CD}^{\mathbb{T}}_b = \{\{c\}\}$, $\mathcal{CD}^{\mathbb{T}}_c = \{\{a, b\}, \{b, d\}\}$, and $\mathcal{CD}^{\mathbb{T}}_d = \{\{a\}, \{c\}\}$.

Lemma 6. *Given an extension-set $\mathbb{S}$ and an argument $a \in Arg_{\mathbb{S}}$, then for each $S \subseteq Arg_{\mathbb{S}}$ with $a \in S$: $(S \setminus \{a\})$ is a model of $\mathcal{D}^{\mathbb{S}}_a$ (resp. $\mathcal{CD}^{\mathbb{S}}_a$) iff there exists an $S' \subseteq S$ with $a \in S'$ such that $S' \in \mathbb{S}$.*

Proof. The if-direction follows straight forwardly by definition of $\mathcal{D}^{\mathbb{S}}_a$, since the conjunction of the elements of $S' \setminus \{a\}$ forms a term of $\mathcal{D}^{\mathbb{S}}_a$ for each $S' \in \mathbb{S}$ with $a \in S'$.

To show the only-if-direction consider some $S \subseteq Arg_{\mathbb{S}}$ with $a \in S$ where $S \setminus \{a\}$ is a model of $\mathcal{D}^{\mathbb{S}}_a$. If $\mathcal{D}^{\mathbb{S}}_a = \top$ then it holds that $\{a\} \in \mathbb{S}$. For $S \setminus \{a\}$ to be a model of $\mathcal{D}^{\mathbb{S}}_a \neq \top$, there must be some term $\tau \in \mathcal{D}^{\mathbb{S}}_a$, whose elements form a subset of $S \setminus \{a\}$. Consider such a term $\tau \in \mathcal{D}^{\mathbb{S}}_a$. Then by construction of $\mathcal{D}^{\mathbb{S}}_a$ there is some $S' \in \mathbb{S}$ with $a \in S'$, where $S' \setminus \{a\}$ coincides with the elements of τ. So $S' \subseteq S$. □

We make use of the defense formulas in the following construction.

Definition 11. Given an extension-set $\mathbb{S}$, the *canonical defense-argumentation-framework* $F^{def}_{\mathbb{S}} = (A^{def}_{\mathbb{S}}, R^{def}_{\mathbb{S}})$ extends the canonical AF $F^{cf}_{\mathbb{S}} = (Arg_{\mathbb{S}}, R^{cf}_{\mathbb{S}})$ as follows:

$$A^{def}_{\mathbb{S}} = Arg_{\mathbb{S}} \cup \bigcup_{a \in Arg_{\mathbb{S}}} \{\alpha_{a\gamma} \mid \gamma \in \mathcal{CD}^{\mathbb{S}}_a\}, \text{ and}$$

$$R^{def}_{\mathbb{S}} = R^{cf}_{\mathbb{S}} \cup \bigcup_{a \in Arg_{\mathbb{S}}} \{(b, \alpha_{a\gamma}), (\alpha_{a\gamma}, \alpha_{a\gamma}), (\alpha_{a\gamma}, a) | \gamma \in \mathcal{CD}^{\mathbb{S}}_a, b \in \gamma\}.$$

$F^{def}_{\mathbb{S}}$ consists of all arguments given in the extension-set plus a certain amount of additional arguments. Each $\alpha_{a\gamma}$ attacks argument a and is attacked by all arguments occurring as literals in clause γ of the CNF-defense-formula of a. So in $F^{def}_{\mathbb{S}}$ for a to be defended from $\alpha_{a\gamma}$ it takes at least one argument of these occurring as atoms in clause γ of $\mathcal{CD}^{\mathbb{S}}_a$.

Example 5. Again consider extension-set $\mathbb{T} = \{\emptyset, \{a\}, \{b, c\}, \{a, c, d\}\}$. We have given the CNF-defense-formulas in Example 4. $F^{def}_{\mathbb{T}}$ is given by the AF in Figure 4. Considering, for instance, argument c, where $\mathcal{CD}^{\mathbb{T}}_c = \{\{a, b\}, \{b, d\}\}$, one can see that in $F^{def}_{\mathbb{T}}$ it takes a or b in order to defend c from $\alpha_{c\{a,b\}}$, and b or d in order to defend c from $\alpha_{c\{b,d\}}$.

Proposition 6. *For each adm-closed extension-set $\mathbb{S}$ where $\emptyset \in \mathbb{S}$, $adm(F^{def}_{\mathbb{S}}) = \mathbb{S}$.*

[1] In what follows, we write CNF formulas in clause form.

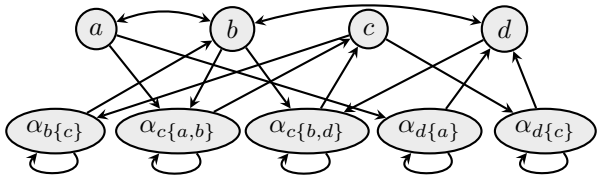


Figure 4: AF $F^{def}_{\mathbb{T}}$ used in Example 5.

Proof. $\mathbb{S} \subseteq adm(F^{def}_{\mathbb{S}})$: Let $S \in \mathbb{S}$. If $S = \emptyset$, the assertion trivially holds. If $S = \{a\}$, then $\mathcal{CD}^{\mathbb{S}}_a$ is the empty set of clauses. By definition of $R^{def}_{\mathbb{S}}$, a is then defended in $F^{def}_{\mathbb{S}}$ and thus $S \in adm(F^{def}_{\mathbb{S}})$. Thus let $S \in \mathbb{S}$ contain at least two arguments. By construction, S is conflict-free in $F^{def}_{\mathbb{S}}$. It remains to show that each $s \in S$ is defended by S in $F^{def}_{\mathbb{S}}$. Let $s \in S$. We know from Lemma 6 that $S \setminus \{s\}$ is a model of $\mathcal{CD}^{\mathbb{S}}_s$. Hence, each clause $\gamma \in \mathcal{CD}^{\mathbb{S}}_s$ contains at least one variable $t_\gamma \in S \setminus \{s\}$. Thus, by construction of $R^{def}_{\mathbb{S}}$, $(S \setminus \{s\}) \mapsto \alpha_{s\gamma}$ for each $\gamma \in \mathcal{CD}^{\mathbb{S}}_s$, i.e. S defends s in $F^{def}_{\mathbb{S}}$.

$adm(F^{def}_{\mathbb{S}}) \subseteq \mathbb{S}$: Let $S \in adm(F^{def}_{\mathbb{S}})$. For $S = \emptyset$, $S \in \mathbb{S}$ by definition. If $S = \{a\}$, then $a \mapsto b$ whenever some argument b attacks a. Hence, $\mathcal{CD}^{\mathbb{S}}_a = \{\}$, i.e. $\mathcal{D}^{\mathbb{S}}_a = \top$ must hold, therefore $S \in \mathbb{S}$. Now assume S contains at least two arguments. As S is conflict-free in $F^{def}_{\mathbb{S}}$, the construction of $R^{def}_{\mathbb{S}}$ guarantees that for all $a, b \in S$, $(a, b) \in Pairs_{\mathbb{S}}$. Let $s \in S$ with $\{s\} \notin adm(F^{def}_{\mathbb{S}})$. Then we have $\alpha_{s\gamma} \mapsto s$ for each $\gamma \in \mathcal{CD}^{\mathbb{S}}_s$. Since s is defended by S, for each $\gamma \in \mathcal{CD}^{\mathbb{S}}_s$, $\exists t_\gamma \in (S \setminus \{s\}) : t_\gamma \mapsto \alpha_{s\gamma}$. By definition of $F^{def}_{\mathbb{S}}$, thus t_γ occurs in the clause γ. It follows that $T = \{t_\gamma \mid \gamma \in \mathcal{CD}^{\mathbb{S}}_s\}$ is a model of $\mathcal{CD}^{\mathbb{S}}_s$ and $\mathcal{D}^{\mathbb{S}}_s$. Then by Lemma 6 there is some $S'_s \subseteq T \cup \{s\}$ (note that also $S'_s \subseteq S$ as $s \in S$ and for each $t_\gamma \in T$ also $t_\gamma \in S$) with $s \in S'_s$ such that $S'_s \in \mathbb{S}$. Recall also that in case $\{s\} \in adm(F^{def}_{\mathbb{S}})$, we know that $\{s\} \in \mathbb{S}$ (say $S'_s = \{s\}$). Now since $(a, b) \in Pairs_{\mathbb{S}}$ for all $a, b \in S$ and since $\mathbb{S}$ is adm-closed, it must hold that $S'_{s_1} \cup S'_{s_2} \in \mathbb{S}$ for any $s_1, s_2 \in S$. Hence $S = \left(\bigcup_{s_i \in S} S'_{s_i}\right) \in \mathbb{S}$. □

The construction also works for further semantics.

Proposition 7. *For each non-empty, incomparable and adm-closed extension-set $\mathbb{S}$ there exist F, F' with $pref(F) = \mathbb{S}$ and $sem(F') = \mathbb{S}$.*

Proof. By Proposition 6 and Lemma 4.3, $adm(F^{def}_{\mathbb{S}}) = \mathbb{S} \cup \{\emptyset\}$. Since $\mathbb{S}$ is incomparable, $pref(F^{def}_{\mathbb{S}}) = \mathbb{S}$. Moreover, as shown by Dvořák and Woltran (2011), for each AF F there is an AF F' such that $pref(F) = sem(F')$. □

Each extension-set $\mathbb{S}$ realizable under the admissible semantics, can also be realized under the complete semantics: extend $F^{def}_{\mathbb{S}}$ for each $a \in Arg_{\mathbb{S}}$ by a self-attacking argument with an attack from and to a. As Example 3 already suggests, the complete semantics is however capable of realizing more diverse extension-sets. However, the property of being com-closed (cf. Definition 7) so far only gives an approximation of the sets which can realized.

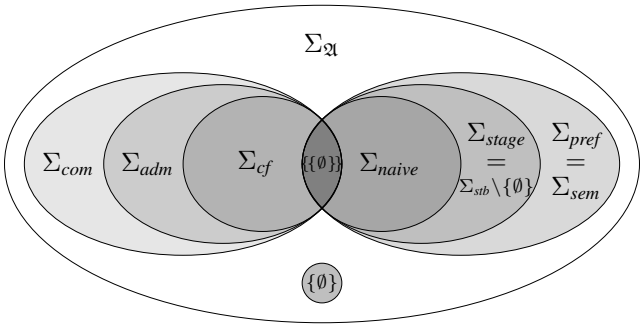


Figure 5: Relations between signatures.

Example 6. Let $\mathbb{S} = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b, c\}, \{a, d, e\}, \{b, d, f\}, \{x, c\}, \{x, d\}\}$. $\mathbb{S}$ is com-closed and and satisfies $(\bigcap_{S \in \mathbb{S}} S) = \emptyset \in \mathbb{S}$. We argue that $\mathbb{S}$ is not *com*-realizable. Towards a contradiction consider an AF F such that $com(F) = \mathbb{S}$. First as $\{x, c\}, \{x, d\} \in \mathbb{S}$ and no superset of $\{x, c, d\}$ is in $\mathbb{S}$, there must be a conflict between c and d. In case $(c, d) \in R$ and $(d, c) \notin R$, (x, d) is not admissible (as $(x, c) \in Pairs_{\mathbb{S}}$). Similar for $(d, c) \in R$ and $(c, d) \notin R$. Thus $(c, d), (d, c) \in R$. Now consider $\{a, b\}$ which must be admissible as $\{a\}, \{b\}$ are admissible and $(a, b) \in Pairs_{\mathbb{S}}$. But as $\{a, b\} \notin \mathbb{S}$ it has to defend argument c ($C_{\mathbb{S}}(\{a, b\}) = \{a, b, c\}$). But we have $(d, c) \in R$ and $(a, d), (b, d) \in Pairs_{\mathbb{S}}$. Hence $\{a, b\}$ cannot defend c.

5 Signatures & Limits of Disagreement

We now state our main result, i.e. exact characterizations of the signatures for the semantics dealt with in this paper.

Theorem 1. *The signatures for the considered semantics are given by the following collections of extension-sets.*

$$\Sigma_{cf} = \{\mathbb{S} \neq \emptyset \mid \mathbb{S} \text{ is downward-closed and tight}\}$$
$$\Sigma_{naive} = \{\mathbb{S} \neq \emptyset \mid \mathbb{S} \text{ is incomparable and } dcl(\mathbb{S}) \text{ is tight}\}$$
$$\Sigma_{stb} = \{\mathbb{S} \mid \mathbb{S} \text{ is incomparable and tight}\}$$
$$\Sigma_{stage} = \{\mathbb{S} \neq \emptyset \mid \mathbb{S} \text{ is incomparable and tight}\}$$
$$\Sigma_{adm} = \{\mathbb{S} \neq \emptyset \mid \mathbb{S} \text{ is adm-closed and contains } \emptyset\}$$
$$\Sigma_{pref} = \Sigma_{sem} = \{\mathbb{S} \neq \emptyset \mid \mathbb{S} \text{ is incomparable and adm-closed}\}$$

All relations of the theorem are immediate from results in the previous section and the corresponding characteristics from Propositions 1 and 2. Moreover, $\Sigma_{com} \subset \{\mathbb{S} \neq \emptyset \mid \mathbb{S} \text{ is com-closed and } (\bigcap_{S \in \mathbb{S}} S) \in \mathbb{S}\}$, cf. Proposition 3.

By inspecting the respective properties, we can now immediately put the signatures of different semantics in relation to each other. For illustration see Figure 5, where $\Sigma_{\mathfrak{A}}$ just denotes the set of all extensions-sets over $\mathfrak{A}$, and the singleton $\{\emptyset\}$, i.e. the extension-set containing no extension, belongs to Σ_{stb}.

Theorem 2. *The following relations hold*

$$\Sigma_{naive} \subset \Sigma_{stage} \subset \Sigma_{sem} = \Sigma_{pref}, \qquad \Sigma_{stb} = \Sigma_{stage} \cup \{\emptyset\}$$
$$\Sigma_{cf} \subset \Sigma_{adm} \subset \Sigma_{com}$$
$$\{dcl(\mathbb{S}) \mid \mathbb{S} \in \Sigma_{naive}\} = \Sigma_{cf}, \quad \Sigma_{adm} \supset \{\mathbb{S} \cup \{\emptyset\} \mid \mathbb{S} \in \Sigma_{pref}\}$$

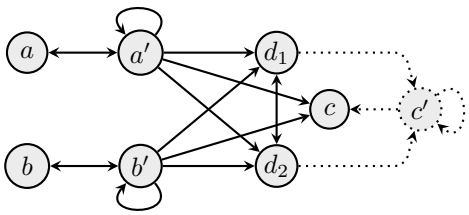


Figure 6: AFs F_1,F_2 with $(com(F_1) \cap com(F_2)) \notin \Sigma_{com}$.

Proof. In what follows, we make implicit use of the results from Theorem 1. First, by Lemma 2.1, $\Sigma_{naive} \subseteq \Sigma_{stage}$; $\Sigma_{naive} \neq \Sigma_{stage}$ is witnessed by Example 1. Relations $\Sigma_{stage} \subseteq \Sigma_{sem}$ and $\Sigma_{cf} \subseteq \Sigma_{adm}$ follow from Lemma 3. $\Sigma_{stage} \neq \Sigma_{sem}$ is by Example 2. $\Sigma_{cf} \neq \Sigma_{adm}$ is seen by extension-set $\{\emptyset, \{a, b\}\}$ which is adm-closed but not downward-closed. The fact that *adm*-realizability implies *com*-realizability and Example 3 show $\Sigma_{adm} \subset \Sigma_{com}$. The relations in the last line follow from the definition of $dcl(\cdot)$ and Lemma 4.3 respectively. □

With our results on signatures at hand, we now provide some interesting implications. The first result examines the question whether the intersection of two extension-sets under a given semantics σ, can always be realized.

Theorem 3. *For each $\sigma \in \{cf, naive, stb, stage, pref, sem, adm\}$ it holds that for any AFs F_1, F_2 there exists an AF F such that $\sigma(F) = \sigma(F_1) \cap \sigma(F_2)$ if $\sigma(F_1) \cap \sigma(F_2) \neq \emptyset$.*[2]

Proof. (Sketch.) We have to show that $\sigma(F_1) \cap \sigma(F_2)$ satisfies the properties according to the signature of the particular semantics. Existence of F is then a consequence of Theorem 1. The result for *stb*, *stage*, *pref*, *sem* follows from Lemma 2.2 and Lemma 4.2. The argument for *adm* is as follows (*cf* and *naive* are handled in a similar way). Assume $\mathbb{S} = adm(F_1) \cap adm(F_2)$ is not adm-closed, i.e. there are $A, B \in \mathbb{S}$ such that for all $a, b \in (A \cup B)$, $(a, b) \in Pairs_{\mathbb{S}}$, but $(A \cup B) \notin \mathbb{S}$. Then there is $i \in \{1, 2\}$, such that $A, B \in adm(F_i)$ but $(A \cup B) \notin adm(F_i)$. But then, $adm(F_i)$ is not adm-closed, a contradiction to Proposition 2. □

The complete semantics turns out to be the only semantics not closed under this form of intersection. Consider $\mathbb{S} = \{\emptyset, \{a\}, \{b\}, \{a, b, c, d_1\}, \{a, b, c, d_2\}\}$, $\mathbb{S}_1 = \mathbb{S} \cup \{\{a, b\}\}$, and $\mathbb{S}_2 = \mathbb{S} \cup \{\{a, b, c\}\}$. $\mathbb{S}_1$ and $\mathbb{S}_2$ are realizable under the complete semantics. Corresponding AFs are depicted in Figure 6: $\mathbb{S}_1$ are the complete extensions of the entire AF, and $\mathbb{S}_2$ the ones of the AF without the dotted part. However, $\mathbb{S} = \mathbb{S}_1 \cap \mathbb{S}_2$ is not com-closed and therefore, by Proposition 3, no AF F exists such that $com(F) = \mathbb{S}$.

Our next results concern limits in expressing multiple extensions. Our first result is positive in a sense that as long as two (maximal) sets of arguments are involved, all semantics are capable to deliver such extensions. Recall that three sets (e.g. $\mathbb{S} = \{\{a, b\}, \{a, c\}, \{b, c\}\}$) are already problematic.

Proposition 8. *For any incomparable extension-set $\mathbb{S} \neq \emptyset$ with $|\mathbb{S}| \leq 2$, $\mathbb{S} \in \Sigma_\sigma$ for $\sigma \in \{naive, stb, stage, pref, sem\}$.*

[2] $\sigma(F_1) \cap \sigma(F_2) \neq \emptyset$ is not a necessary condition for $\sigma = stb$.

Proof. By Theorem 2, it suffices to show the claim for naive semantics. By Theorem 1, we need to show that $dcl(\mathbb{S})$ is tight, which trivially holds for $|\mathbb{S}| = 1$. For $\mathbb{S} = \{S_1, S_2\}$, let $S \in dcl(\mathbb{S})$ and $a \in Arg_{\mathbb{S}}$ such that $S \cup \{a\} \notin dcl(\mathbb{S})$. Wlog. assume $S \subseteq S_1$. Then, $a \in S_2 \setminus S_1$ and $S \not\subseteq S_2$, i.e. there is some $s \in S \setminus S_2$. Since $|\mathbb{S}| = 2$, $(a, s) \notin Pairs_{\mathbb{S}}$. □

How many extensions can maximally be achieved by an AF under a semantics σ? Such results ease checking $\mathbb{S} \in \Sigma_\sigma$ whenever the cardinality of $\mathbb{S}$ exceeds a certain number.

Research in this direction has recently been initiated by Baumann and Strass (2014). They proposed a function giving the maximal number of stable extensions an AF with n arguments can have. In accordance to realizability, we are interested in the number of extensions as a function of a fixed amount of arguments occurring in any extension.

While our results on signatures give insights about the extent of structural diversity a semantics can express, the maximal number of extensions gives, to some degree, an answer to how much quantitative disagreement a semantics can express.

Definition 12. Given a semantics σ, we define the diversity-function $\Delta_\sigma(n) = \max_{F \in AF_{\mathfrak{A}}, \|\sigma(F)\|=n} |\sigma(F)|$.

It is easy to see that $\Delta_\sigma(n) = 2^n$ for $\sigma \in \{cf, adm, com\}$. For the other semantics, we first give a technical lemma.

Lemma 7. *For any* AF F*, there is a symmetric* AF F^{sym} *with* $\|pref(F)\| = \|pref(F^{sym})\|$ *and* $|pref(F)| \leq |pref(F^{sym})|$.

Proof (Sketch). Transform $F = (A, R)$ by (1) removing all arguments $a \notin Arg_{pref(F)}$ together with adjacent attacks, and (2) adding (b, a) to R if $(a, b) \in R$. Obviously conflict-freeness and defense is preserved, i.e. any set admissible in F is admissible in F^{sym}. Moreover, as only attacks to and from arguments not occurring in any preferred extension of F are removed, any conflict between two $E_1, E_2 \in pref(F)$ survives the translation, therefore there must be two different $E_1', E_2' \in pref(F^{sym})$ with $E_1 \subseteq E_1'$ and $E_2 \subseteq E_2'$. Hence $|pref(F)| \leq |pref(F^{sym})|$. As $Arg_{pref(F^{sym})}$ coincides with the arguments of F^{sym} (by symmetry of F^{sym}), it follows that $\|pref(F)\| = \|pref(F^{sym})\|$. □

Using the recent results by Baumann and Strass (2014), Lemma 7 and Theorem 1 allow to specify the function Δ_σ for all incomparable semantics σ.

Theorem 4. *For* $\sigma \in \{naive, stb, stage, pref, sem\}$*, we have*

$$\Delta_\sigma(n) = \begin{cases} 1 & n = 1 \\ 3^s & n \geq 2 \wedge n = 3s \\ 4 \cdot 3^{s-1} & n \geq 2 \wedge n = 3s + 1 \\ 2 \cdot 3^s & n \geq 2 \wedge n = 3s + 2. \end{cases}$$

Proof. Consider a semantics $\sigma \in \{naive, stb, stage, pref, sem\}$ and an AF F having maximal diversity, i.e. $|\sigma(F)| = \Delta_\sigma(\|\sigma(F)\|)$. Since $\Sigma_\sigma \subseteq \Sigma_{pref}$, we can find an AF F' with $pref(F') = \sigma(F)$. By Lemma 7 we can find a symmetric $F^{sym}{=}(A^{sym}, R^{sym})$ such that $\|pref(F^{sym})\| = \|\sigma(F)\|$ and $|pref(F^{sym})| = |\sigma(F)|$ (as we assume F maximally diverse). In this symmetric AF it holds that $pref(F^{sym}) = \sigma(F^{sym})$. As $\|\sigma(F^{sym})\| = |A^{sym}|$ we can generalize the result of (Baumann and Strass 2014) for $\Delta_{stb}(n)$ to all semantics σ. □

6 Complexity

In this section, we first consider the problem of checking membership in a signature for semantics σ, i.e. given an extension-set $\mathbb{S}$, does $\mathbb{S} \in \Sigma_\sigma$ hold? For most of the semantics, it is not hard to see that this can be done in polynomial time in the size of $\mathbb{S}$. The only exception is the naive semantics, since the characterization in Theorem 1 makes use of $dcl(\mathbb{S})$ which is not polynomially bounded in the size of $\mathbb{S}$.

We provide an alternative characterization based on the ternary majority operator maj_3: given $S_1, S_2, S_3 \subseteq \mathfrak{A}$, $maj_3(S_1, S_2, S_3) = (S_1 \cap S_2) \cup (S_2 \cap S_3) \cup (S_1 \cap S_3)$.

Proposition 9. *For an incomparable extension-set* $\mathbb{S}$ *it holds that* $dcl(\mathbb{S})$ *is tight iff for all* $S_1, S_2, S_3 \in \mathbb{S}$ *there is an* $S \in \mathbb{S}$*, such that* $maj_3(S_1, S_2, S_3) \subseteq S$.

Proof. Suppose for all $S_1, S_2, S_3 \in \mathbb{S}$ there is some $S \in \mathbb{S}$ such that $maj_3(S_1, S_2, S_3) \subseteq S$. Towards a contradiction assume that the downward-closure of $\mathbb{S}$ is not tight, i.e. there exist $S' \in dcl(\mathbb{S})$ and $a \in Arg_{\mathbb{S}} = Arg_{dcl(\mathbb{S})}$, such that $(S' \cup \{a\}) \notin dcl(\mathbb{S})$ and for all $s \in S'$, $(a, s) \in Pairs_{\mathbb{S}} = Pairs_{dcl(\mathbb{S})}$. Assume $|S'| = 1$, i.e. $S' = \{s\}$. As $(a, s) \in Pairs_{\mathbb{S}}$ by assumption, $\{a, s\} \in dcl(\mathbb{S})$, a contradiction to $S' \cup \{a\} \notin dcl(\mathbb{S})$. Hence $|S'| > 1$, i.e. $S' = \{s_1, \ldots, s_n\}$ with $n > 1$. By assumption, $\{s_1, \ldots, s_n, a\} \notin \mathbb{S}$, but $(a, s_i) \in Pairs_{\mathbb{S}}$ for each $s_i \in S'$. Hence, for each $s_i \in S'$ there is some $S_i \in \mathbb{S}$ with $\{a, s_i\} \subseteq S_i$ for $i = 1 \ldots n$. Moreover $maj_3(S_i, S_j, S') \supseteq \{s_i, s_j, a\}$ for each $i, j \in \{1, \ldots, n\}$ since $s_i \in S' \cap S_i$, $s_j \in S' \cap S_j$, and $a \in S_i \cap S_j$. Therefore there is some $S_{ij} \in \mathbb{S}$ with $S_{ij} \supseteq \{s_i, s_j, a\}$. Now for some $k \in \{1, \ldots, n\}$, we get $S_{ijk} = maj_3(S_{ij}, S_k, S') \supseteq \{s_i, s_j, s_k, a\}$ and $S_{ijk} \in \mathbb{S}$. Following this procedure for all $1 \ldots n$ yields a $T \in \mathbb{S}$ with $T \supseteq \{s_1, \ldots, s_n, a\}$, a contradiction to $S' \cup \{a\} \notin dcl(\mathbb{S})$.

On the other hand consider some extension-set $\mathbb{S}$ where $dcl(\mathbb{S})$ is tight and assume, towards a contradiction, sets $S_1, S_2, S_3 \in \mathbb{S}$ such that $maj_3(S_1, S_2, S_3) \not\subseteq S$, for all $S \in \mathbb{S}$. Now, consider the maximal $S' \subset maj_3(S_1, S_2, S_3)$ with $S' \in dcl(\mathbb{S})$, i.e. $\exists a \in maj_3(S_1, S_2, S_3) \setminus S'$ such that $S' \cup \{a\} \notin dcl(\mathbb{S})$. As $S' \cup \{a\} \subseteq maj_3(S_1, S_2, S_3)$, all $s \in (S' \cup \{a\})$ are also contained in $((S_1 \cap S_2) \cup (S_2 \cap S_3) \cup (S_1 \cap S_3))$. Thus for each $s \in S'$ there is some $S_i \in \{S_1, S_2, S_3\}$ with $\{s, a\} \subseteq S_i$. Hence, $(s, a) \in Pairs_{\mathbb{S}}$, so that $dcl(\mathbb{S})$ is not tight, a contradiction. □

Testing the majority criterion is done in polynomial time by looping over all triples (S_1, S_2, S_3) stemming from $\mathbb{S}$. We obtain the following theorem.

Theorem 5. *For semantics* $\sigma \in \{cf, naive, stb, stage, adm, pref, sem\}$*, given* $\mathbb{S}$*, testing* $\mathbb{S} \in \Sigma_\sigma$ *is in polynomial time.*

Next we exploit our results to give a preliminary complexity analysis in terms of the problem of *recasting*: given an AF $F_1 \in AF_{\mathfrak{A}}$ and two semantics σ_1, σ_2, decide whether there exists an $F_2 \in AF_{\mathfrak{A}}$, such that $\sigma_1(F_1) = \sigma_2(F_2)$. By the very nature of signatures, this is equivalent to test $\sigma_1(F_1) \in \Sigma_{\sigma_2}$. Table 1 summarizes our results. An entry for row σ_1 and column σ_2 gives the complexity of deciding whether $\sigma_1(F) \in \Sigma_{\sigma_2}$. $\mathcal{C}$-h abbreviates hardness for class $\mathcal{C}$; $\mathcal{C}$-c abbreviates completeness for class $\mathcal{C}$; "trivial" means

Table 1: Complexity of the recasting problem.

	cf	*adm*	*com*
cf	-	trivial	trivial
adm	coNP-c	-	trivial
com	coNP-c	coNP-c	-

	naive	*stb*	*stage*	*pref*	*sem*
naive	-	trivial	trivial	trivial	trivial
stb	coNP-h	-	NP-c	NP-c	NP-c
stage	coNP-h	trivial	-	trivial	trivial
pref	coNP-h	Π_2^P-c	Π_2^P-c	-	trivial
sem	coNP-h	Π_2^P-c	Π_2^P-c	trivial	-

that each instance is a "Yes"-instance. For both tables, these "trivial" entries are immediate from Theorem 2.

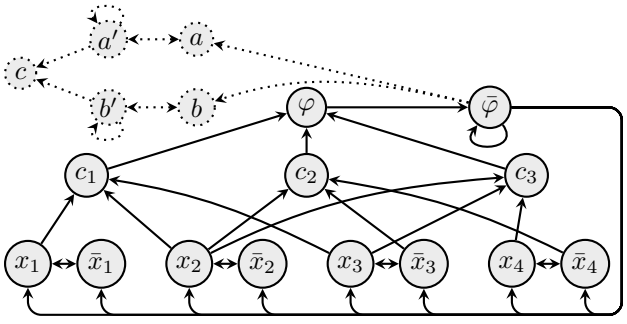
Theorem 6. *The complexity results depicted in Table 1 hold.*

Proof. (Sketch.) Due to the space constraints, we exemplify only one of the coNP-membership arguments for the upper table and provide a non-deterministic algorithm disproving, given an AF F, $com(F) \in \Sigma_{adm}$. Here, we check (i) $\emptyset \notin com(F)$ or (ii) $com(F)$ is not *adm*-closed. Clearly (i) can be done in polynomial time. For (ii) notice that if $com(F)$ is not *adm*-closed then there exist sets $A, B \in com(F)$ such that $A \cup B \in cf(F)$ but $A \cup B \notin com(F)$. But then there exists a set C with $A \cup B \subset C \in com(F)$. So to disprove $com(F)$ being *adm*-closed we guess sets A, B, C and test whether $A, B, C \in com(F)$, $A \cup B \subset C$ but $A \cup B \notin com(F)$.

For coNP-hardness, we first give the result for $\sigma_1 \in \{adm, com\}$ and $\sigma_2 = cf$. We use a standard reduction from CNF formulas $\varphi(X) = \bigwedge_{c \in C} c$ with each clause $c \in C$ a disjunction of literals from X to $F_\varphi = (A_\varphi, R_\varphi)$ given as

$$A_\varphi = \{\varphi, \bar{\varphi}\} \cup C \cup X \cup \bar{X}$$
$$R_\varphi = \{(c, \varphi) \mid c \in C\} \cup \{(x, \bar{x}), (\bar{x}, x) \mid x \in X\} \cup \{(x, c) \mid x \text{ occurs in } c\} \cup \{(\bar{x}, c) \mid \neg x \text{ occurs in } c\} \cup \{(\varphi, \bar{\varphi}), (\bar{\varphi}, \bar{\varphi})\} \cup \{(\bar{\varphi}, x), (\bar{\varphi}, \bar{x}) \mid x \in X\}.$$

We illustrate the framework F_φ (ignore the dashed part for the moment) for the CNF-formula $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee \neg x_4) \wedge (x_2 \vee x_3 \vee x_4)$.



It holds that φ is satisfiable iff there is an $S \neq \emptyset$ in $\sigma_1(F_\varphi)$ iff there is an $S \in \sigma_1(F_\varphi)$ with $\varphi \in S$ (Dimopoulos and Torres 1996). Now if $adm(F_\varphi) = com(F_\varphi) = \{\emptyset\}$ then $\sigma_1(F_\varphi)$ is clearly *cf*-realizable. On the other hand $\{\varphi\} \notin adm(F)$ (and thus $\{\varphi\} \notin com(F)$). So if there is an $S \in \sigma_1(F_\varphi)$ with $\varphi \in S$ then $\sigma_1(F_\varphi)$ is not downward closed and thus not *cf*-realizable. Thus we obtain that φ is satisfiable iff $\sigma_1(F_\varphi)$ is not *cf*-realizable.

The result for $\sigma_1 = com$ and $\sigma_2 = adm$ can be shown via an extension of above reduction to $F'_\varphi = (A'_\varphi, R'_\varphi)$ with $A'_\varphi = A_\varphi \cup \{a, a', b, b', c\}$ and $R'_\varphi = R_\varphi \cup \{(a, a'), (a', a), (a', a'), (a', c), (b, b'), (b', b), (b', b'), (b', c), (\bar{\varphi}, a), (\bar{\varphi}, b)\}$. The AF F'_φ for the example CNF is illustrated in the above figure including the dashed part.

We turn to second table and use $\Sigma_{stb} = \Sigma_{stage} \cup \{\emptyset\}$, $\Sigma_{stage} \subset \Sigma_{sem} = \Sigma_{pref}$ from our main theorems. Thus, $stb(F) \in \Sigma_\sigma$ ($\sigma \in \{stage, pref, sem\}$) iff $stb(F) \neq \emptyset$. Deciding whether an AF has a stable extension is known to be NP-complete.

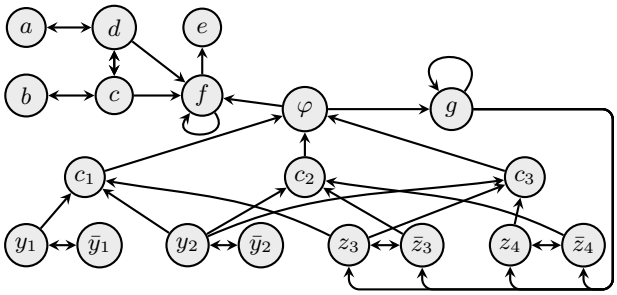
Next, we consider the coNP-hardness of recasting to *naive* semantics. Again, we extend F_φ from above and define $F^*_\varphi = (A^*_\varphi, R^*_\varphi)$ using an additional gadget (see Figure 1) via $A^*_\varphi = A_\varphi \cup \{a_1, a_2, a_3, b_1, b_2, b_3\}$ and $R^*_\varphi = \big(R_\varphi \setminus \{(\bar{\varphi}, \bar{\varphi})\}\big) \cup \{(\bar{\varphi}, a_i), (\bar{\varphi}, b_i) \mid 1 \leq i \leq 3\} \cup \{(\bar{\varphi}, \varphi)\} \cup \{(a_i, a_j), (a_i, b_i) \mid 1 \leq i, j \leq 3, i \neq j\}$. It is known that φ is satisfiable iff $\{\bar{\varphi}\} \cup C$ is not the unique preferred extension of F^*_φ (Dunne 2009) (the additional part in the AF does not affect the claim). As $\{\bar{\varphi}\} \cup C$ is also a stable extension this result extends to *stb*, *sem* and *stage* semantics. Let $\sigma \in \{stb, pref, sem, stage\}$. If φ is unsatisfiable then $\{\bar{\varphi}\} \cup C$ is the only $S \in \sigma(F^*_\varphi)$. Otherwise, for each model M of φ, there exist $S_1, S_2, S_3 \in \sigma(F^*_\varphi)$, such that $maj_3(S_1, S_2, S_3) = M \cup \overline{X \setminus M} \cup \{\varphi, b_1, b_2, b_3\}$ is a maximal conflict-free set but not admissible and thus not in $\sigma(F^*_\varphi)$. By Proposition 9 and Theorem 1, $\sigma(F^*_\varphi) \notin \Sigma_{naive}$. Hence φ is satisfiable iff $\sigma(F^*_\varphi)$ is not *naive*-realizable.

Finally, we consider the Π_2^P-entries, i.e. $\sigma_1 \in \{pref, sem\}$, $\sigma_2 \in \{stb, stage\}$. Since $\sigma_1(F) \neq \emptyset$ for any AF F, and $\Sigma_{stb} = \Sigma_{stage} \cup \{\emptyset\}$, we can stick to $\sigma_2 = stb$. Membership is by an algorithm that, given an $F = (A, R)$, disproves $\sigma_1(F) \in \Sigma_{stb}$, that is it looks for counter examples (violating tightness) from $\sigma_1(F)$ as follows: guess sets $S \subseteq A$, $\{A_s \subseteq A \mid s \in S\}$ and $a \in A \setminus B$; check $S \in \sigma_1(F)$ and $S \cup \{a\} \notin \sigma_1(F)$, and for all $s \in B$ check $A_s \in \sigma_1(F)$ and $\{a, s\} \subseteq A_s$ (for all checks, an NP-oracle is sufficient (Dunne and Bench-Capon 2002; Dvořák and Woltran 2010)). For sets passing all the checks we have that S and a contradict tightness as $S \cup \{a\} \notin \sigma_1(F)$ but for each $s \in S$, $(a, s) \in Pairs_{\sigma_1(F)}$. By Theorem 1, $\sigma_1(F) \notin \Sigma_{stb}$.

We show Π_2^P-hardness for $\sigma_1 = pref$ (*pref* semantics can be efficiently reduced to *sem* semantics (Dvořák and Woltran 2011), the result for $\sigma_1 = sem$ thus follows): Given QBF $\Phi = \forall Y \exists Z \varphi(Y, Z)$, where φ is a CNF $\bigwedge_{c \in C} c$ with each c a disjunction of literals from $X = Y \cup Z$, let $F_\Phi = (A_\Phi, R_\Phi)$ with $A_\Phi = \{\varphi, g\} \cup C \cup X \cup \bar{X} \cup \{a, b, c, d, e, f\}$ and

$$R_\Phi = \{(c, \varphi) \mid c \in C\} \cup \{(x, \bar{x}), (\bar{x}, x) \mid x \in X\} \cup \{(x, c) \mid x \text{ occurs in } c\} \cup \{(\bar{x}, c) \mid \neg x \text{ occurs in } c\} \cup \{(\varphi, g), (g, g)\} \cup \{(g, z), (g, \bar{z}) \mid z \in Z\} \cup \{(a, d), (d, a), (b, c), (c, b), (c, d), (d, c), (c, f), (d, f), (f, e), (f, f), (\varphi, f)\}$$

We illustrate F_Φ for the QBF $\Phi = \forall y_1, y_2 \exists z_3, z_4 \big((y_1 \vee y_2 \vee z_3) \wedge (y_2 \vee \neg z_3 \vee \neg z_4) \wedge (y_2 \vee z_3 \vee z_4)\big)$.



F_Φ links the reduction from (Dunne and Bench-Capon 2002) with an AF similar to Figure 2. It holds that $\mathit{pref}(F_\Phi) \in \Sigma_{\mathit{stb}}$ iff φ is contained in all $E \in \mathit{pref}(F_\Phi)$. □

7 Discussion

In this work, we initiated a study on the characteristics the sets of extensions w.r.t. a given semantics satisfy. For the semantics naive, stable, stage, preferred, and semi-stable we have an exact picture fully describing their signatures. These results also tell about the limits of global disagreement (a notion introduced in (Booth et al. 2012)) that can be modelled within AFs, e.g. our results show that preferred and semi-stable semantics are able to express more disagreement than stage semantics: $\Sigma_{\mathit{stage}} \subset \Sigma_{\mathit{pref}} = \Sigma_{\mathit{sem}}$.

Besides an exact characterization of the signature of complete semantics, future work includes an investigation of other important semantics, in particular resolution-based grounded (Baroni, Dunne, and Giacomin 2011) and cf2-semantics (Baroni, Giacomin, and Guida 2005), and an according extension of our complexity analysis. On the one hand, exact bounds for the recasting problem involving naive extensions is still missing. On the other hand, the question of how complexity may be affected when $\mathbb{S}$ is specified through some other concise form, e.g. a propositional formula whose models are exactly $S \in \mathbb{S}$ is of some interest and the subject of current investigation. We also want to explore how our results allow for pruning the search space in algorithms for abstract argumentation. Finally, since we have viewed semantics here only in an extension-based way, it would also be of high interest to extend our studies to labelling-based semantics (Caminada and Gabbay 2009).

References

Baroni, P., and Giacomin, M. 2007. On principle-based evaluation of extension-based argumentation semantics. *Artif. Intell.* 171(10-15):675–700.

Baroni, P.; Dunne, P. E.; and Giacomin, M. 2011. On the resolution-based family of abstract argumentation semantics and its grounded instance. *Artif. Intell.* 175(3-4):791–813.

Baroni, P.; Giacomin, M.; and Guida, G. 2005. SCC-Recursiveness: A general schema for argumentation semantics. *Artif. Intell.* 168(1-2):162–210.

Baumann, R., and Brewka, G. 2010. Expanding argumentation frameworks: Enforcing and monotonicity results. In *COMMA*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, 75–86. IOS Press.

Baumann, R., and Strass, H. 2014. On the maximal and average numbers of stable extensions. In *Proc. TAFA*, volume 8306 of *LNAI*, 111–126. Springer.

Baumann, R. 2012. What does it take to enforce an argument? Minimal change in abstract argumentation. In *Proc. ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, 127–132. IOS Press.

Bench-Capon, T. J. M., and Dunne, P. E. 2007. Argumentation in artificial intelligence. *Artif. Intell.* 171(10-15):619–641.

Booth, R.; Caminada, M.; Podlaszewski, M.; and Rahwan, I. 2012. Quantifying disagreement in argument-based reasoning. In *Proc. AAMAS*, 493–500. IFAAMAS.

Caminada, M., and Amgoud, L. 2007. On the evaluation of argumentation formalisms. *Artif. Intell.* 171(5-6):286–310.

Caminada, M., and Gabbay, D. M. 2009. A logical account of formal argumentation. *Studia Logica* 93(2):109–145.

Caminada, M.; Carnielli, W. A.; and Dunne, P. E. 2012. Semi-stable semantics. *J. Log. Comput.* 22(5):1207–1254.

Coste-Marquis, S.; Konieczny, S.; Mailly, J.-G.; and Marquis, P. 2013. On the revision of argumentation systems: Minimal change of arguments status. In *Proc. TAFA*.

Dimopoulos, Y., and Torres, A. 1996. Graph theoretical structures in logic programs and default theories. *Theoretical Computer Science* 170(1-2):209–244.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.* 77(2):321–357.

Dunne, P. E., and Bench-Capon, T. J. M. 2002. Coherence in finite argument systems. *Artif. Intell.* 141(1/2):187–203.

Dunne, P. E. 2009. The computational complexity of ideal semantics. *Artificial Intelligence* 173(18):1559–1591.

Dvořák, W., and Spanring, C. 2012. Comparing the expressiveness of argumentation semantics. In *Proc. COMMA*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, 261–272. IOS Press.

Dvořák, W., and Woltran, S. 2010. Complexity of semi-stable and stage semantics in argumentation frameworks. *Inf. Process. Lett.* 110(11):425–430.

Dvořák, W., and Woltran, S. 2011. On the intertranslatability of argumentation semantics. *J. Artif. Intell. Res. (JAIR)* 41:445–475.

Falappa, M. A.; García, A. J.; Kern-Isberner, G.; and Simari, G. R. 2011. On the evolving relation between belief revision and argumentation. *Knowledge Eng. Review* 26(1):35–43.

Rahwan, I., and Simari, G. R., eds. 2009. *Argumentation in Artificial Intelligence*. Springer.

Verheij, B. 1996. Two approaches to dialectical argumentation: admissible sets and argumentation stages. In *Proc. NAIC*, 357–368.

The Parameterized Complexity of Reasoning Problems Beyond NP

Ronald de Haan* and **Stefan Szeider***
Institute of Information Systems
Vienna University of Technology
dehaan@kr.tuwien.ac.at stefan@szeider.net

Abstract

Today's propositional satisfiability (SAT) solvers are extremely powerful and can be used as an efficient back-end for solving NP-complete problems. However, many fundamental problems in knowledge representation and reasoning are located at the second level of the Polynomial Hierarchy or even higher, and hence polynomial-time transformations to SAT are not possible, unless the hierarchy collapses. Recent research shows that in certain cases one can break through these complexity barriers by fixed-parameter tractable (fpt) reductions which exploit structural aspects of problem instances in terms of problem parameters.

In this paper we develop a general theoretical framework that supports the classification of parameterized problems on whether they admit such an fpt-reduction to SAT or not. We instantiate our theory by classifying the complexities of several case study problems, with respect to various natural parameters. These case studies include the consistency problem for disjunctive answer set programming and a robust version of constraint satisfaction.

1 Introduction

Over the last two decades, propositional satisfiability (SAT) has become one of the most successful and widely applied techniques for the solution of NP-complete problems. Today's SAT-solvers are extremely efficient and robust, instances with hundreds of thousands of variables and clauses can be solved routinely. In fact, due to the success of SAT, NP-complete problems have lost their scariness, as in many cases one can efficiently encode NP-complete problems to SAT and solve them by means of a SAT-solver (Gomes et al., 2008; Biere et al., 2009; Sakallah and Marques-Silva, 2011; Malik and Zhang, 2009). However, many important computational problems, most prominently in knowledge representation and reasoning, are located above the first level of the Polynomial Hierarchy (PH) and thus considered "harder" than SAT. Hence we cannot hope for polynomial-time reductions from these problems to SAT, as such transformations would cause the (unexpected) collapse of the PH.

*Supported by the European Research Council (ERC), project 239962 (COMPLEX REASON), and the Austrian Science Fund (FWF), project P26200 (Parameterized Compilation).
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Realistic problem instances are not random and often contain some kind of "hidden structure." Recent research succeeded to exploit such hidden structure to break the complexity barriers between levels of the PH, for problems that arise in disjunctive answer set programming (Fichte and Szeider, 2013) and abductive reasoning (Pfandler, Rümmele, and Szeider, 2013). The idea is to exploit problem structure in terms of a problem *parameter*, and to develop reductions to SAT that can be computed efficiently as long as the problem parameter is reasonably small. The theory of *parameterized complexity* (Downey and Fellows, 1999; Flum and Grohe, 2006; Niedermeier, 2006) provides exactly the right type of reduction suitable for this purpose, called *fixed-parameter tractable reductions*, or *fpt-reductions* for short. Now, for a suitable choice of the parameter, one can aim at developing fpt-reductions from the hard problem under consideration to SAT.

Such positive results go significantly beyond the state-of-the-art of current research in parameterized complexity. By shifting the scope from fixed-parameter tractability to fpt-reducibility (to SAT), parameters can be less restrictive and hence larger classes of inputs can be processed efficiently. Therefore, the potential for positive tractability results is greatly enlarged. In fact, there are some known reductions that, in retrospect, can be seen as fpt-reductions to SAT. A prominent example is Bounded Model Checking (Biere et al., 1999), which can be seen as an fpt-reduction from the model checking problem for linear temporal logic (LTL), which is PSPACE-complete, to SAT, where the parameter is an upper bound on the size of a counterexample. Bounded Model Checking is widely used for hardware and software verification at industrial scale (Biere, 2009).

New Contributions The aim of this paper is to establish a general theoretical framework that supports the classification of hard problems on whether they admit an fpt-reduction to SAT or not. The main contribution is the development of a new hardness theory that can be used to provide evidence that certain problems do not admit an fpt-reduction to SAT, similar to NP-hardness which provides evidence against polynomial-time tractability (Garey and Johnson, 1979) and W[1]-hardness which provides evidence against fixed-parameter tractability (Downey and Fellows, 1999).

At the center of our theory are two hierarchies of parameterized complexity classes: the $*$-k hierarchy and the k-$*$ hierarchy. We define the complexity classes in terms of weighted variants of the quantified Boolean satisfiability problem with

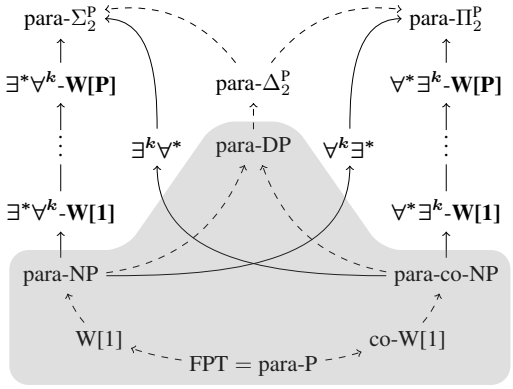


Figure 1: The parameterized complexity classes of the $*$-k and k-$*$ hierarchies (bold) in relation to existing classes. Arrows indicate inclusion relations. Dashed arrows indicate previously known relations. The classes highlighted in gray allow fpt-reductions to SAT; the other classes are unlikely to allow this.

one quantifier alternation, which is canonical for the second level of the PH. For the classes in the k-$*$ hierarchy, the (Hamming) weight of the assignment to the variables in the first quantifier block is bounded by the parameter k, the weight of the second quantifier block is unrestricted ("$*$"). For the classes in the $*$-k hierarchy it is the other way around, the weight in the second block restricted by k and the first block is unrestricted. Both hierarchies span various degrees of hardness between the classes para-NP and para-co-NP at the bottom and para-Σ_2^P at the top (para-C contains all parameterized problems that, after fpt-time preprocessing, ultimately belong to complexity class C (Flum and Grohe, 2003)). Figure 1 illustrates the relationship between the various parameterized complexity classes under consideration.

To illustrate the usefulness of our theory, we consider as a running example the fundamental problem of *answer set programming* which asks whether a disjuncive logic program has a stable model. This problem is Σ_2^P-complete (Eiter and Gottlob, 1995), and exhibits completeness or hardness for various of our complexity classes; see Table 1 for an overview. As a second case study, we will classify the complexity of a robust version of constraint satisfaction (Abramsky, Gottlob, and Kolaitis, 2013). In addition we were able to identify many other natural problems that populate our new complexity classes. We refer to a technical report corresponding to this paper which contains full proofs of all results, contains a compendium of problems, and is available on arXiv (http://arxiv.org/abs/1312.1672).

2 Preliminaries

2.1 Parameterized Complexity Theory

We introduce some core notions from parameterized complexity theory. For an in-depth treatment we refer to other sources (Downey and Fellows, 1999; Flum and Grohe, 2006; Niedermeier, 2006). A *parameterized problem* L is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. For an instance $(I, k) \in \Sigma^* \times \mathbb{N}$, we call I the *main part* and k the *parameter*. The following generalization of polynomial time computability is commonly regarded as the tractability notion of parameterized complexity theory. A parameterized problem L is *fixed-parameter tractable* if there exists a computable function f and a constant c such that there exists an algorithm that decides whether $(I, k) \in L$ in time $O(f(k)\|I\|^c)$, where $\|I\|$ denotes the size of I. Such an algorithm is called an *fpt-algorithm*, and this amount of time is called *fpt-time*. FPT is the class of all fixed-parameter tractable decision problems. If the parameter is constant, then fpt-algorithms run in polynomial time where the order of the polynomial is independent of the parameter. This provides a good scalability in the parameter in contrast to running times of the form $\|I\|^k$, which are also polynomial for fixed k, but are already impractical for, say, $k > 3$.

Parameterized complexity also offers a hardness theory, similar to the theory of NP-hardness, that allows researchers to give strong theoretical evidence that some parameterized problems are not fixed-parameter tractable. This theory is based on the *Weft hierarchy* of complexity classes FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ W[SAT] $\subseteq$ W[P], where all inclusions are believed to be strict. For a hardness theory, a notion of reduction is needed. Let $L \subseteq \Sigma^* \times \mathbb{N}$ and $L' \subseteq (\Sigma')^* \times \mathbb{N}$ be two parameterized problems. An *fpt-reduction* from L to L' is a mapping $R : \Sigma^* \times \mathbb{N} \to (\Sigma')^* \times \mathbb{N}$ from instances of L to instances of L' such that there exist some computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all $(I, k) \in \Sigma^* \times \mathbb{N}$: (i) (I, k) is a yes-instance of L if and only if $(I', k') = R(I, k)$ is a yes-instance of L', (ii) $k' \leq g(k)$, and (iii) R is computable in fpt-time. We write $L \leq_{\text{fpt}} L'$ if there is an fpt-reduction from L to L'.

The parameterized complexity classes W[t], $t \geq 1$, W[SAT] and W[P] are based on the satisfiability problems of Boolean circuits and formulas. We consider *Boolean circuits* with a single output gate. We call input nodes *variables*. We distinguish between *small* gates, with fan-in ≤ 2, and *large gates*, with fan-in > 2. The *depth* of a circuit is the length of a longest path from any variable to the output gate. The *weft* of a circuit is the largest number of large gates on any path from a variable to the output gate. We let Nodes(C) denote the set of all nodes of a circuit C. A *Boolean formula* can be considered as a Boolean circuit where all gates have fan-out ≤ 1. We adopt the usual notions of truth assignments and satisfiability of a Boolean circuit. We say that a truth assignment for a Boolean circuit has *weight* k if it sets exactly k of the variables of the circuit to true. We denote the class of Boolean circuits with depth u and weft t by $\Gamma_{t,u}$. We denote the class of all Boolean circuits by Γ, and the class of all Boolean formulas by Φ. For any class $\mathcal{C}$ of Boolean circuits, we define the following parameterized problem.

p-WSAT[$\mathcal{C}$] *Instance:* A Boolean circuit $C \in \mathcal{C}$, and an integer k. *Parameter:* k. *Question:* Does there exist an assignment of weight k that satisfies C?

We denote closure under fpt-reductions by $[\ \cdot\ ]^{\text{fpt}}$. The classes W[t] are defined by letting W[t] $= [\ \{\, p\text{-WSAT}[\Gamma_{t,u}] : u \geq 1 \,\}\ ]^{\text{fpt}}$ for all $t \geq 1$. The classes W[SAT] and W[P] are defined by letting W[SAT] $= [\ p\text{-WSAT}[\Phi]\ ]^{\text{fpt}}$ and W[P] $= [\ p\text{-WSAT}[\Gamma]\ ]^{\text{fpt}}$.

Parameterized complexity theory also offers complexity classes for problems that lie higher in the polynomial hierarchy. Let K be a classical complexity class, e.g., NP. The parameterized complexity class para-K is then defined as the class of all parameterized problems $L \subseteq \Sigma^* \times \mathbb{N}$, for some finite alphabet Σ, for which there exist an alphabet Π, a computable function $f : \mathbb{N} \to \Pi^*$, and a problem $P \subseteq \Sigma^* \times \Pi^*$ such that $P \in K$ and for all instances $(x, k) \in \Sigma^* \times \mathbb{N}$ of L we have that $(x, k) \in L$ if and only if $(x, f(k)) \in P$. Intuitively, the class para-C consists of all problems that are in C after a precomputation that only involves the parameter (Flum and Grohe, 2003). The class para-NP can also be defined via nondeterministic fpt-algorithms.

2.2 The Polynomial Hierarchy

There are many natural decision problems that are not contained in the classical complexity classes P and NP. The *Polynomial Hierarchy* (Meyer and Stockmeyer, 1972; Stockmeyer, 1976; Wrathall, 1976; Papadimitriou, 1994) contains a hierarchy of increasing complexity classes Σ_i^{P}, for all $i \geq 0$. We give a characterization of these classes based on the satisfiability problem of various classes of quantified Boolean formulas. A *quantified Boolean formula* is a formula of the form $Q_1 X_1 Q_2 X_2 \dots Q_m X_m \psi$, where each Q_i is either $\forall$ or $\exists$, the X_i are disjoint sets of propositional variables, and ψ is a Boolean formula over the variables in $\bigcup_{i=1}^{m} X_i$. The quantifier-free part of such formulas is called the *matrix* of the formula. Truth of such formulas is defined in the usual way. Let $\gamma = \{x_1 \mapsto d_1, \dots, x_n \mapsto d_n\}$ be a function that maps some variables of a formula φ to other variables or to truth values. We let $\varphi[\gamma]$ denote the application of such a substitution γ to the formula φ. We also write $\varphi[x_1 \mapsto d_1, \dots, x_n \mapsto d_n]$ to denote $\varphi[\gamma]$. For each $i \geq 1$ we define the following decision problem.

QSAT_i
Instance: A quantified Boolean formula $\varphi = \exists X_1 \forall X_2 \exists X_3 \dots Q_i X_i \psi$, where Q_i is a universal quantifier if i is even and an existential quantifier if i is odd.
Question: Is φ true?

Input formulas to the problem QSAT_i are called Σ_i^{P}-formulas. For each nonnegative integer $i \leq 0$, the complexity class Σ_i^{P} can be characterized as the closure of the problem QSAT_i under polynomial-time reductions (Stockmeyer, 1976; Wrathall, 1976). The Σ_i^{P}-hardness of QSAT_i holds already when the matrix of the input formula is restricted to 3CNF for odd i, and restricted to 3DNF for even i. Note that the class Σ_0^{P} coincides with P, and the class Σ_1^{P} coincides with NP. For each $i \geq 1$, the class Π_i^{P} is defined as co-Σ_i^{P}.

2.3 Fpt-reductions to SAT

Every problem in NP $\cup$ co-NP can be solved with one call to a SAT solver, and every problem in DP $= \{ L_1 \cap L_2 : L_1 \in$ NP, $L_2 \in$ co-NP $\}$ can be solved with two calls to a SAT solver. The Boolean Hierarchy (Cai and Hemachandra, 1986) contains all problems that can be solved with a constant number of calls to a SAT solver. On the other hand, (under complexity theoretic assumptions) there are problems in Δ_2^{P} that cannot be solved efficiently with a constant number of calls to a SAT solver. Hence, in particular, by showing that a parameterized problem is in para-NP or para-co-NP (see Section 2.1) we establish that the problem admits an fpt-reduction to SAT (see Figure 1 and Table 1).

In addition, we could consider the class of parameterized problems that can be solved by an fpt-algorithm that makes $f(k)$ many calls to a SAT solver, for some function f. This notion opens another possibility to obtain (parameterized) tractability results for problems beyond NP (cf. De Haan and Szeider, 2014).

2.4 Answer Set Programming

We will use the logic programming setting of answer set programming (ASP) (cf. Marek and Truszczynski, 1999; Brewka, Eiter, and Truszczynski, 2011) as a running example in the remainder of the paper. A *disjunctive logic program* (or simply: a *program*) P is a finite set of rules of the form $r = (a_1 \vee \dots \vee a_k \leftarrow b_1, \dots, b_m, \mathit{not}\ c_1, \dots, \mathit{not}\ c_n)$, for $k, m, n \geq 0$, where all a_i, b_j and c_l are atoms. A rule is called *disjunctive* if $k > 1$, and it is called *normal* if $k \leq 1$ (note that we only call rules with strictly more than one disjunct in the head disjunctive). A rule is called *negation-free* if $n = 0$. A program is called normal if all its rules are normal, and called negation-free if all its rules are negation-free. We let $\mathrm{At}(P)$ denote the set of all atoms occurring in P. By *literals* we mean atoms a or their negations $\mathit{not}\ a$. With $\mathrm{NF}(r)$ we denote the rule $(a_1 \vee \dots \vee a_k \leftarrow b_1, \dots, b_m)$. The *(GL) reduct* of a program P with respect to a set M of atoms, denoted P^M, is the program obtained from P by: (i) removing rules with $\mathit{not}\ a$ in the body, for each $a \in M$, and (ii) removing literals $\mathit{not}\ a$ from all other rules (Gelfond and Lifschitz, 1991). An *answer set* A of a program P is a subset-minimal model of the reduct P^A. The following decision problem is concerned with the question of whether a given program has an answer set.

ASP-CONSISTENCY
Instance: A disjunctive logic program P.
Question: Does P have an answer set?

Many implementations of answer set programming already employ SAT solving techniques, e.g., Cmodels (Giunchiglia, Lierler, and Maratea, 2006), ASSAT (Lin and Zhao, 2004), and Clasp (Gebser et al., 2007). Work has also been done on translations from ASP to SAT, both for classes of programs that allow reasoning within NP or co-NP (Ben-Eliyahu and Dechter, 1994; Fages, 1994; Lin and Zhao, 2004; Janhunen et al., 2006) and for classes of programs for which reasoning is beyond NP and co-NP (Janhunen et al., 2006; Lee and Lifschitz, 2003; Lifschitz and Razborov, 2006). We hope that our work provides new means for a theoretical study of these and related approaches to ASP.

3 Parameterizations for ASP

ASP-CONSISTENCY is Σ_2^{P}-complete in general, and can therefore (under complexity theoretic assumptions) not be reduced to SAT in polynomial time. With the aim of identifying fpt-reductions from ASP-CONSISTENCY to SAT, we consider several parameterizations.

Parameter	Complexity
normality-bd size	para-NP-complete (Fichte and Szeider, 2013)
# contingent atoms	para-co-NP-complete (Prop 1)
# contingent rules	$\exists^k\forall^*$-complete (Thms 7 and 8)
# disjunctive rules	$\exists^*\forall^k$-W[1]-hard (Thm 14)
max atom occurrence	para-Σ_2^{P}-complete (Cor 16)

Table 1: Complexity results for different parameterizations of ASP-CONSISTENCY.

Fichte and Szeider (2013) identified one parameterization of ASP-CONSISTENCY under which the problem is contained in para-NP. This parameterization is based on the notion of backdoors to normality for disjunctive logic programs. A set X of atoms is a *normality-backdoor* for a program P if deleting the atoms $x \in X$ and their negations $not\ x$ from the rules of P results in a normal program. ASP-CONSISTENCY is contained in para-NP, when parameterized by the size of a smallest normality-backdoor of the input program.

Two other parameterizations that we consider are related to atoms that must be part of any answer set of a program P. We identify a subset $\mathrm{Comp}(P)$ of *compulsory atoms*, that any answer set must include. Given a program P, we let $\mathrm{Comp}(P)$ be the smallest set such that: (i) if $(w \leftarrow not\ w)$ is a rule of P, then $w \in \mathrm{Comp}(P)$; and (ii) if $(b \leftarrow a_1, \ldots, a_n)$ is a rule of P, and $a_1, \ldots, a_n \in \mathrm{Comp}(P)$, then $b \in \mathrm{Comp}(P)$. We then let the set $\mathrm{Cont}(P)$ of *contingent atoms* be those atoms that occur in P but are not in $\mathrm{Comp}(P)$. We call a rule *contingent* if it contains contingent atoms in the head. (In fact, we could use any polynomial time computable algorithm A that computes for every program P a set $\mathrm{Comp}_A(P)$ of atoms that must be included in any answer set of P.)

The following are candidates for additional parameters that could result in fpt-reductions to SAT: (i) the number of disjunctive rules in the program (i.e., the number of rules with strictly more than one disjunct in the head); (ii) the number of contingent atoms in the program; and (iii) the number of contingent rules in the program. We will often denote the parameterized problems based on ASP-CONSISTENCY and these parameters (i) ASP-CONSISTENCY(#disj.rules), (ii) ASP-CONSISTENCY(#cont.atoms) and (iii) ASP-CONSISTENCY(#cont.rules), respectively.

The question that we would like to answer is which (if any) of these parameterizations allows an fpt-reduction to SAT. Tools from classical complexity theory seem unfit to distinguish these parameters from each other and from the parameterization by Fichte and Szeider: if the parameter values are given as part of the input, the problem remains Σ_2^{P}-complete in all cases; if we bound the parameter values by a constant, then in all cases the complexity of the problem decreases to the first-level of the PH (a proof of this can be found in the technical report). However, some of the parameterizations allow an fpt-reduction to SAT, whereas others seemingly do not.

Also the existing tools from parameterized complexity theory are unfit to distinguish between these different parameterizations of ASP-CONSISTENCY. Practically all existing parameterized complexity classes that can be used to show that an fpt-reduction is unlikely to exist (such as the classes of the W-hierarchy) are located below para-NP. Therefore, these classes do not allow us to differentiate between problems that are in para-NP and problems that are not.

However, using the parameterized complexity classes developed in this paper we will be able to make the distinction between parameterizations that allow an fpt-reduction to SAT and parameterizations that seem not to allow this. Furthermore, our theory relates the latter ones in such a way that an fpt-reduction to SAT for any of them gives us an fpt-reduction to SAT for all of them. As can be seen in Table 1, ASP-CONSISTENCY(#cont.atoms) can be fpt-reduced to SAT, whereas we have evidence that this is not possible for ASP-CONSISTENCY(#disj.rules) and ASP-CONSISTENCY(#cont.rules).

We will use ASP-CONSISTENCY together with the various parameterizations discussed above as a running example, which allows us to demonstrate the developed theoretical tools. We begin with showing a positive result for ASP-CONSISTENCY(#cont.atoms).

Proposition 1. ASP-CONSISTENCY(#cont.atoms) *is* para-co-NP-*complete.*

Proof. Hardness for para-co-NP follows from the reduction of Eiter and Gottlob (1995, Theorem 3). We show membership in para-co-NP. Let P be a program that contains k many contingent atoms. We sketch an fpt-reduction to SAT for the problem whether P has no answer set. There are 2^k candidate sets that could be an answer set, namely $N \cup \mathrm{Comp}(P)$ for each $N \subseteq \mathrm{Cont}(P)$. For each such set $M_N = N \cup \mathrm{Comp}(P)$ it can be checked in deterministic polynomial time whether M_N is a model of P^{M_N}, and it can be checked by an NP-algorithm whether M_N is not a minimal model of P^{M_N}. Therefore, by the NP-completeness of SAT, for each $N \subseteq \mathrm{Cont}(P)$, there exists a propositional formula φ_N that is satisfiable if and only if M_N is not a minimal model of P^{M_N}. All together, the statement that for no $N \subseteq \mathrm{Cont}(P)$ the set $N \cup \mathrm{Comp}(P)$ is an answer set holds true if and only if the disjunction $\bigvee_{N \subseteq \mathrm{Cont}(P)} \varphi_N$ is satisfiable. □

4 The Hierarchies $*$-k and k-$*$

We are going to define two hierarchies of parameterized complexity classes that will act as intractability classes in our hardness theory. All classes will be based on weighted variants of the satisfiability problem QSAT_2. An instance of the problem QSAT_2 has both an existential quantifier and a universal quantifier block. Therefore, there are several ways of restricting the weight of assignments. Restricting the weight of assignments to the existential quantifier block will result in the k-$*$ hierarchy, and restricting the weight of assignments to the universal quantifier block will result in the $*$-k hierarchy. The two hierarchies are based on the following two parameterized decision problems. Let $\mathcal{C}$ be a class of Boolean circuits. The problem $\exists^k\forall^*$-$\mathrm{WSAT}(\mathcal{C})$ provides the foundation for the k-$*$ hierarchy.

$\exists^k\forall^*$-WSAT($\mathcal{C}$)
Instance: A Boolean circuit $C \in \mathcal{C}$ over two disjoint sets X and Y of variables, and an integer k.
Parameter: k.
Question: Does there exist a truth assignment α to X with weight k such that for all truth assignments β to Y the assignment $\alpha \cup \beta$ satisfies C?

Similarly, the problem $\exists^*\forall^k$-WSAT($\mathcal{C}$) provides the foundation for the $*$-k hierarchy.

$\exists^*\forall^k$-WSAT($\mathcal{C}$)
Instance: A Boolean circuit $C \in \mathcal{C}$ over two disjoint sets X and Y of variables, and an integer k.
Parameter: k.
Question: Does there exist a truth assignment α to X such that for all truth assignments β to Y with weight k the assignment $\alpha \cup \beta$ satisfies C?

For convenience, instances to these two problems consisting of a circuit C over sets X and Y of variables and an integer k, we will denote by $(\exists X.\forall Y.C, k)$. We now define the following parameterized complexity classes, that together form the k-$*$ hierarchy. We let $\exists^k\forall^*$-W[t] $= [\, \{\, \exists^k\forall^*\text{-WSAT}(\Gamma_{t,u}) : u \geq 1 \,\} \,]^{\text{fpt}}$, we let $\exists^k\forall^*$-W[SAT] $= [\, \exists^k\forall^*\text{-WSAT}(\Phi) \,]^{\text{fpt}}$, and we let $\exists^k\forall^*$-W[P] $= [\, \exists^k\forall^*\text{-WSAT}(\Gamma) \,]^{\text{fpt}}$.

We define the classes of the $*$-k hierarchy similarly. We let $\exists^*\forall^k$-W[t] $= [\, \{\, \exists^*\forall^k\text{-WSAT}(\Gamma_{t,u}) : u \geq 1 \,\} \,]^{\text{fpt}}$, we let $\exists^*\forall^k$-W[SAT] $= [\, \exists^*\forall^k\text{-WSAT}(\Phi) \,]^{\text{fpt}}$, and we let $\exists^*\forall^k$-W[P] $= [\, \exists^*\forall^k\text{-WSAT}(\Gamma) \,]^{\text{fpt}}$. Note that these definitions are analogous to those of the parameterized complexity classes of the W-hierarchy (Downey and Fellows, 1999).

We can define dual classes for each of the parameterized complexity classes in the k-$*$ and $*$-k hierarchies. These co-classes are based on problems complementary to the problems $\exists^k\forall^*$-WSAT and $\exists^*\forall^k$-WSAT, i.e., these problems have as yes-instances exactly the no-instances of $\exists^k\forall^*$-WSAT and $\exists^*\forall^k$-WSAT, respectively. Equivalently, these complementary problems can be considered as variants of $\exists^k\forall^*$-WSAT and $\exists^*\forall^k$-WSAT where the existential and universal quantifiers are swapped, and are therefore denoted with $\forall^k\exists^*$-WSAT and $\forall^*\exists^k$-WSAT. We use a similar notation for the dual complexity classes, e.g., we denote co-$\exists^*\forall^k$-W[t] by $\forall^*\exists^k$-W[t].

5 The Class $\exists^k\forall^*$

In this section, we consider the k-$*$ hierarchy. It turns out that this hierarchy collapses entirely into a single parameterized complexity class. This class we will denote by $\exists^k\forall^*$. As we will see, the class $\exists^k\forall^*$ turns out to be quite robust. We start this section with showing that that the k-$*$ hierarchy collapses. We discuss how this class is related to existing parameterized complexity classes, and we show how it can be used to show the intractability of a variant of the answer set existence problem whose complexity the existing theory cannot classify properly.

5.1 Collapse of the k-$*$ hierarchy

Theorem 2 (Collapse of the k-$*$ hierarchy). $\exists^k\forall^*$-W[1] $= \exists^k\forall^*$-W[2] $= \ldots = \exists^k\forall^*$-W[SAT] $= \exists^k\forall^*$-W[P].

Proof. Since by definition $\exists^k\forall^*$-W[1] $\subseteq \exists^k\forall^*$-W[2] $\subseteq \ldots \subseteq \exists^k\forall^*$-W[P], it suffices to show that $\exists^k\forall^*$-W[P] $\subseteq \exists^k\forall^*$-W[1]. We show this by giving an fpt-reduction from $\exists^k\forall^*$-WSAT(Γ) to $\exists^k\forall^*$-WSAT(3DNF). Since 3DNF $\subseteq \Gamma_{1,3}$, this suffices. We remark that this reduction is based on the standard Tseitin transformation that transforms arbitrary Boolean formulas into 3CNF by means of additional variables.

Let (φ, k) be an instance of $\exists^k\forall^*$-WSAT(Γ) with $\varphi = \exists X.\forall Y.C$. Assume without loss of generality that C contains only binary conjunctions and negations. Let o denote the output gate of C. We construct an instance (φ', k) of $\exists^k\forall^*$-WSAT(3DNF) as follows. The formula φ' will be over the set of variables $X \cup Y \cup Z$, where $Z = \{\, z_r : r \in \text{Nodes}(C) \,\}$. For each $r \in \text{Nodes}(C)$, we define a subformula χ_r. We distinguish three cases. If $r = r_1 \wedge r_2$, then we let $\chi_r = (z_r \wedge \neg z_{r_1}) \vee (z_r \wedge \neg z_{r_2}) \vee (z_{r_1} \wedge z_{r_2} \wedge \neg z_r)$. If $r = \neg r_1$, then we let $\chi_r = (z_r \wedge z_{r_1}) \vee (\neg z_r \wedge \neg z_{r_1})$. If $r = w$, for some $w \in X \cup Y$, then we let $\chi_r = (z_r \wedge \neg w) \vee (\neg z_r \wedge w)$. Now we define $\varphi' = \exists X.\forall Y \cup Z.\psi$, where $\psi = \bigvee_{r \in \text{Nodes}(C)} \chi_r \vee z_o$. It is straightforward to verify that this reduction is correct. □

As mentioned above, in order to simplify notation, we will use $\exists^k\forall^*$ to denote the class $\exists^k\forall^*$-W[1] $= \ldots = \exists^k\forall^*$-W[P]. Also, we will denote $\exists^k\forall^*$-WSAT(Γ) by $\exists^k\forall^*$-WSAT. We make some observations about the relation of $\exists^k\forall^*$ to existing parameterized complexity classes. It is straightforward to see that $\exists^k\forall^* \subseteq$ para-Σ_2^{P}. In polynomial time, any formula $\exists X.\forall Y.\psi$ can be transformed into a Σ_2^{P}-formula that is true if and only if for some assignment α of weight k to the variables X the formula $\forall Y.\psi[\alpha]$ is true. Trivially, para-co-NP $\subseteq \exists^k\forall^*$. To summarize, we obtain the following inclusions: para-co-NP $\subseteq \exists^k\forall^* \subseteq \Sigma_2^{\text{P}}$, and para-NP $\subseteq \forall^k\exists^* \subseteq \Pi_2^{\text{P}}$. This immediately leads to the following result.

Proposition 3. *If* $\exists^k\forall^* \subseteq$ para-NP*, then* NP $=$ co-NP.

A natural question to ask is whether para-NP $\subseteq \exists^k\forall^*$. The following result indicates that this is unlikely.

Proposition 4. *If* para-NP $\subseteq \exists^k\forall^*$*, then* NP $=$ co-NP.

Proof (sketch). Let SAT be the language of satisfiable propositional formulas, and UNSAT the language of unsatisfiable propositional formulas. The parameterized problem $P = \{\, (\varphi, 1) : \varphi \in \text{SAT} \,\}$ is in para-NP. Since the parameter value is constant for all instances of P, an fpt-reduction from P to $\exists^k\forall^*$-WSAT can be transformed into an polynomial time reduction from SAT to UNSAT. □

This implies that $\exists^k\forall^*$ is very likely to be a strict subset of para-Σ_2^{P}.

Corollary 5. *If* $\exists^k\forall^* =$ para-Σ_2^{P}*, then* NP $=$ co-NP.

The following result shows another way in which the class $\exists^k\forall^*$ relates to the existing complexity class co-NP. Let P be a parameterized decision problem, and let $c \geq 1$ be an integer. We say that the *c-th slice of P*, denoted P_c, is the (unparameterized) decision problem $\{\, x : (x, c) \in P \,\}$.

Proposition 6. *Let P be a parameterized problem complete for $\exists^k\forall^*$, and let $c \geq 1$ be an integer. Then P_c is in* co-NP*. Moreover, there exists some integer $d \geq 1$ such that $P_1 \cup \cdots \cup P_d$ is* co-NP*-complete.*

A proof of this statement can be found in the technical report.

5.2 Answer set programming and completeness for the k-$*$ hierarchy

Now that we defined this new intractability class $\exists^k\forall^*$ and that we have some basic results about it in place, we are able to prove the intractability of a variant of our running example problem. In fact, we show that one variant of our running example is complete for the class $\exists^k\forall^*$.

Theorem 7. ASP-CONSISTENCY(#cont.rules) *is $\exists^k\forall^*$-hard.*

Proof. We give an fpt-reduction from $\exists^k\forall^*$-WSAT(3DNF). This reduction is a parameterized version of a reduction of Eiter and Gottlob (1995, Theorem 3). Let (φ, k) be an instance of $\exists^k\forall^*$-WSAT(3DNF), where $\varphi = \exists X.\forall Y.\psi$, $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, $\psi = \delta_1 \vee \cdots \vee \delta_u$, and $\delta_\ell = l^\ell_1 \wedge l^\ell_2 \wedge l^\ell_3$ for each $1 \leq \ell \leq u$. We construct a disjunctive program P. We consider the sets X and Y of variables as atoms. In addition, we introduce fresh atoms $v_1, \ldots, v_n$, $z_1, \ldots, z_m$, w, and x^j_i for all $1 \leq j \leq k$, $1 \leq i \leq n$. We let P consist of the following rules:

$$
\begin{array}{lll}
x^j_1 \vee \cdots \vee x^j_n \leftarrow & \text{for } 1 \leq j \leq k; & (1)\\
\leftarrow x^j_i, x^{j'}_i & \text{for } 1 \leq i \leq n, 1 \leq j < j' \leq k; & (2)\\
y_i \vee z_i \leftarrow, \quad w \leftarrow y_i, z_i & \text{for } 1 \leq i \leq m; & (3)\\
y_i \leftarrow w, \quad z_i \leftarrow w & \text{for } 1 \leq i \leq m; & (4)\\
x_i \leftarrow w, \quad v_i \leftarrow w & \text{for } 1 \leq i \leq n; & (5)\\
x_i \leftarrow x^j_i & \text{for } 1 \leq i \leq n, 1 \leq j \leq k; & (6)\\
v_i \leftarrow not\ x^1_i, \ldots, not\ x^k_i & \text{for } 1 \leq i \leq n; & (7)\\
w \leftarrow \sigma(l^\ell_1), \sigma(l^\ell_2), \sigma(l^\ell_3) & \text{for } 1 \leq \ell \leq u; & (8)\\
w \leftarrow not\ w. & & (9)
\end{array}
$$

Here we let $\sigma(x_i) = x_i$ and $\sigma(\neg x_i) = v_i$ for each $1 \leq i \leq n$; and we let $\sigma(y_j) = y_j$ and $\sigma(\neg y_j) = z_j$ for each $1 \leq j \leq m$. Intuitively, v_i corresponds to $\neg x_i$, and z_j corresponds to $\neg y_j$. The main difference with the reduction of Eiter and Gottlob is that we use the rules in (1), (2), (6) and (7) to let the variables x_i and v_i represent an assignment of weight k to the variables in X. The rules in (5) ensure that the atoms v_i and x_i are compulsory. It is straightforward to verify that $\text{Comp}(P) = \{w\} \cup \{\, x_i, v_i : 1 \leq i \leq n \,\} \cup \{\, y_i, z_i : 1 \leq i \leq m \,\}$. Notice that P has exactly k contingent rules, namely the rules in (1). A full proof that $(\varphi, k) \in \exists^k\forall^*$-WSAT if and only if P has an answer set can be found in the technical report. □

Theorem 8. ASP-CONSISTENCY(#cont.rules) *is in $\exists^k\forall^*$.*

Proof. We show membership in $\exists^k\forall^*$ by reducing ASP-CONSISTENCY(#cont.rules) to $\exists^k\forall^*$-WSAT. Let P be a program, where $r_1, \ldots, r_k$ are the contingent rules of P and where $\text{At}(P) = \{d_1, \ldots, d_n\}$. We construct a quantified Boolean formula $\varphi = \exists X.\forall Y \cup Z \cup W.\psi$ such that $(\varphi, k) \in \exists^k\forall^*$-WSAT if and only if P has an answer set.

In order to do so, we firstly construct a Boolean formula $\psi_P(z_1, \ldots, z_n, z'_1, \ldots, z'_n)$ (or, for short: ψ_P) over variables $z_1, \ldots, z_n, z'_1, \ldots, z'_n$ such that for any $M \subseteq \text{At}(P)$ and any $M' \subseteq \text{At}(P)$ holds that M is a model of $P^{M'}$ if and only if $\psi_P[\alpha_M \cup \alpha_{M'}]$ evaluates to true, where $\alpha_M : \{z_1, \ldots, z_n\} \to \{0, 1\}$ is defined by letting $\alpha_M(z_i) = 1$ if and only if $d_i \in M$, and $\alpha_{M'} : \{z'_1, \ldots, z'_n\} \to \{0, 1\}$ is defined by letting $\alpha_{M'}(z'_i) = 1$ if and only if $d_i \in M'$, for all $1 \leq i \leq n$. We define $\psi_P = \bigwedge_{r \in P}(\psi^1_r \vee \psi^2_r)$, where $\psi^1_r = (z'_{i^3_1} \vee \cdots \vee z'_{i^3_c})$ and $\psi^2_r = ((z_{i^1_1} \vee \cdots \vee z_{i^1_a}) \leftarrow (z_{i^2_1} \wedge \cdots \wedge z_{i^2_b}))$ for $r = (d_{i^1_1} \vee \cdots \vee d_{i^1_a} \leftarrow d_{i^2_1}, \ldots, d_{i^2_b}, not\ d_{i^3_1}, \ldots, not\ d_{i^3_c})$. It is easy to verify that ψ_P satisfies the required property.

We now introduce the set X of existentially quantified variables of φ. For each contingent rule r_i of P we let $a^i_1, \ldots, a^i_{\ell_i}$ denote the atoms that occur in the head of r_i. For each r_i, we introduce variables $x^i_0, x^i_1, \ldots, x^i_{\ell_i}$, i.e., $X = \{\, x^i_j : 1 \leq i \leq k, 0 \leq j \leq \ell_i \,\}$. Furthermore, for each atom d_i, we add universally quantified variables y_i, z_i and w_i, i.e., $Y = \{\, y_i : 1 \leq i \leq n \,\}$, $Z = \{\, z_i : 1 \leq i \leq n \,\}$, and $W = \{\, w_i : 1 \leq i \leq n \,\}$.

We then construct ψ as follows:

$$
\begin{array}{rl}
\psi = & \psi_X \wedge \left(\psi^1_Y \vee \psi_W \vee \psi_{\text{min}}\right) \wedge \left(\psi^1_Y \vee \psi^2_Y\right);\\
\psi_X = & \displaystyle\bigwedge_{1 \leq i \leq k} \left(\bigvee_{0 \leq j \leq \ell_i} x^i_j \wedge \bigwedge_{0 \leq j < j' \leq \ell_i} (\neg x^i_j \vee \neg x^i_{j'}) \right);\\
\psi^1_Y = & \displaystyle\bigvee_{\substack{1 \leq i \leq k\\ 1 \leq j \leq \ell_i}} \psi^{i,j}_y \vee \bigvee_{d_i \in \text{Cont}(P)} \psi^{d_i}_y \vee \bigvee_{d_i \in \text{Comp}(P)} \neg y_i;\\
\psi^{d_m}_y = & (y_m \wedge \neg x^{i_1}_{j_1} \wedge \cdots \wedge \neg x^{i_u}_{j_u})\\
& \text{if } \{\, x^i_j : 1 \leq i \leq k, 1 \leq j \leq \ell_i, a^i_j = d_m \,\} =\\
& \{a^{i_1}_{j_1}, \ldots, a^{i_u}_{j_u}\}, \text{ and}\\
\psi^{d_m}_y = & \bot\\
& \text{if } \{\, x^i_j : 1 \leq i \leq k, 1 \leq j \leq \ell_i, a^i_j = d_m \,\} = \emptyset;\\
\psi^{i,j}_y = & (x^i_j \wedge \neg y_m) \qquad \text{where } a^i_j = d_m;\\
\psi^2_Y = & \psi_P(y_1, \ldots, y_n, y_1, \ldots, y_n);\\
\psi_W = & \displaystyle\bigvee_{1 \leq i \leq n} (w_i \leftrightarrow (y_i \leftrightarrow z_i));\\
\psi_{\text{min}} = & \psi^1_{\text{min}} \vee \psi^2_{\text{min}} \vee \psi^3_{\text{min}};\\
\psi^1_{\text{min}} = & \displaystyle\bigvee_{1 \leq i \leq n} (z_i \wedge \neg y_i);\\
\psi^2_{\text{min}} = & (\neg w_1 \wedge \cdots \wedge \neg w_m); \text{ and}\\
\psi^3_{\text{min}} = & \neg\psi_P(z_1, \ldots, z_n, y_1, \ldots, y_n).
\end{array}
$$

The idea behind this construction is the following. The variables in X represent guessing at most one atom in the head of each contingent rule to be true. Such a guess represents a possible answer set $M \subseteq \text{At}(P)$ (a proof of this can be found in the technical report). The formula ψ_X ensures that for each $1 \leq i \leq k$, exactly one x^i_j is set to true. The formula ψ^1_Y filters out every assignment in which the variables Y are not set corresponding to M. The formula ψ^2_Y filters out every assignment corresponding to a candidate $M \subseteq \text{At}(P)$ such that $M \not\models P$. The formula ψ_W filters out every assignment such that w_i is not set to the value $(y_i \text{ XOR } z_i)$. The formula ψ^1_{min} filters out every assignment where the variables Z correspond to a set M' such that $M' \not\subseteq M$. The formula ψ^2_{min} filters out every assignment where the variables Z correspond to the set M, by referring to the variables w_i. The formula ψ^3_{min}, finally, ensures that in every remaining assignment, the

variables Z do not correspond to a set $M' \subseteq M$ such that $M' \models P$. A full proof that P has an answer set if and only if $(\varphi, k) \in \exists^k\forall^*$-WSAT can be found in the technical report. □

Corollary 9. ASP-CONSISTENCY(#cont.rules) *is* $\exists^k\forall^*$-*complete.*

6 The $*$-k Hierarchy

We now turn our attention to the $*$-k hierarchy. Unlike in the k-$*$ hierarchy, in the canonical quantified Boolean satisfiability problems of the $*$-k hierarchy, we cannot add auxiliary variables to the second quantifier block whose truth assignment is not restricted. Therefore, because of similarity to the W-hierarchy, we believe that the classes of the $*$-k hierarchy are distinct. We will mainly focus on the first level of the $*$-k hierarchy. We begin with proving some basic properties. The main result in this section is a normalization result for $\forall^*\exists^k$-W[1].

Proposition 10. *If* para-co-NP $\subseteq \exists^*\forall^k$-W[P]*, then* NP $=$ co-NP.

Proof (sketch). With an argument similar to the one in the proof of Proposition 4, a polynomial-time reduction from UNSAT to SAT can be constructed. An additional technical observation needed for this case is that SAT is in NP also when the input is a Boolean circuit. □

Next, we show that the problem $\exists^*\forall^k$-WSAT is already $\exists^*\forall^k$-W[1]-hard when the input circuits are restricted to formulas in c-DNF, for any constant $c \geq 2$. In order to make our life easier, we switch our perspective to the co-problem $\forall^*\exists^k$-WSAT when stating and proving the following results. Note that the proofs of the following results make heavy use of the original normalization proof for the class W[1] by Downey and Fellows (1995; 1999).

Lemma 11. *For any* $u \geq 1$, $\forall^*\exists^k$-WSAT$(\Gamma_{1,u}) \leq_{\text{fpt}} \forall^*\exists^k$-WSAT$(s$-CNF$)$*, where* $s = 2^u + 1$.

Proof (sketch). The reduction is completely analogous to the reduction used in the proof of Downey and Fellows (1995, Lemma 2.1), where the presence of universally quantified variables is handled in four steps. In Steps 1 and 2, in which only the form of the circuit is modified, no changes are needed. In Step 3, universally quantified variables can be handled exactly as existentially quantified variables. Step 4 can be performed with only a slight modification, the only difference being that universally quantified variables appearing in the input circuit will also appear in the resulting clauses that verify whether a given product-of-sums or sum-of-products is satisfied. It is straightforward to verify that this reduction with the mentioned modifications works for our purposes. □

Theorem 12. $\forall^*\exists^k$-WSAT(2CNF) *is* $\forall^*\exists^k$-W[1]*-complete.*

Proof (sketch). Clearly $\forall^*\exists^k$-WSAT(2CNF) is in $\forall^*\exists^k$-W[1], since a 2CNF formula can be considered as a constant-depth circuit of weft 1. To show that $\forall^*\exists^k$-WSAT(2CNF) is $\forall^*\exists^k$-W[1]-hard, we give an fpt-reduction from $\forall^*\exists^k$-WSAT$(\Gamma_{1,u})$ to $\forall^*\exists^k$-WSAT(2CNF), for arbitrary $u \geq 1$. By Lemma 11, we know that we can reduce $\forall^*\exists^k$-WSAT$(\Gamma_{1,u})$ to $\forall^*\exists^k$-WSAT$(s$-CNF$)$, for $s = 2^u + 1$. We continue the reduction in multiple steps. In each step, we let C denote the circuit resulting from the previous step, and we let Y denote the universally quantified and X the existentially quantified input variables of C, and we let k denote the parameter value. We only briefly describe the last two steps, since these are completely analogous to constructions in the work of Downey and Fellows (1999).

Step 1: contracting the universally quantified variables. This step transforms C into a CNF formula C' such that each clause contains at most one variable in Y such that (C, k) is a yes-instance if and only if (C', k) is a yes-instance. We introduce new universally quantified variables Y' containing a variable y'_A for each set A of literals over Y of size at least 1 and at most s. Now, it is straightforward to construct a set D of polynomially many ternary clauses over Y and Y' such that the following property holds. An assignment α to $Y \cup Y'$ satisfies D if and only if for each subset $A = \{l_1, \ldots, l_b\}$ of literals over Y it holds that $\alpha(l_1) = \alpha(l_2) = \ldots = \alpha(l_b) = 1$ if and only if $\alpha(y'_A) = 1$. Note that we do not directly add the set D of clauses to the formula C'.

We introduce $k - 1$ many new existentially quantified variables $x^\star_1, \ldots, x^\star_{k-1}$. We add binary clauses to C' that enforce that the variables $x^\star_1, \ldots, x^\star_{k-1}$ all get the same truth assignment. Also, we add binary clauses to C' that enforce that each $x \in X$ is set to false if $x^\star_1$ is set to true.

We introduce $|D|$ many existentially quantified variables, including a variable x''_d for each clause $d \in D$. For each $d \in D$, we add the following clauses to C'. Let $d = (l_1, l_2, l_3)$, where each l_i is a literal over $Y \cup Y'$. We add the clauses $(\neg x''_d \vee \neg l_1)$, $(\neg x''_d \vee \neg l_2)$ and $(\neg x''_d \vee \neg l_3)$, enforcing that the clause d cannot be satisfied if x''_d is set to true.

We then modify the clauses of C as follows. Let $c = (l^x_1, \ldots, l^x_{s_1}, l^y_1, \ldots, l^y_{s_2})$ be a clause of C, where $l^x_1, \ldots, l^x_{s_1}$ are literals over X, and $l^y_1, \ldots, l^y_{s_2}$ are literals over Y. We replace c by the clause $(l^x_1, \ldots, l^x_{s_1}, x^\star_1, y'_B)$, where $B = \{l^y_1, \ldots, l^y_{s_2}\}$. Clauses c of C that contain no literals over the variables Y remain unchanged.

The idea of this reduction is the following. If $x^\star_1$ is set to true, then exactly one of the variables x''_d must be set to true, which can only result in an satisfying assignment if the clause $d \in D$ is not satisfied. Therefore, if an assignment α to the variables $Y \cup Y'$ does not satisfy D, there is a satisfying assignment of weight k that sets both $x^\star_1$ and x''_d to true, for some $d \in D$ that is not satisfied by α. Otherwise, we know that the value α assigns to variables y'_A corresponds to the value α assigns to $\bigwedge_{a \in A} a$, for $A \subseteq \text{Lit}(Y)$. Then any satisfying assignments of weight k for C is also a satisfying assignments of weight k for C'.

Step 2: making C antimonotone in X. This step transforms C into a circuit C' that has only negative occurrences of existentially quantified variables, and transforms k into k' depending only on k, such that (C, k) is a yes-instance if and only if (C', k') is a yes-instance. The reduction is completely analogous to the reduction in the proof of Downey and Fellows (1999, Theorem 10.6).

Step 3: contracting the existentially quantified variables. This step transforms C into a circuit C' in CNF

that contains only clauses with two variables in X and no variables in Y and clauses with one variable in X and one variable in Y, and transforms k into k' depending only on k, such that (C, k) is a yes-instance if and only if (C', k') is a yes-instance. The reduction is completely analogous to the reduction in the proof of Downey and Fellows (1999, Theorem 10.7). □

Corollary 13. *For any fixed integer $r \geq 2$, the problem $\exists^*\forall^k$-WSAT(r-DNF) is $\exists^*\forall^k$-W[1]-complete.*

6.1 Answer set programming and hardness for the $*$-k hierarchy

We now turn to another variant of our running example problem.

Theorem 14. ASP-CONSISTENCY(#disj.rules) *is $\exists^*\forall^k$-W[1]-hard.*

Proof. We give an fpt-reduction from $\exists^*\forall^k$-WSAT(3DNF), which we know to be $\exists^*\forall^k$-W[1]-hard from Corollary 13. This is, like the reduction in the proof of Theorem 7 above, a parameterized version of a reduction of Eiter and Gottlob (1995, Theorem 3). Let (φ, k) be an instance of $\exists^*\forall^k$-WSAT(3DNF), where $\varphi = \exists X.\forall Y.\psi$, $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, $\psi = \delta_1 \vee \cdots \vee \delta_u$, and $\delta_\ell = l^\ell_1 \wedge l^\ell_2 \wedge l^\ell_3$ for each $1 \leq \ell \leq u$. By step 2 in the proof of Theorem 12, we may assume without loss of generality that all universally quantified variables $y_1, \ldots, y_m$ occur only positively in ψ. We construct a disjunctive program P. We consider the variables X and Y as atoms. In addition, we introduce fresh atoms $v_1, \ldots, v_n, w$, and y^j_i for all $1 \leq j \leq k$, $1 \leq i \leq m$. We let P consist of the following rules:

$$x_i \leftarrow \mathit{not}\ v_i \qquad \text{for } 1 \leq i \leq n; \quad (10)$$
$$v_i \leftarrow \mathit{not}\ x_i \qquad \text{for } 1 \leq i \leq n; \quad (11)$$
$$y^j_1 \vee \cdots \vee y^j_m \leftarrow \qquad \text{for } 1 \leq j \leq k; \quad (12)$$
$$y_i \leftarrow y^j_i \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq k; \quad (13)$$
$$y^j_i \leftarrow w \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq k; \quad (14)$$
$$w \leftarrow y^j_i, y^{j'}_i \qquad \text{for } 1 \leq i \leq m, 1 \leq j < j' \leq k; \quad (15)$$
$$w \leftarrow \sigma(l^\ell_1), \sigma(l^\ell_2), \sigma(l^\ell_3) \qquad \text{for } 1 \leq \ell \leq u; \quad (16)$$
$$w \leftarrow \mathit{not}\ w. \quad (17)$$

Here we let $\sigma(\neg x_i) = v_i$ for each $1 \leq i \leq n$; we let $\sigma(x_i) = x_i$ for each $1 \leq i \leq n$; and we let $\sigma(y_i) = y_i$ for each $1 \leq i \leq m$. Intuitively, v_i corresponds to $\neg x_i$. The main difference with the reduction of Eiter and Gottlob is that we use the rules in (12)–(15) to let the variables y_i represent an assignment of weight k to the variables in Y. Note that P has k disjunctive rules, namely the rules (12). A full proof that $(\varphi, k) \in \exists^*\forall^k$-WSAT if and only if P has an answer set can be found in the technical report. □

This hardness result holds even for the case where each atom occurs only a constant number of times in the input program. In order to show this, we consider the following polynomial-time transformation on disjunctive logic programs. Let P be an arbitrary program, and let x be an atom of P that occurs $\ell \geq 3$ many times. We introduce new atoms x_j for all $1 \leq j \leq \ell$. We replace each occurrence of x in P by a unique atom x_j. Then, to P, we add the rules $(x_{j+1} \leftarrow x_j)$, for all $1 \leq j < \ell$, and the rule $(x_1 \leftarrow x_\ell)$. We call the resulting program P'. It is straightforward to verify that P has an answer set if and only if P' has an answer set.

Proposition 15. ASP-CONSISTENCY(#disj.rules) *is $\exists^*\forall^k$-W[1]-hard, even when each atom occurs at most 3 times.*

Proof. The transformation described above can be repeatedly applied to ensure that each atom occurs at most 3 times. Since this transformation does not introduce new disjunctive rules, the result follows by Theorem 14. □

Repeated application of the described transformation also gives us the following result.

Corollary 16. ASP-CONSISTENCY(max.atom.occ.) *is* para-Σ^{P}_2*-complete.*

7 Robust Constraint Satisfaction

Next, we consider another application of our hardness theory to a reasoning problem that originates in the domain of knowledge representation. We consider the class of robust constraint satisfaction problems, introduced recently by Gottlob (2012) and Abramsky, Gottlob, and Kolaitis (2013). These problems are concerned with the question of whether every partial assignment of a particular size can be extended to a full solution, in the setting of constraint satisfaction problems. As we will see, a natural parameterized variant of this class of problems is complete for the class $\forall^k\exists^*$.

A *CSP instance* N is a triple (X, D, C), where X is a finite set of *variables*, the *domain* D is a finite set of *values*, and C is a finite set of *constraints*. Each constraint $c \in C$ is a pair (S, R), where $S = \mathrm{Var}(c)$, the *constraint scope*, is a finite sequence of distinct variables from X, and R, the *constraint relation*, is a relation over D whose arity matches the length of S, i.e., $R \subseteq D^r$ where r is the length of S.

Let $N = (X, D, C)$ be a CSP instance. A *partial instantiation* of N is a mapping $\alpha : X' \to D$ defined on some subset $X' \subseteq X$. We say that α *satisfies* a constraint $c = ((x_1, \ldots, x_r), R) \in C$ if $\mathrm{Var}(c) \subseteq X'$ and $(\alpha(x_1), \ldots, \alpha(x_r)) \in R$. If α satisfies all constraints of N then it is a *solution* of N. We say that α *violates* a constraint $c = ((x_1, \ldots, x_r), R) \in C$ if there is no extension β of α defined on $X' \cup \mathrm{Var}(c)$ such that $(\beta(x_1), \ldots, \beta(x_r)) \in R$.

Let k be a nonnegative integer. We say that a CSP instance $N = (X, D, C)$ is *k-robustly satisfiable* if for each instantiation $\alpha : X' \to D$ defined on some subset $X' \subseteq X$ of k many variables (i.e., $|X'| = k$) that does not violate any constraint in C, it holds that α can be extended to a solution for the CSP instance (X, D, C). Now consider the following parameterized problem.

ROBUST-CSP-SAT
Instance: A CSP instance (X, D, C), and a nonnegative integer k.
Parameter: k.
Question: Is (X, D, C) k-robustly satisfiable?

We show that this problem is complete for $\forall^k\exists^*$.

Theorem 17. ROBUST-CSP-SAT *is in $\forall^k\exists^*$.*

Proof. We give an fpt-reduction from ROBUST-CSP-SAT to $\forall^k\exists^*$-WSAT. Let (X, D, C, k) be an instance of ROBUST-CSP-SAT, where (X, D, C) is a CSP instance, $X = \{x_1, \ldots, x_n\}$, $D = \{d_1, \ldots, d_m\}$, and k is an integer. We

construct an instance (φ, k) of $\forall^k\exists^*$-WSAT. For the formula φ, we use propositional variables $Z = \{ z^i_j : 1 \leq i \leq n, 1 \leq j \leq m \}$ and $Y = \{ y^i_j : 1 \leq i \leq n, 1 \leq j \leq m \}$. Intuitively, the variables z^i_j will represent an arbitrary assignment α that assigns values to k variables in X. Any variable z^i_j represents that variable x_i gets assigned value d_j. The variables y^i_j will represent the solution β that extends the arbitrary assignment α. Similarly, any variable y^i_j represents that variable x_i gets assigned value d_j. We then let $\varphi = \forall Z.\exists Y.\psi$ with $\psi = (\psi^Z_{\text{proper}} \wedge \neg\psi^Z_{\text{violate}}) \rightarrow (\psi^{Y,Z}_{\text{corr}} \wedge \psi^Y_{\text{proper}} \wedge \bigwedge_{c \in C} \psi^Y_c)$. We will describe the subformulas of φ below, as well as the intuition behind them. We start with the formula ψ^Z_{proper}. This formula represents whether for each variable x_i at most one value is chosen for the assignment α. We let $\psi^Z_{\text{proper}} = \bigwedge_{1 \leq i \leq n} \bigwedge_{1 \leq j < j' \leq m} (\neg z^i_j \vee \neg z^i_{j'})$. Next, we consider the formula ψ^Z_{violate}. This subformula encodes whether the assignment α violates some constraint $c \in C$. We let $\psi^Z_{\text{violate}} = \bigvee_{c=(S,R) \in C} \bigwedge_{\overline{d} \in R} \bigvee_{z \in \Psi^{\overline{d},c}}$, where we define the set $\Psi^{\overline{d},c} \subseteq Z$ as follows. Let $c = ((x_{i_1}, \ldots, x_{i_r}), R) \in C$ and $\overline{d} = (d_{j_1}, \ldots, d_{j_r}) \in R$. Then we let $\Psi^{\overline{d},c} = \{ z^{i_\ell}_j : 1 \leq \ell \leq r, j \neq j_\ell \}$. Intuitively, the set $\Psi^{\overline{d},c}$ contains the variables z^i_j that represent those variable assignments in α that prevent that β satisfies c by assigning $\text{Var}(c)$ to $\overline{d}$. Then, the formula ψ^Y_{proper} ensures that for each variable x_i exactly one value d_j is chosen in β. We define: $\psi^Y_{\text{proper}} = \bigwedge_{1 \leq i \leq n} [\bigvee_{1 \leq j \leq m} y^i_j \wedge \bigwedge_{1 \leq j < j' \leq m} (\neg y^i_j \vee \neg y^i_{j'})]$. Next, the formula $\psi^{Y,Z}_{\text{corr}}$ ensures that β is indeed an extension of α. We define: $\psi^{Y,Z}_{\text{corr}} = \bigwedge_{1 \leq i \leq n} \bigwedge_{1 \leq j \leq m} (z^i_j \rightarrow y^i_j)$. Finally, for each $c \in C$, the formula ψ^Y_c represents whether β satisfies the constraint c. Let $c = ((x_{i_1}, \ldots, x_{i_r}), R) \in C$. We define $\psi^Y_c = \bigvee_{(d_{j_1}, \ldots, d_{j_r}) \in R} \bigwedge_{1 \leq \ell \leq r} y^{i_\ell}_{j_\ell}$. A full proof that $(X, D, C, k) \in$ ROBUST-CSP-SAT if and only if $(\varphi, k) \in \forall^k\exists^*$-WSAT can be found in the technical report. □

In order to prove $\forall^k\exists^*$-hardness, we need the following technical lemma.

Lemma 18. *Let (φ, k) be an instance of $\exists^k\forall^*$-WSAT with $\varphi = \exists X.\forall Y.\psi$. In polynomial time, we can construct an equivalent instance (φ', k) of $\exists^k\forall^*$-WSAT with $\varphi' = \exists X.\forall Y'.\psi'$, such that for any assignment $\alpha : X \rightarrow \{0,1\}$ that has weight $m \neq k$, it holds that $\forall Y'.\psi'[\alpha]$ is true.*

Proof (sketch). We introduce a set Z of additional universally quantified variables, and use these to verify whether k many existentially quantified variables are set to true. We do so by constructing a propositional formula χ containing variables in X and Z that is satisfiable if and only if exactly k many variables in X are set to true. We let $\chi' = (\exists Z.\chi) \rightarrow \psi$. We then know that $\forall Y.\chi'[\alpha]$ is true for all truth assignments $\alpha : X \rightarrow \{0,1\}$ of weight $m \neq k$. Moreover, the formula $\forall Y.\chi'$ is equivalent to the formula $\forall Y \cup Z.(\neg\chi) \vee \psi$. □

Theorem 19. ROBUST-CSP-SAT *is $\forall^k\exists^*$-hard, even when the domain size $|D|$ is restricted to* 2.

Proof. We give an fpt-reduction from $\forall^k\exists^*$-WSAT(3CNF) to ROBUST-CSP-SAT. Let (φ, k) be an instance of $\forall^k\exists^*$-WSAT(3CNF), with $\varphi = \forall X.\exists Y.\psi$, and $\psi = c_1 \wedge \cdots \wedge c_u$. By Lemma 18, we may assume without loss of generality that for any assignment $\alpha : X \rightarrow \{0,1\}$ of weight $m \neq k$, we have that $\exists Y.\psi[\alpha]$ is false. We construct an instance (Z, D, C, k) of ROBUST-CSP-SAT as follows. We define the set Z of variables by $Z = X \cup Y'$, where $Y' = \{ y^i : y \in Y, 1 \leq i \leq 2k+1 \}$, and we let $D = \{0,1\}$. We will define the set C of constraints below, by representing them as a set of clauses whose length is bounded by $f(k)$, for some fixed function f.

The intuition behind the construction of C is the following. We replace each variable $y \in Y$, by $2k+1$ copies y^i of it. Assigning a variable $y \in Y$ to a value $b \in \{0,1\}$ will then correspond to assigning a majority of variables y^i to b, i.e., assigning at least $k+1$ variables y^i to b. In order to encode this transformation in the constraints of C, intuitively, we will replace each occurrence of a variable y by the conjuction $\psi_y = \bigwedge_{1 \leq i_1 < \cdots < i_{k+1} \leq 2k+1} (y^{i_1} \vee \cdots \vee y^{i_{k+1}})$, and replace each occurrence of a literal $\neg y$ by a similar conjunction. We will then multiply the resulting formula out into CNF. Note that whenever a majority of variables y^i is set to $b \in \{0,1\}$, then the formula ψ_y will also evaluate to b.

In the construction of C, we will directly encode the CNF formula that is a result of the transformation described above. For each literal $l = y \in Y$, let l^i denote y^i, and for each literal $l = \neg y$ with $y \in Y$, let l^i denote $\neg y^i$. For each literal l over the variables $X \cup Y$, we define a set $\sigma(l)$ of clauses: $\sigma(l) = (l^{i_1} \vee \cdots \vee l^{i_{k+1}}) : 1 \leq i_1 < \cdots < i_{k+1} \leq 2k+1 \}$ if l is a literal over Y, and $\sigma(l) = l$ if l is a literal over X.

Note that for each literal l, it holds that $|\sigma(l)| \leq g(k) = \binom{2k+1}{k+1}$. Next, for each clause $c_i = l^i_1 \vee l^i_2 \vee l^i_3$ of ψ, we introduce to C a set $\sigma(c_i)$ of clauses: $\sigma(c_i) = \{ d_1 \vee d_2 \vee d_3 : d_1 \in \sigma(l^i_1), d_2 \in \sigma(l^i_2), d_3 \in \sigma(l^i_3) \}$. Note that $|\sigma(c_i)| \leq g(k)^3$. Formally, we let C be the set of constraints corresponding to the set $\bigcup_{1 \leq i \leq u} \sigma(c_i)$ of clauses. Since each such clause is of length at most $3(k+1)$, representing a clause by means of a constraint can be done by specifying $\leq 2^{3(k+1)} - 1$ tuples, i.e., all tuples satisfying the clause. Therefore, the instance (Z, D, C, k) can be constructed in fpt-time. A full proof that $(\varphi, k) \in \forall^k\exists^*$-WSAT(3CNF) if and only if $(Z, D, C, k) \in$ ROBUST-CSP-SAT can be found in the technical report. □

8 Conclusion

We developed a general theoretical framework that supports the classification of parameterized problems on whether they admit an fpt-reduction to SAT or not. Our theory is based on two new hierarchies of complexity classes, the k-$*$ and $*$-k hierarchies. We illustrated the use of this theoretical toolbox by means of two case studies, in which we studied the complexity of the consistency problem for disjunctive answer set programming and a robust version of constraint satisfaction, with respect to various natural parameters. There are many more problems that occur in knowledge representation and reasoning where our theory can be used. In the technical report corresponding to this paper, we use our theory to analyze various additional problems, including the problem of minimizing DNF formulas and the problem of minimizing implicant cores. Additionally, we illustrate the robustness of

our theory by showing a number of complete problems for the newly introduced classes from various domains, as well as providing alternative characterizations of the complexity classes based on first-order model checking and alternating Turing machines. There are many more problems that can be analyzed within our framework.

We focused our attention on the range between the first and the second level of the PH, since many natural problems lie there (Schaefer and Umans, 2002). In general, any fpt-reduction from a problem whose complexity is higher in the PH to a lower level in the PH would be interesting. With this more general aim in mind, it would be helpful to have tools to gather evidence that an fpt-reduction across some complexity border in the PH is not possible. We hope that this paper provides a starting point for further developments.

References

Abramsky, S.; Gottlob, G.; and Kolaitis, P. G. 2013. Robust constraint satisfaction and local hidden variables in quantum mechanics. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013*. AAAI Press/IJCAI.

Ben-Eliyahu, R., and Dechter, R. 1994. Propositional semantics for disjunctive logic programs. *Ann. Math. Artif. Intell.* 12(1):53–87.

Biere, A.; Cimatti, A.; Clarke, E. M.; and Zhu, Y. 1999. Symbolic model checking without BDDs. In Cleaveland, R., ed., *Tools and Algorithms for Construction and Analysis of Systems, 5th International Conference, TACAS '99, Part of the European Joint Conferences on Software, ETAPS'99, Amsterdam, The Netherlands, March 22-28, 1999, Proceedings*, volume 1579 of *Lecture Notes in Computer Science*, 193–207. Springer Verlag.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2009. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Biere, A. 2009. Bounded model checking. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. 457–481.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Communications of the ACM* 54(12):92–103.

Cai, J., and Hemachandra, L. 1986. The boolean hierachy: hardware over NP. In *Proceedings of the 1st Structure in Compexity Theory Conference*, number 223 in Lecture Notes in Computer Science, 105–124. Springer Verlag.

Downey, R. G., and Fellows, M. R. 1995. Fixed-parameter tractability and completeness. II. On completeness for $W[1]$. *Theoretical Computer Science* 141(1-2):109–131.

Downey, R. G., and Fellows, M. R. 1999. *Parameterized Complexity*. Monographs in Computer Science. New York: Springer Verlag.

Eiter, T., and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: propositional case. *Ann. Math. Artif. Intell.* 15(3-4):289–323.

Fages, F. 1994. Consistency of Clark's completion and existence of stable models. *Methods of Logic in Computer Science* 1(1):51–60.

Fichte, J. K., and Szeider, S. 2013. Backdoors to normality for disjunctive logic programs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2013*, 320–327. AAAI Press.

Flum, J., and Grohe, M. 2003. Describing parameterized complexity classes. *Information and Computation* 187(2):291–319.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Berlin: Springer Verlag.

Garey, M. R., and Johnson, D. R. 1979. *Computers and Intractability*. San Francisco: W. H. Freeman and Company, New York.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2007. Conflict-driven answer set solving. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*, 386–392. MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Comput.* 9(3/4):365–386.

Giunchiglia, E.; Lierler, Y.; and Maratea, M. 2006. Answer set programming based on propositional satisfiability. *Journal of Automated Reasoning* 36:345–377.

Gomes, C. P.; Kautz, H.; Sabharwal, A.; and Selman, B. 2008. Satisfiability solvers. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*. Elsevier. 89–134.

Gottlob, G. 2012. On minimal constraint networks. *Artificial Intelligence* 191-192:42–60.

de Haan, R., and Szeider, S. 2014. Fixed-parameter tractable reductions to SAT. Manuscript, submitted, February 2014.

Janhunen, T.; Niemelä, I.; Seipel, D.; Simons, P.; and You, J.-H. 2006. Unfolding partiality and disjunctions in stable model semantics. *ACM Trans. Comput. Log.* 7(1):1–37.

Lee, J., and Lifschitz, V. 2003. Loop formulas for disjunctive logic programs. In Palamidessi, C., ed., *Logic Programming, 19th International Conference, ICLP 2003, Mumbai, India, December 9-13, 2003, Proceedings*, volume 2916 of *Lecture Notes in Computer Science*, 451–465. Springer Verlag.

Lifschitz, V., and Razborov, A. 2006. Why are there so many loop formulas? *ACM Trans. Comput. Log.* 7(2):261–268.

Lin, F., and Zhao, X. 2004. On odd and even cycles in normal logic programs. In Cohn, A. G., ed., *Proceedings of the 19th national conference on Artifical intelligence (AAAI 04)*, 80–85. AAAI Press.

Malik, S., and Zhang, L. 2009. Boolean satisfiability from theoretical hardness to practical success. *Communications of the ACM* 52(8):76–82.

Marek, V. W., and Truszczynski, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*. Springer. 169–181.

Meyer, A. R., and Stockmeyer, L. J. 1972. The equivalence problem for regular expressions with squaring requires exponential space. In *SWAT*, 125–129. IEEE Computer Soc.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford: Oxford University Press.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison-Wesley.

Pfandler, A.; Rümmele, S.; and Szeider, S. 2013. Backdoors to abduction. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013*. AAAI Press/IJCAI.

Sakallah, K. A., and Marques-Silva, J. 2011. Anatomy and empirical evaluation of modern SAT solvers. *Bulletin of the European Association for Theoretical Computer Science* 103:96–121.

Schaefer, M., and Umans, C. 2002. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT News* 33(3):32–49.

Stockmeyer, L. J. 1976. The polynomial-time hierarchy. *Theoretical Computer Science* 3(1):1–22.

Wrathall, C. 1976. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science* 3(1):23–33.

On OBDDs for CNFs of Bounded Treewidth

Igor Razgon
Department of Computer Science and Information Systems
Birkbeck University of London
igor@dcs.bbk.ac.uk

Abstract

In this paper we show that a CNF cannot be compiled into an Ordered Binary Decision Diagram (OBDD) of fixed-parameter size parameterized by the primal graph treewidth of the CNF. Thus we provide a parameterized separation between OBDDs and Sentential Decision Diagrams for which such fixed-parameter compilation is possible. We also show that the best existing parameterized upper bound for OBDDs in fact holds for incidence graph treewidth parameterization.

Introduction

Knowledge compilation is a rewriting approach to propositional knowledge representation. The 'knowledge base' is initially represented as a CNF or even as a Boolean circuit. For these representations many important types of queries are NP-hard to answer. Therefore, the initial representation is compiled into another one for which the minimal requirement is that the clausal entailment query (can the given partial assignment be extended to a complete satisfying assignment?) can be answered in a polynomial time (Darwiche and Marquis 2002). Such transformation can result in exponential blow up of the representation size. A possible way to circumvent this issue is to identify a structural parameter of the input CNF such that the resulting transformation is exponential in this parameter and polynomial in the number of variables. A notable result in this direction is an $O(2^k n)$ upper bound on the size of Decomposable Negation Normal Form (DNNF) (Darwiche 2001), where n is the number of variables of the given CNF and k is the treewidth of its primal graph. Quite recently, the same upper bound has been shown to hold for Sentential Decision Diagrams (SDD) (Darwiche 2011), a subclass of DNNF that can be seen as a generalization of the famous Ordered Binary Decision Diagrams (OBDD) and shares with the OBDD the key nice features (e.g. poly-time equivalence testing). It is known that a CNF of treewidth k can be compiled into an OBDD of size $O(n^k)$ (Ferrara, Pan, and Vardi 2005). A natural question is whether OBDD, similarly to SDD, admits a 'fixed-parameter' upper bound of form $f(k)n^c$ for some constant c.

In this paper we provide a negative answer to this question. In particular, we demonstrate an infinite class of CNFs of the primal graph treewidth at most k for which the OBDD size is at least $f(k)n^{k/4}$ where f is a function exponentially small in k. In other words, we show that the OBDD size of these CNFs is $\Omega(n^{k/4})$ for every fixed k. This result provides a separation from SDD and essentially matches the upper bound of (Ferrara, Pan, and Vardi 2005). In fact, this result shows impossibility of not only a fixed-parameter upper bound, but also of a sublinear dependence on k in the base of the exponent or even of an exponent k/C for some large constant C.

Our second result is 'strengthening' of the upper bound $O(n^k)$ of (Ferrara, Pan, and Vardi 2005) by showing that it holds if k is the treewidth of the *incidence* graph of the given CNF thus extending the upper bound to the case of sparse CNFs with large clauses.

In order to obtain the lower bound, we introduce a notion of *matching width* of a graph and prove that if a CNF F of the considered class has matching width r of the primal graph then for any ordering of the variables of F there is a prefix S such that the number of distinct functions that can be obtained from F by assigning the variables of S is at least 2^r. This will immediately imply that any OBDD realizing F will have at least 2^r nodes. Finally we will prove that the matching width of the considered CNFs is $\Omega(logn * k)$. Substituting this lower bound instead r will get the desired lower bound for the OBDD size.

Similarly to the case of primal graph, the upper bound is obtained by showing that if *pathwidth* of the incidence graph of the given CNF is at most p then this CNF can be compiled into an OBDD of size $O(2^p n)$. Then the $O(n^k)$ upper bound is obtained using a well known relation $p = O(k * logn)$ between the treewidth and the pathwidth of the given graph. The approach to obtain the $O(2^p n)$ bound is similar to (Ferrara, Pan, and Vardi 2005): variables are ordered 'along' the path decomposition and it is observed that the for each prefix the number of functions caused by assigning the 'previous' variables is $O(2^p)$. The technical difference is that in our case the bags of the path decomposition include clauses and this circumstance must be taken into account.

The proposed results contribute to a large body of existing results concerning the space complexity of OBDDs. To begin with, there are many results concerning the complexity of OBDDs for *particular* classes of Boolean functions, see e.g. the book (Wegener 2000) and the survey (Wegener

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

2004). The space complexity of OBDD remains polynomial if parameterized by the treewidth of a *circuit* representing the given function (Jha and Suciu 2012), however the dependence on the treewidth becomes double exponential. A fixed-parameter upper bound can be achieved if tree of OBDDs is used instead of a single OBDD (McMillan 1994; Subbarayan, Bordeaux, and Hamadi 2007). In the complexity theory the OBDD is classified as the *oblivious read-once branching program*, see the book (Jukna 2012) for the results concerning the complexity of branching programs on particular classes of formulas

The proposed lower bound also contributes to the understanding of relationship between OBDD and SDD. Other results in this direction are (Xue, Choi, and Darwiche 2012) showing an exponential separation between SDD and OBDD based on the same order of variables (the order of variables for SDD is defined as the order of visiting the corresponding nodes of the underlying *vtree* by a left-right tree traversal algorithm) and (Choi and Darwiche 2013) empirically showing that conceptually similar heuristics produce SDDs orders of magnitude smaller than OBDDs.

The rest of the paper is structured as follows. The next section introduces the necessary background. The section after that proves the lower bound, the proofs of auxiliary statements are provided in the two following sections. Then follows the section presenting the upper bound for the parameterization by the treewidth of the incidence graph. The last section outlines relevant directions of further research.

Preliminaries

The structure of this section is the following. First, we introduce notational conventions. Then we define the OBDD and specify the approach we use to prove the lower bound. Next, we introduce terminology related to CNFs. Finally, we define the notion of treewidth.

In this paper by a *set of literals* we mean one that does not contain an occurrence of a variable and its negation. For a set S of literals we denote by $Var(S)$ the set of variables whose literals occur in S. If F is a Boolean function or its representation by a CNF or OBDD, we denote by $Var(F)$ the set of variables of F. A truth assignment to $Var(F)$ on which F is true is called a *satisfying assignment* of F. A set S of literals represents the truth assignment to $Var(S)$ where variables occurring positively in S (i.e. whose literals in S are positive) are assigned with $true$ and the variables occurring negatively are assigned with $false$. We denote by F_S a function whose set of satisfying assignments consists of S' such that $S \cup S'$ is a satisfying assignment of F. We call F_S a *subfunction* of F. In other words, a Boolean function F' is a subfunction of a Boolean function F is F' can be obtained from F by giving a truth assignment to a subset of variables of F.

An OBDD Z representing a Boolean function F is a directed acyclic graph (DAG) with one root and two leaves labelled by $true$ and $false$. The internal nodes are labelled with variables of F. There is a fixed permutation SV of $Var(F)$ (that is, elements of $Var(F)$ are linearly ordered according to SV) so that the vertices along any path from the root to a leaf are labelled with variables according to this order. Each internal vertex is associated with 2 leaving edges labelled with $true$ and $false$. Each path P from the root of Z is called a *computational path* and is associated with truth assignment to the variables labelling all the vertices but the last one. In particular, each variable is assigned with the value labelling the edge of the path that leaves the corresponding vertex. We denote by $A(P)$ the assignment associated with the computational path P. The set of all $A(P)$ where P is a computational path ending at the $true$ leaf is precisely the set of satisfying assignments of F.

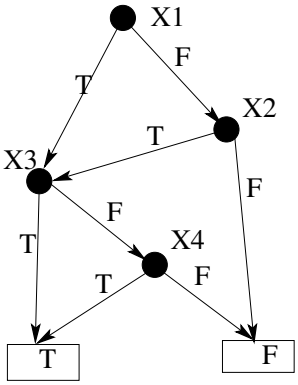


Figure 1: An OBDD for $(x_1 \vee x_2) \wedge (x_3 \vee x_4)$ under permutation (x_1, x_2, x_3, x_4)

Figure 1 shows an OBDD for the function $(x_1 \vee x_2) \wedge (x_3 \vee x_4)$ under the permutation (x_1, x_2, x_3, x_4). Consider the path $P = (x_1, x_2, x_3)$. Then $A(P) = \{\neg x_1, x_2\}$.

In order to obtain the lower bound on the OBDD size we use a standard approach of counting subfunctions. See (Wegener 2000) for examples of application of this approach. This approach is based on the following statement.

Proposition 1 *Let F be a Boolean function on a set V of variables and let SV be a permutation of V. Partition SV into a prefix SV_1 and a suffix SV_2 and suppose that the number of distinct subfunctions of F obtained by giving truth assignments to all the variables of SV_1 is at least x. Then an* OBDD *of F with the underlying order SV contains at least x nodes.*

The standard way to utilize Proposition 1 is to show that for *any* permutation SV of V there is a partition of SV into a prefix SV_1 and a suffix SV_2 such that the instantiation of variables of SV_1 results in at least x different subfunctions. Then Proposition 1 immediately implies that x is a lower bound on the size of OBDD for *any* underlying order.

Given a CNF F, its *primal graph* has the set of vertices corresponding to the variables of F. Two vertices are adjacent if and only if there is a clause of F where the corresponding variables both occur. In the *incidence graph* of F the vertices are partitioned into those corresponding to the variables of F and those corresponding to its clauses. A variable vertex is adjacent to a clause vertex if and only if the corresponding variable occurs in the corresponding clause.

Given a graph G, its *tree decomposition* is a pair $(T, \mathbf{B})$ where T is a tree and $\mathbf{B}$ is a set of bags $B(t)$ corresponding

to the vertices t of T. Each $B(t)$ is a subset of $V(G)$ and the bags obey the rules of *union* (that is, $\bigcup_{t \in V(T)} B(t) = V(G)$), *containment* (that is, for each $\{u, v\} \in E(G)$ there is $t \in V(t)$ such that $\{u, v\} \subseteq B(t)$), and *connectedness* (that is for each $u \in V(G)$, the set of all t such that $u \in B(t)$ induces a subtree of T). The *width* of $(T, \mathbf{B})$ is the size of the largest bag minus one. The treewidth of G is the smallest width of a tree decomposition of G. If T is a path then we use the respective notions of *path decomposition* and *pathwidth*.

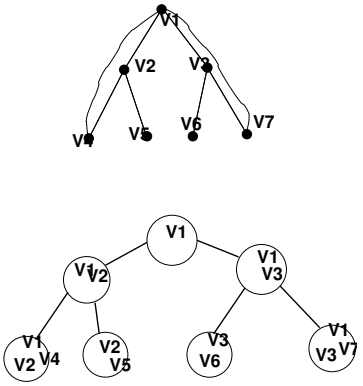


Figure 2: A graph and its tree decomposition

Figure 2 shows a graph and its tree decomposition. The width of this tree decomposition is 2 since the size of the largest bag is 3.

The lower bound

In this section, given two integers r and k we define a class of CNFs, roughly speaking, based on complete binary trees of height r where each node is associated with a clique of size k. Then we prove that the treewidth of the primal graphs of CNFs of this class is linearly bounded by k. Further on, we state the main technical theorem (proven in the next section) that claims that the smallest OBDD size for CNFs of this class exponentially depends on rk. Finally, we re-interpret this lower bound in terms of the number of variables and the treewidth to get the lower bound announced in the Introduction.

Let G be a graph. A *graph based* CNF denoted by $CNF(G)$ is defined as follows. The set of variables consists of variables X_u for each $u \in V(G)$ and variables $X_{u,v} = X_{v,u}$ for each $\{u, v\} \in E(G)$. The set of clauses consists of clauses $C_{u,v} = C_{v,u} = (X_u \vee X_{u,v} \vee X_v)$ for each $\{u, v\} \in E(G)$. In other words, the variables of $CNF(G)$ correspond to the vertices and edges of G. The clauses correspond to the edges of G.

Denote by T_r a complete binary tree of height r. Let $CT_{r,k}$ be the graph obtained from T_r by associating each vertex with a clique of size k and, for each edge $\{u, v\}$ of G, making all the vertices of the cliques associated with u and v mutually adjacent. Denote $CNF(CT_{r,k})$ by $F_{r,k}$.

Figure 3 shows T_2 and $CT_{2,3}$. To avoid shading the picture of $CT_{2,3}$ with many edges, the cliques corresponding to

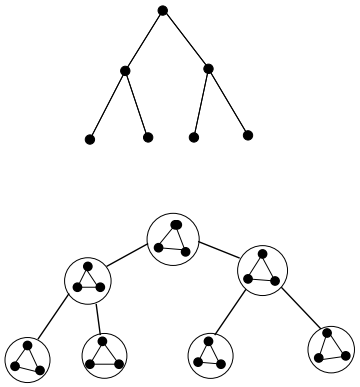


Figure 3: T_2 and $CT_{2,3}$

the vertices of T_2 are marked by circles and the bold edges between the circles mean that that there are edges between all pairs of vertices of the corresponding cliques.

Lemma 1 *The treewidth of the primal graph of $F_{r,k}$ is at least $k - 1$ at most $2k - 1$. In fact, for $r \geq 1$, this treewidth is exactly $2k - 1$.*

Proof. The primal graph of $F_{r,k}$ can be obtained from $CT_{r,k}$ by adding one vertex v_e for each edge e of $CT_{r,k}$ and making this vertex adjacent to the ends of e.

The lower bound follows from existence of a clique of size k in $CT_{r,k}$. Indeed, in any tree decomposition of $CT_{r,k}$, there is a bag containing all the vertices of such a clique (Bodlaender and Möhring 1993). Consequently, the width of any tree decomposition is at least $k - 1$. In fact if $r \geq 1$ then $CT_{r,k}$ has a clique of size $2k$ created by cliques of two adjacent nodes. Hence, due to the same argumentation, the treewidth of $CT_{r,k}$ is at least $2k - 1$ for $r \geq 1$.

For the upper bound, consider the following tree decomposition $(T, \mathbf{B})$ of $CT_{r,k}$. T is just T_r. We look upon T_r as a rooted tree, the centre of T_r being the root. The bag $B(u)$ of each node u contains the clique of $CT_{r,k}$ corresponding to u. In addition, if u is not the root vertex then $B(u)$ also contains the clique corresponding to the parent of u. Observe that $(T, \mathbf{B})$ satisfies the connectivity property. Indeed, each vertex appears in the bag corresponding to its 'own' clique and the cliques of its children. Clearly, the set of nodes corresponding to the bags induce a connected subgraph. The rest of the tree decomposition properties can be verified straightforwardly. We conclude that $(T, \mathbf{B})$ is indeed a tree decomposition of $CT_{r,k}$.

In order to 'upgrade' $(T, \mathbf{B})$, add $\binom{k}{2}$ new adjacent vertices to each vertex of T. These vertices will correspond to the edges of cliques associated with the respective nodes of T_r. In addition, add k^2 new adjacent vertices to each non-root vertex of T. These vertices will correspond to the edges between the clique associated with the corresponding node of T_r and the clique of its parent. The bag of each new vertex will contain v_e, corresponding to the edge e associated with this bag, plus the ends of e. A direct inspection shows that this is indeed a tree decomposition of the primal graph of $F_{r,k}$ and that the size of each bag is at most $2k$.

Notice that for $r \geq 1$ the lower and upper bounds coincide, thus allowing to state the treewidth precisely. ■

The following is the main technical result whose proof is given in the next section.

Theorem 1 *The size of* OBDD *computing $F_{r,k}$ is at least $2^{rk/2}$.*

Let us reformulate the statement of Theorem 1 in terms of the number of variables of $F_{r,k}$ and the treewidth of its primal graph, having in mind the bounds on the treewidth as in Lemma 1.

First of all, taking into account that $k \geq p/2$, where p is the treewidth of the primal graph of $F_{r,k}$, the lower bound can be seen as $2^{rp/4}$. Next, let m be the number of variables of $F_{r,k}$. Then, it is not hard to observe that $2^r \geq \frac{m}{2^{\binom{k}{2}}} \geq \frac{m}{2^{\binom{(p+1)}{2}}}$. Replacing 2^r this way in $2^{rp/4}$, we obtain $(\frac{m}{2^{\binom{(p+1)}{2}}})^{p/4} = \binom{2(p+1)}{2}^{-p/4} * m^{p/4}$ as a lower bound on the OBDD size for $F_{r,k}$. Clearly, if we consider p as a constant, this lower bound can be seen as $\Omega(m^{p/4})$.

Now we are ready to state the main result.

Corollary 1 *There is a function f such that for each $p \geq 1$ there is an infinite sequence of* CNFs *$F_1, F_2 \ldots$, of treewidth at most p of their primal graphs such that for each F_i the size of* OBDD *computing it is at least $f(p) * m^{p/4}$ where m is the number of variables of F_i. Put it differently, for each fixed p, there is a class of* CNFs *of treewidth at most p of the primal graph for which the* OBDD *size is $\Omega(m^{p/4})$.*

Proof. For an odd p, consider the CNFs $F_{r,(p+1)/2}$ for all $r \geq 1$ and for an even p, consider the CNFs $F_{r,p/2}$ for all $r \geq 1$. Observe that for an even p the primal graph treewidth of $F_{r,p}$ is $p-1$ and that the above argumentation still applies. Indeed, since $k = p/2$, it is legitimate to represent the lower bound as $2^{rp/4}$. Further on, in the inequality that follows, the occurrence of $p+1$ in the denominator (as an upper bound of the actual treewidth) even strengthens this inequality. ■

Proof of Theorem 1

The plan of the proof is the following. We introduce the notion of matching width of a graph. Then we provide two statements regarding this notion. The first statement (Lemma 2) claims a linear in rk lower bound for the matching width of graphs $CT_{r,k}$ underlying the considered class $F_{r,k}$ (the proof of the lemma is provided in the next section). The second statement (Lemma 3) claims that if a graph G has a matching width t then any permutation of the variables of $CNF(G)$ can be partitioned into a suffix and a prefix so that there are at least 2^t subfunctions of $CNF(G)$ resulting from instantiation of variables of the prefix. The proof of Lemma 3 constitutes the essential part of this section. Finally, we provide a proof of Theorem 1. In this proof we notice that according to the approach outlined in the Preliminaries section, Lemma 3 together with Proposition 1 implies that the size of an OBDD of $CNF(G)$ is at least 2^t. Taking $CT_{r,k}$ as G and substituting the lower bound claimed by Lemma 2, we obtain the desired lower bound for $F_{r,k} = CNF(CT_{r,k})$.

The *matching width* is defined as follows. Let SV be a *permutation* of the set $V = V(G)$ of vertices of a graph G. Let S_1 be a *prefix* of SV (i.e. all vertices of $SV \setminus S_1$ are ordered after S_1). Let us call the matching width of S_1, the largest *matching* (that is, a set of edges not having common ends) consisting of the edges between S_1 and $V \setminus S_1$ (we take the liberty to use sequences as sets, the correct use will be always clear from the context). Further on, the matching width of SV is the largest matching width of a prefix of SV. Finally the matching width of G, denoted by $mw(G)$, is the smallest matching width of a permutation of $V(G)$.

Example 1 *Consider a path of 10 vertices $v_1, \ldots, v_{10}$ so that v_i is adjacent to v_{i+1} for $1 \leq i < 10$. The matching width of permutation $(v_1, \ldots, v_{10})$ is 1 since between any suffix and prefix there is only one edge. However, the matching width of the permutation $(v_1, v_3, v_5, v_7, v_9, v_2, v_4, v_6, v_8, v_{10})$ is 5 as witnessed by the partition $\{v_1, v_3, v_5, v_7, v_9\}$ and $\{v_2, v_4, v_6, v_8, v_{10}\}$. Since the matching width of a graph is determined by the permutation having the smallest matching width, and, since the graph has edges, there cannot be a permutation of matching width 0, we conclude that the matching width of this graph is 1.*

Lemma 2 *For any r, the matching width of $CT_{r,k}$ is at least $rk/2$.*

The proof of Lemma 2 is provided is the next section.

Remark. The above definition of matching width is a special case of a more general notion of *maximum matching width* as defined in (Vatshelle 2012). In particular our notion of matching width can be seen as a variant of maximum matching width of (Vatshelle 2012) where the tree T involved in the definition is a caterpillar.

We are now showing that for CNFs of form $CNF(G)$, a large matching width of G is sufficient for establishing a strong lower bound.

Lemma 3 *Let G be a graph having matching width t. Denote $CNF(G)$ by F. Then any permutation SF of $Var(F)$ has a prefix SF_1 such that there are at least 2^t different functions of form F_{S_1} such that S_1 is a truth assignment to the variables of SF_1.*

Proof. Let us partition $Var(F)$ into sets VV of variables corresponding to the vertices of G and EV of variables corresponding to the edges of G. Let SV be the permutation of VV ordered in the way as they are ordered in SF. Let SV_1 be a prefix of SV *witnessing* the matching width t of SV. (Recall that the matching width of SV is at least the matching width of G.) The word 'witnessing' in this context means that there is a matching $M = \{\{u_1, v_1\}, \ldots, \{u_t, v_t\}\}$ between SV_1 and $V(G) \setminus SV_1$. Let SF_1 be the prefix of SF ending with the last element of SV_1. Thus the variables $X_{u_1}, \ldots X_{u_t}$ corresponding to $u_1, \ldots, u_t$ belong to SF_1 while the variables $X_{v_1}, \ldots, X_{v_t}$ corresponding to $v_1, \ldots, v_t$ do not. We denote the set of clauses $(X_{u_i} \vee X_{u_i,v_i} \vee X_{v_i})$ by TCL.

In the rest of the proof we essentially show that 2^t different assignments to variables $X_{u_1}, \ldots X_{u_t}$ produce 2^t different subfunctions of F thus confirming the lemma. Roughly speaking, this is done by showing that by a careful fixing the assignments to *the rest* of the variables of SF_1 we can achieve the effect that an assignment to X_{u_i} does not 'influence' an assignment to X_{v_j} for $i \neq j$. As a result no two assignments to $X_{u_1}, \ldots, X_{u_t}$ can have the same effect on $X_{v_1}, \ldots, X_{v_t}$ and this guarantees that desired large set of subfunctions.

We start from defining a set of 2^t assignments for which we then claim that any two assignments induce two distinct subfunctions of F. In particular, let $\mathbf{S}$ be the set of all assignments to the variables of SF_1 that assign the variables X_{u_i,v_i} (of course, those of them that belong to SF_1) with $false$ and the rest of variables except $X_{u_1}, \ldots, X_{u_t}$ with $true$. It is easy to see by construction that $\mathbf{S}$ is in a natural one-to-one correspondence with the set of possible assignments to $X_{u_1}, \ldots, X_{u_t}$. In particular, each $S \in \mathbf{S}$ corresponds to the assignment A to $X_{u_1}, \ldots, X_{u_t}$ contained in it. Indeed, the assignments of the rest of the variables are fixed in $\mathbf{S}$ by construction. It follows that the size of $\mathbf{S}$ is 2^t.

We are going to show that for any distinct $S_1, S_2 \in \mathbf{S}$, $F_{S_1} \neq F_{S_2}$, confirming the lemma. Due to the correspondence established above, we can specify u_i such that S_1 and S_2 assign X_{u_i} with distinct values. Assume w.l.o.g. that X_{u_i} is assigned with $true$ by S_1 and with $false$ by S_2. Observe that F does not have a satisfying assignment including S_2 and assigning both X_{u_i,v_i} and X_{v_i} with $false$. Indeed, as a result, the clause $(X_{u_i} \vee X_{u_i,v_i} \vee X_{v_i})$ is falsified. We are going to show that both X_{u_i,v_i} and X_{v_i} can be assigned with $false$ in a satisfying assignment of F including S_1. Indeed, assign all the variables of $Var(F) \setminus (Var(S_1) \cup \{X_{u_i,v_i}, X_{v_i}\})$ with $true$ and see that the resulting assignment together with S_1 satisfies all the clauses of F. Indeed, if a clause $(X_u \vee X_{u,v} \vee X_v)$ does not belong to TCL then $X_{u,v}$ is assigned with $true$ (by construction, the only 'edge' variables assigned by $false$ are X_{u_i,v_i}, that is those that occur in the clauses of TCL) . Furthermore, for any clause $(X_{u_j} \vee X_{u_j,v_j} \vee X_{v_j})$ of TCL such that $i \neq j$, X_{v_j} is assigned with $true$. Finally X_{u_i} is assigned with $true$ by S_1. It follows that indeed all the clauses of F are satisfied.

Assume that $X_{u_i,v_i} \notin Var(S_1)$. Then, by the reasoning as above, F_{S_1} has a satisfying assignment including $\{\neg X_{u_i,v_i}, \neg X_{v_i}\}$ while F_{S_2} does not implying that $F_{S_1} \neq F_{S_2}$. Otherwise, if $X_{u_i,v_i} \in Var(S_1)$, it is assigned with $false$ in both S_1 and S_2, by construction. It follows that F_{S_1} has a satisfying assignment including $\neg X_{v_i}$ while F_{S_2} does not. It follows again that $F_{S_1} \neq F_{S_2}$. ■

Remark. Notice the role of variables $X_{u,v}$ in the proof of Lemma 3. They allow the values of X_{u_i} to *not influence* the values of X_{v_j} for $i \neq j$ and thus keep the number of different subfunctions up to the desired bound. Due to the same reason, it is important that the edges $\{u_1, v_1\}, \ldots, \{u_r, v_r\}$ constitute a *matching*, i.e. have disjoint ends.

Proof of Theorem 1 Lemma 3 combined with Proposition 1 says that if G has matching width at least t then for any permutation of $Var(CNF(G))$ the corresponding OBDD has at least 2^t nodes. In other words, 2^t is a lower bound on the OBDD size for $CNF(G)$. Taking $G = CT_{r,k}$ and hence $CNF(G) = F_{r,k}$ and substituting $rk/2$ for t according to Lemma 2, we obtain a lower bound of $2^{rk/2}$ on the OBDD size of $F_{r,k}$, as required. ■

Proof of Lemma 2

This section is organized as follows. First, we introduce the notion of *induced permutation*. Then we provide proof of Lemma 2 for $k = 1$. After that, we outline how to upgrade this special case to a complete proof. Finally, we provide the complete proof. Note that the proof of the special case and the following outline are *technically* redundant. However, the reader may find them useful as they provide a *sketch* reflecting the proof idea.

The notion of *induced permutation* is defined as follows. Let P_1 be a permutation of elements of a set S_1 and let $S_2 \subseteq S_1$. Then P_1 induces a permutation P_2 of S_2 where the elements of S_2 are ordered exactly as they are ordered in P_1. For example, let $S_1 = \{1, \ldots, 10\}$ and let S_2 be the subset of even numbers of S_1. Let $P_1 = (1, 8, 2, 9, 5, 6, 7, 3, 4, 10)$. Then $P_2 = (8, 2, 6, 4, 10)$.

Proof of the special case of Lemma 2 for $k = 1$ We are going to prove that for an odd r, the matching width of T_r is at least $(r + 1)/2$. For an even r we can simply take a subgraph of T_r isomorphic to T_{r-1} (it is not hard to see that the matching width of a graph is not less than the matching width of its subgraph).

The proof goes by induction on r. For $r = 1$, this is clear, so consider the case $r > 1$. Imagine T_r rooted in the natural way, the root being its centre. Then T_r has 4 grandchildren, the subtree rooted by each of them being T_{r-2}. Denote these grandchildren by $T^1, \ldots, T^4$. Let PV be any permutation of the vertices of T_r. This permutation induces respective permutations $PV_1, \ldots, PV_4$ of vertices of $T^1, \ldots, T^4$ being ordered exactly as in PV. By the induction assumption, we know that each of $PV_1, \ldots, PV_4$ can be partitioned into a prefix and a suffix so that the edges between the prefix and the suffix induce graph having matching of size at least $(r-1)/2$. Each of these prefixes naturally corresponds to the prefix of PV ending with the same vertex. Since $PV_1, \ldots, PV_4$ are pairwise disjoint, this correspondence supplies 4 *distinct* prefixes $P_1^*, \ldots, PV_4^*$ of PV. Moreover, for each PV_i^* we know that the graph G_i^* induced by the edges between the vertices of PV_i^* and the rest of the vertices has a matching of size $(r-1)/2$ consisting *only* of the edges of T^i. In order to 'upgrade' this matching by 1 and hence to reach the required size of $(r + 1)/2$, all we need to show is that in an least one G_i^* there is an edge both ends are not vertices of T^i and hence this edge can be safely added to the matching.

At this point we make a notational assumption that does not lead to loss of generality and is convenient for the further exposition. By construction, $PV_1^*, \ldots, PV_4^*$ are linearly ordered by containment and we assume w.l.o.g. that the ordering is by the increasing order of the subscript, that is $PV_1^* \subset PV_2^* \subset PV_3^* \subset PV_4^*$. We claim that the upgrade to the matching as specified above is possible for PV_2^*.

Indeed, observe that $T_r \setminus T^2$ is a connected graph. Thus

all we need to show is that at least one vertex of $T_r \setminus T^2$ gets into PV_2^* and at least one vertex of $T_r \setminus T^2$ gets outside PV_2^*, that is in $V(T_r) \setminus PV_2^*$.

For the former, recall that $PV_1^* \subset PV_2^*$ and that by construction, PV_1^* contains $(r-1)/2$ vertices of T^1 being a subgraph of $T_r \setminus T^2$. Thus we conclude that PV_2^* contains vertices of $T_r \setminus T^2$ For the latter, observe that since $PV_2^* \subset PV_3^*, V(T_r) \setminus PV_3^* \subset V(T_r) \setminus PV_2^*$. Furthermore, by construction, $V(T_r) \setminus PV_3^*$ contains $(r-1)/2$ vertices of T^3 being a subgraph of $T_r \setminus T^2$. Thus we conclude that $V(T_r) \setminus PV_2^*$ contains vertices of $T_r \setminus T^2$ as well, thus finishing the proof. ■

A proof for the general case of Lemma 2 proceeds by induction on r similarly to the special case above. Of course we need to keep in mind that instead of nodes of T_r we have cliques of size k. The consequence of this substitution is that at the inductive step of moving from T_{r-2} to T_r we can increase the matching width by k rather than by 1 as above. The auxiliary Lemma 4 allows us to demonstrate the possibility of this upgrade essentially in the same way as we did for $k = 1$: we just show that the considered prefix and suffix of the given permutation both contain at least k vertices outside the grandchild serving the part of the matching guaranteed by the induction assumption.

Lemma 4 *Let T be a tree with at least 2 nodes and let k be a positive integer. Let CT be a graph obtained from T by associating with each vertex of T a clique of an arbitrary size $k' \geq k$ and making the vertices of cliques associated with adjacent vertices of T mutually adjacent. Let W, B standing for 'white' and 'black' be a partition of $V(CT)$ such that $|W| \geq k$ and $|B| \geq k$. Then CT has a matching of size k formed by edges with one white and one black end.*

Proof. The proof is by induction on the number of nodes of T. It is clearly true when there are 2 nodes. Assume that the tree has $n > 2$ nodes and let u be a leaf of T and v be its only neighbour.

Let $k' \geq k$ be the size of the clique VU associated with u in CT. Assume w.l.o.g. that $|W \cap VU| \leq |B \cap VU|$. Denote $|W \cap VU|$ by k_1. Clearly, the k_1 vertices of $W \cap VU$ can be matched with the vertices of $B \cap VU$. If $k_1 \geq k$, we are done. Next, if $|B \setminus VU| \geq k - k_1$, then the lemma follows by induction assumption applied on $T \setminus u$.

Consider the remaining possibility where $|B \setminus VU| = k - k_1 - t$ for some $t > 0$. Observe that $t \leq k' - 2k_1$. Indeed, the total number of vertices of B is $k' - k_1 + k - k_1 - t$ so, $t > k' - 2k_1$ will imply $|B| < k$, a contradiction.

Let VV be the clique associated with the neighbour v of u. It follows from our assumption that $|W \cap VV| \geq k_1 + t$ because at most $k - k_1 - t$ vertices of VV can be black. Match k_1 vertices of $W \cap VU$ with vertices of $B \cap VU$ (this is possible due to our assumption that $|W \cap VU| \leq |B \cap VU|$). Match t unmatched vertices of $B \cap VU$ (there are $k' - 2k_1$ unmatched vertices of $B \cap VU$ and we have just shown that $t \leq k' - 2k_1$) with t vertices of $W \cap VV$. We are in the situation where in $G \setminus u$ there are at least $k - k_1 - t$ vertices of W, at least $k - k_1 - t$ vertices of B and the size of each associated clique is clearly at least $k - k_1 - t$. Hence, the lemma follows by the induction assumption. ■

Proof of Lemma 2. We prove that for an odd r, the matching width of $CT_{r,k}$ is at least $(r+1)k/2$. For an even r, it will be enough to consider a subgraph of $CT_{r,k}$ being isomorphic to $CT_{r-1,k}$. The proof is by induction on r. Assume first that $r = 1$. Then the lemma holds according to Lemma 4.

For $r > 1$, let us view T_r as a rooted tree with its centre rt being the root. Let $T^1, \ldots, T^4$ be the 4 subtrees of T_r rooted by the 'grandchildren' of rt. Let $K_1, \ldots, K_4$ be the subgraphs of $CT_{r,k}$ 'corresponding' to $T^1, \ldots, T^4$. That is, each K_i is a subgraph of $CT_{r,k}$ induced by (the vertices of) cliques associated with the vertices of T^i. It is not hard to see that each T^i is isomorphic to T_{r-2} and each K_i is isomorphic to $CT_{r-2,k}$ and that $K_1, \ldots, K_4$ are pairwise disjoint.

Let PV be an arbitrary permutation of $V(CT_{r,k})$. Let $PV_1, \ldots, PV_4$ be the respective permutations of $V(K_1), \ldots, V(K_4)$ induced by PV. By the induction assumption for each PV_i there is a prefix PV_i' such that the edges of K_i with one end in PV_i' and the other end in $PV_i \setminus PV_i'$ induce a graph having matching of size at least $(r-1)k/2$. Let $u_1, \ldots, u_4$ be the last vertices of $PV_1', \ldots PV_4'$, respectively. Assume w.l.o.g. that these vertices occur in PV in exactly this order. Let PV' be the prefix of PV with final vertex u_2. We are going to show that the subgraph of $CT_{r,k}$ induced by the edges between PV' and $PV \setminus PV'$ has matching of size at least $(r+1)k/2$. In fact, as specified above, we already have matching of size $(r-1)k/2$ if we confine ourself to the edges between $PV' \cap PV_2$ and $(PV \setminus PV') \cap PV_2$. Thus, it only remains to show the existence of matching of size k in the subgraph of $CT_{r,k}$ induced by the edges between $PV_1^* = PV' \setminus PV_2$ and $PV_2^* = (PV \setminus PV') \setminus PV_2$. Observe that PV_1^*, PV_2^* is a partition of vertices of $CT_{r,k} \setminus K_2$. Therefore, it is sufficient to show that $|PV_1^*| \geq k$ and $|PV_2^*| \geq k$ and then the existence of the desired matching of size k will follow from Lemma 4.

Due to our assumption that u_1 precedes u_2 in PV, it follows that PV_1' is contained in PV'. Moreover, since K_1 and K_2 are disjoint, PV_1' is disjoint with PV_2 and hence $PV_1' \subseteq PV_1^*$. Recall that by the induction assumption, the vertices of PV_1' serve as ends of a matching of size $(r-1)k/2$ with no two vertices sharing the same edge of the matching. That is $|PV_1'| \geq (r-1)k/2$. Since $r > 1$ by assumption, we conclude that $|PV_1'| \geq k$ and hence $|PV_1^*| \geq k$.

The proof that $|PV_2^*| \geq k$ is symmetrical. By our assumption, u_2 precedes u_3 is PV and hence $PV_3 \setminus PV_3'$ is contained in $PV \setminus PV'$ and due to the disjointness of K_2 and K_3, $PV_3 \setminus PV_3'$ is in fact contained in PV_2^*. That $|PV_3 \setminus PV_3'| \geq k$ is derived analogously to the proof that $|PV_1'| \geq k$. ■

OBDDs parameterized by the treewidth of the incidence graph

Recall that the incidence graph of the given CNF F has the set of vertices corresponding to its variables and clauses and a variable vertex is adjacent to a clause vertex if and only

if the corresponding variable occurs in the corresponding clause. The upper bound of (Ferrara, Pan, and Vardi 2005) does not straightforwardly apply to the case of incidence graphs because there are classes of CNFs having constant treewidth of the incidence graph and unbounded treewidth of the primal graph. Indeed, consider, for example a CNF with one large clause. Nevertheless, we show in this section that the $O(n^k)$ upper bound on the size of OBDD holds if k is the treewidth of the incidence graph of the considered CNF.

As in (Ferrara, Pan, and Vardi 2005), we show that if p is the pathwidth of the *incidence graph* G of the given CNF F then the function of F can be realized by an OBDD of size $O(2^p n)$ implying (through the $k = O(p * logn)$) the $O(n^k)$ upper bound where k is the treewidth of G. The resulting OBDD is seen as a DAG whose nodes are partitioned into layers, each layer consisting of nodes labelled by the same variable. The main technical lemma shows that under the right permutation of variables the nodes of each layer correspond to $O(2^p)$ subfunctions of F. Consequently, $O(2^p)$ nodes per layer are sufficient, which in turn, immediately implies the desired upper bound.

Let us start from fixing the notation. Let F be a CNF and G be its incidence graph, whose nodes are $X_1, \ldots, X_n$ (corresponding to the variables of F) and $C_1, \ldots, C_m$ (corresponding to the clauses of F) and X_i and adjacent to C_j if and only if X_i occurs in C_j (for the sake of brevity, we identify the vertices of G with the corresponding variables and clauses). Let $(P, \mathbf{B})$ be a path decomposition of G. Fix an end vertex of P and enumerate the vertices of P along the path starting from this fixed vertex. Let $v_1, \ldots, v_r$ be the enumeration. For each X_i, let $f(X_i)$ be the smallest j such that $X_i \in B(v_j)$. We call a linear ordering SV of $X_1, \ldots, X_n$ such $X_i < X_j$ whenever $f(X_i) < f(X_j)$ an ordering *respecting* f.

Now we are ready to prove the main technical lemma.

Lemma 5 *Let SV be an ordering respecting f. Let SV_1 be a prefix of SV. Then the number of distinct F_S such that S is an assignment to SV_1 is at most $1 + 2 * 2^p$ where p is the width of $(P, \mathbf{B})$.*

Proof. Let X be the last variable of SV_1. Denote $f(X)$ by q. We assume w.l.o.g. that all the clauses of F are pairwise distinct and hence identify a CNF with its set of clauses. Partition F into three sets of clauses: FP, consisting of those that appear in some $B(v_j)$ for $j < q$ and do not appear in $B(v_q)$; FC, consisting of those that appear in $B(V_q)$ and FF consisting of those that appear in $B(v_j)$ for some $j > q$ and do not appear in $B(V_q)$. Observe that this is indeed a partition of clauses. Indeed, otherwise $FP \cap FF \neq \emptyset$ as all other possibilities contradict the definition of the sets FP, FC, FF. Then due to the connectedness property of $(P, \mathbf{B})$, either $FP \cap B(v_q) \neq \emptyset$ or $FF \cap B(v_q) \neq \emptyset$. However, both these possibilities contradict the definition of FP and FF. We conclude that FP, FC, FF indeed partition the clauses of F.

Denote by $\mathbf{FS}$ the set of all functions F_S such that S is an assignment to SV_1. Denote by $\mathbf{FPS}$, $\mathbf{FCS}$, $\mathbf{FFS}$ the analogous sets regarding FP, FC, and FF, respectively.

Let us compute the sizes of the latter 3 sets. Let C be a clause of FP. By definition $Var(C)$ is a subset of variables appearing in the bags $B(v_j)$ for $j < q$. By definition, these variables are ordered *before* X. It follows that $Var(C) \subset Var(SV_1)$ and hence any assignment to SV_1 either satisfies or falsifies C. Consequently FP_S is either $true$ or $false$.

It is not hard to see that FC_S is obtained from FC by removal of all the clauses that are satisfied by S and removal of the occurrences of $Var(S)$ from the rest of the clauses. It follows that if FC_{S_1} and FC_{S_2} have the same set of satisfied clauses then $FC_{S_1} = FC_{S_2}$ in other words, FC_S is completely determined by a set of satisfied clauses. Hence $|\mathbf{FCS}|$ is bounded above by the number of subsets of clauses of FCS, i.e. it is at most 2^{t_1} where t_1 is the number of clauses of FCS.

Finally let $SV^* = SV_1 \cap Var(FF)$. It is not hard to see that for an assignment S to SV_1, FF_S is completely determined by the subset of S assigning the variables of SV^*. Therefore, the number of distinct functions FF_S is at most as the number of distinct assignments to SV^*, which is 2^{t_2} where $t_2 = |SV^*|$.

Let S be an assignment on SV_1. It is not hard to see that $F_S = FP_S \wedge FC_S \wedge FF_S$. If $FP_S = false$ then $F_S = false$. Otherwise, $FP_S = true$ and hence $F_S = FC_S \wedge FF_S$. In other words, F_S is either false or there are $F_1 \in \mathbf{FCS}$ and $F_2 \in \mathbf{FFS}$ such that $F_S = F_1 \wedge F_2$. That is $|\mathbf{FS}| \leq 1 + 2^{t_1 + t_2}$.

We claim that $t_1 + t_2 \leq p+1$ implying the lemma. Indeed, the clauses of FC all belong to $B(v_q)$ by definition. Observe that $SV^* \subseteq B(v_q)$ as well. Indeed, let $Y \in SV^*$. Since Y is either X or ordered before X, there must be $j_1 \leq q$ such that $Y \in B(v_{j_1})$. On the other hand, by definition of FF, there must be $j_2 > q$ such that $Y \in B(v_{j_2})$. By the connectedness property $Y \in B(v_q)$. Since FC and SV^* are clearly disjoint being a set of 'clause vertices' and a set of 'variable vertices', the size of their union is the sum of their sizes and the size of their union cannot be larger that $|B(v_q)| \leq p + 1$, as required. ■

The upper bound can now be formally stated.

Theorem 2 *Let F be a CNF with n variables and the pathwidth p of its incidence graph. Then F can be compiled into an OBDD of size $O(2^p n)$.*

Proof. In fact we prove that the $O(2^p n)$ upper bound holds even for *uniform* OBDDs where each path from the root to a leaf includes *all* the variables. Notice that the uniformity is not required by the definition of the OBDD, only the order of variables along a computational path is essential. For instance, the OBDD shown in Figure 1 is not uniform.

Let SV be an ordering respecting f as above. Let Z be a smallest possible uniform OBDD of F with SV being the underlying ordering. It is well known that the subgraph of Z induced by any internal node u and all the vertices reachable from u (the labels on vertices and edges are retained) is an OBDD whose function is $F_{A(P)}$ where P is an arbitrary path from the root to u. Moreover, the minimality of Z implies that all the nodes marked with the same variable represent distinct functions. Indeed, if there are 2 nodes representing the same function then one of them can be removed, with the in-edges of the removed node becoming the in-edges of an-

other node associated with the same function and with possible removal of some nodes that become not reachable from the root. This produces another uniform OBDD implementing the same function and having a smaller size in contradiction to the minimality of Z.

By construction the function of a node labeled with a variable x of F is a subfunction of F obtained by an assignment to the variables preceding x in SV. According to Lemma 5 the number of such subfunctions is $O(2^p)$. Since distinct nodes labeled by x are associated with distinct subfunctions, there are $O(2^p)$ nodes labeled by x. Multiplying this by the number n of variables of F, we obtain the desired $O(2^p n)$ bound on the number of nodes of Z. ■

Corollary 2 *A* CNF *with n variables and having treewidth k can be compiled into an* OBDD *of size $O(n^k)$.*

We close this section with discussion of yet another parameter of CNFs, introduced in (Huang and Darwiche 2004), whose fixed value guarantees a linear size OBDD. In (Huang and Darwiche 2004) this parameter has not been given a name so, let us name it *combined width*. Let SV be a linear ordering on variables of the given CNF F. For each variable x in this ordering we define the *cutwidth* of x (w.r.t. to SV) as the number of clauses with one variable ordered before x and one variable ordered after x in SV. Further on, we define the *pathwidth* of x (w.r.t. to SV) as the number of variables ordered before x that occur in clauses having at least one occurrence of a variable ordered after x. The combined width of x is the minimum of the cutwidth and the pathwdith of x. The combined width of SV is the maximum over all the combined widths of the variables. Finally, the combined width of F is the minimum of combined widths of all possible orders of the variables of F. It is shown in (Huang and Darwiche 2004) that a CNF of combined width w can be complied into an OBDD of size $O(2^w n)$.

The combined width of F is a mixture of two parameters of the primary graph of F: the cutwidth (maximum cutwidth of a variable in the given permutation taken minimum over all permutations) and the pathwidth. Moreover, the combined width is not just their minimum but can in fact be much smaller than both cutwidth and pathwidth. Consider for example a CNF $F = F_1 \wedge F_2$ where F_1 and F_2 are CNFs defined as follows. $F_1 = (x \vee x_1) \wedge \ldots \wedge (x \vee x_m)$ and $F_2 = (y_1, \ldots, y_m)$ We assume that the variables of F_1 are disjoint with the variables of F_2 and that m can be arbitrarily large. The primary graph of F_1 has a large cutwidth. Indeed, for any ordering of variables of F_1 there is a subset V' of $\{x_1, \ldots, x_m\}$ of size at least $m/2$ that are either all smaller than x or all larger than x. Specify a variable $y \in V'$ that is a 'median' of V' according to the considered order. Then the cutwidth of this variable will be about $m/4$. Furthermore, the pathwidth of the primary graph of F_2 is large because this graph is just one big clique. On the other hand, the combined width of F_1 and F_2 is small. Indeed, order the variables as follows: $x, x_1, \ldots, x_m, y_1, \ldots, y_m$. Then the pathwidth index of the first $m + 1$ variables is 1 and hence the combined width will be at most 1 as well. Further, the cutwidth of the last m variable is 1 and hence the combined width of these variables is 1 as well. Thus the combined width of this order is 1 and hence the combined width of $F_1 \wedge F_2$ is at most 1 which is clearly much smaller than the minimum of the pathwdith and the cutwidth of F (determined by the respective connected components of the primary graph of F). We leave the relationship between the incidence graph treewidth and the combined width as an open question.

Discussion

In this paper we have demonstrated an infinite class of CNFs of primal graph treewidth at most k their primal graphs for which the sizes of respective OBDDs are at least $f(k)n^{k/4}$ for some function f. This result rules out the possibility of compiling a CNF into an OBDD of fixed-parameter size parameterized by the primal graph treewidth of the CNF. Our second result shows that a CNF of incidence graph treewidth at most k can be compiled into an OBDD of size at most $O(n^k)$.

Two open questions naturally arise from these results. For the first question recall that the Free Binary Decision Diagram FBDD (a.k.a. read-once branching program) is a generalization of OBDD that allows querying variables in different orders along different computational paths. Does the above lower bound hold for FBDDs realizing CNFs of bounded treewidth? The second question is: what is the space complexity of SDD parameterized by the *incidence graph treewidth* of the input CNF?

Acknowledgement

I would like to thank the anonymous reviewers for a number of insightful comments that have helped to significantly improve the quality of the paper.

References

Bodlaender, H. L., and Möhring, R. H. 1993. The pathwidth and treewidth of cographs. *SIAM J. Discrete Math.* 6(2):181–188.

Choi, A., and Darwiche, A. 2013. Dynamic minimization of sentential decision diagrams. In *AAAI*.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *J. Artif. Intell. Res. (JAIR)* 17:229–264.

Darwiche, A. 2001. Decomposable negation normal form. *J. ACM* 48(4):608–647.

Darwiche, A. 2011. SDD: A new canonical representation of propositional knowledge bases. In *IJCAI*, 819–826.

Ferrara, A.; Pan, G.; and Vardi, M. Y. 2005. Treewidth in verification: Local vs. global. In *LPAR*, 489–503.

Huang, J., and Darwiche, A. 2004. Using dpll for efficient obdd construction. In *SAT*.

Jha, A. K., and Suciu, D. 2012. On the tractability of query compilation and bounded treewidth. In *ICDT*, 249–261.

Jukna, S. 2012. *Boolean Function Complexity: Advances and Frontiers*. Springer-Verlag.

McMillan, K. L. 1994. Hierarchical representations of discrete functions, with application to model checking. In *CAV*, 41–54.

Subbarayan, S.; Bordeaux, L.; and Hamadi, Y. 2007. Knowledge compilation properties of tree-of-BDDs. In *AAAI*, 502–507.

Vatshelle, M. 2012. *New width parameters of graphs*. Ph.D. Dissertation, Department of Informatics, University of Bergen.

Wegener, I. 2000. *Branching Programs and Binary Decision Diagrams*. SIAM.

Wegener, I. 2004. Bdds–design, analysis, complexity, and applications. *Discrete Applied Mathematics* 138(1-2):229–251.

Xue, Y.; Choi, A.; and Darwiche, A. 2012. Basing decisions on sentences in decision diagrams. In *AAAI*.

Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory

Hannes Strass
Computer Science Institute
Leipzig University, Germany

Johannes Peter Wallner
Institute of Information Systems
Vienna University of Technology, Austria

Abstract

Abstract dialectical frameworks (ADFs) have recently been proposed as a versatile generalization of Dung's abstract argumentation frameworks (AFs). In this paper, we present a comprehensive analysis of the computational complexity of ADFs. Our results show that while ADFs are one level up in the polynomial hierarchy compared to AFs, there is a useful subclass of ADFs which is as complex as AFs while arguably offering more modeling capacities. As a technical vehicle, we employ the approximation fixpoint theory of Denecker, Marek and Truszczyński, thus showing that it is also a useful tool for complexity analysis of operator-based semantics.

Introduction

Formal models of argumentation are increasingly being recognized as viable tools in knowledge representation and reasoning (Bench-Capon and Dunne 2007). A particularly popular formalism are Dung's abstract argumentation frameworks (AFs) (1995). AFs treat arguments as abstract entities and natively represent only attacks between them using a binary relation. Typically, abstract argumentation frameworks are used as a target language for translations from more concrete languages. For example, the Carneades formalism for structured argumentation (Gordon, Prakken, and Walton 2007) has been translated to AFs (Van Gijzel and Prakken 2011); Caminada and Amgoud (2007) and Wyner et al. (2013) translate rule-based defeasible theories into AFs. Despite their popularity, abstract argumentation frameworks have limitations. Most significantly, their limited modeling capacities are a notable obstacle for applications: arguments can only attack one another. Furthermore, Caminada and Amgoud (2007) observed how AFs that arise as translations of defeasible theories sometimes lead to unintuitive conclusions.

To address the limitations of abstract argumentation frameworks, researchers have proposed quite a number of generalizations of AFs (Brewka, Polberg, and Woltran 2013). Among the most general of those are Brewka and Woltran's *abstract dialectical frameworks (ADFs)* (2010). ADFs are even more abstract than AFs: while in AFs arguments are abstract and the relation between arguments is fixed to attack, in ADFs also the relations are abstract (and called *links*). The relationship between different arguments (called *statements* in ADFs) is specified by *acceptance conditions*. These are Boolean functions indicating the conditions under which a statement s can be accepted when given the acceptance status of all statements with a direct link to s (its *parents*). ADFs have been successfully employed to address the shortcomings of AFs: Brewka and Gordon (2010) translated Carneades to ADFs and for the first time allowed cyclic dependencies amongst arguments; for rule-based defeasible theories we (Strass 2013b) showed how to deal with the problems observed by Caminada and Amgoud (2007).

There is a great number of semantics for AFs already, and many of them have been generalized to ADFs. Thus it might not be clear to potential ADF users which semantics are adequate for a particular application domain. In this regard, knowing the computational complexity of semantics can be a valuable guide. However, existing complexity results for ADFs are scattered over different papers, miss several semantics and some of them present upper bounds only. In this paper, we provide a comprehensive complexity analysis for ADFs. In line with the literature, we represent acceptance conditions by propositional formulas as they provide a compact and elegant way to represent Boolean functions.

Technically, we base our complexity analysis on the approximation fixpoint theory (AFT) by Denecker, Marek and Truszczyński (2000; 2003; 2004). This powerful framework provides an algebraic account of how monotone and non-monotone two-valued operators can be approximated by monotone three- or four-valued operators. (As an example of an operator to be approximated, think of the two-valued van Emden-Kowalski consequence operator from logic programming.) AFT embodies the intuitions of decades of KR research; we believe that this is very valuable also for relatively recent languages (such as ADFs), because we get the enormously influential formalizations of intuitions of Reiter and others for free. (As a liberal variation on Newton, we could say that approximation fixpoint theory allows us to take the elevator up to the shoulders of giants instead of walking up the stairs.) In fact, approximation fixpoint theory can be and partially has already been used to define some of the semantics of ADFs (Brewka et al. 2013; Strass 2013a). There, we generalized various AF and logic programming semantics to ADFs using AFT, which has provided us with two families of semantics, that we call – for rea-

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

sons that will become clear later – *approximate* and *ultimate*, respectively. Intuitively speaking, both families approximate the original two-valued model semantics of ADFs, where the ultimate family is more *precise* in a formally defined sense. The present paper employs approximating operators for complexity analysis and thus shows that AFT is also well-suited for studying the computational complexity of formalisms.

Along with providing a comparison of the approximate and ultimate families of semantics, our main results can be summarized as follows. We show that: (1) the computational complexity of ADF decision problems is one level up in the polynomial hierarchy from their AF counterparts (Dunne and Wooldridge 2009); (2) the ultimate semantics are as complex as the approximate semantics, with the notable exception of two-valued stable models; (3) there is a certain subclass of ADFs, called *bipolar* ADFs (BADFs), which is of the same complexity as AFs. Intuitively, in bipolar ADFs all links between statements are supporting or attacking. To formalize these notions, Brewka and Woltran (2010) gave a precise semantical definition of support and attack. In our work, we assume that the link types are specified by the user along with the ADF. We consider this a harmless assumption since the existing applications of ADFs produce bipolar ADFs where the link types are known (Brewka and Gordon 2010; Strass 2013b). This attractiveness of bipolar ADFs from a KR point of view is the most significant result of the paper: it shows that BADFs offer – in addition to AF-like and more general notions of attack – also syntactical notions of support *without any increase in computational cost*.

Previously, Brewka, Dunne and Woltran (2011) translated BADFs into AFs for two-valued semantics and suggested indirectly that the complexities align. Here we go a direct route, which has more practical relevance since it immediately affects algorithm design. Our work was also inspired by the complexity analysis of assumption-based argumentation by Dimopoulos, Nebel and Toni (2002) – they derived generic results in a way similar to ours.

The paper proceeds as follows. We first provide the background on approximation fixpoint theory, abstract dialectical frameworks and the necessary elements of complexity theory. In the section afterwards, we define the relevant decision problems, survey existing complexity results, use examples to illustrate how operators revise ADF interpretations and show generic upper complexity bounds. In the main section on complexity results for general ADFs, we back up the upper bounds with matching lower bounds; the section afterwards does the same for bipolar ADFs. We end with a brief discussion of related and future work. An earlier version of this paper with more details and all proofs is available as a technical report (Strass and Wallner 2013).

Background

A *complete lattice* is a partially ordered set $(A, \sqsubseteq)$ where every subset of A has a least upper and a greatest lower bound. In particular, a complete lattice has a least and a greatest element. An operator $O : A \to A$ is *monotone* if for all $x \sqsubseteq y$ we find $O(x) \sqsubseteq O(y)$. An $x \in A$ is a *fixpoint* of O if $O(x) = x$; an $x \in A$ is a *prefixpoint* of O if $O(x) \sqsubseteq x$ and a *postfixpoint* of O if $x \sqsubseteq O(x)$. Due to a fundamental result by Tarski and Knaster, for any monotone operator O on a complete lattice, the set of its fixpoints forms a complete lattice itself (Davey and Priestley 2002, Theorem 2.35). In particular, its least fixpoint $lfp(O)$ exists.

In this paper, we will be concerned with more general algebraic structures: complete partially ordered sets (CPOs). A CPO is a partially ordered set with a least element where each directed subset has a least upper bound. A set is directed iff it is nonempty and each pair of elements has an upper bound in the set. Clearly every complete lattice is a complete partially ordered set, but not necessarily vice versa. Fortunately, complete partially ordered sets still guarantee the existence of (least) fixpoints for monotone operators.

Theorem 1 ((Davey and Priestley 2002, Theorem 8.22))**.** *In a complete partially ordered set $(A, \sqsubseteq)$, any $\sqsubseteq$-monotone operator $O : A \to A$ has a least fixpoint.*

Approximation Fixpoint Theory

Denecker, Marek and Truszczyński (2000) introduce the important concept of an approximation of an operator. In the study of semantics of knowledge representation formalisms, elements of lattices represent objects of interest. Operators on lattices transform such objects into others according to the contents of some knowledge base. Consequently, fixpoints of such operators are then objects that are fully updated – informally, the knowledge base can neither increase nor decrease the amount of information in a fixpoint.

To study fixpoints of operators O, DMT study their *approximation operators* $\mathcal{O}$. When O operates on a set A, its approximation $\mathcal{O}$ operates on pairs $(x, y) \in A \times A$. Such a pair (x, y) can be seen as representing a *set* of lattice elements by providing a lower bound x and an upper bound y. Consequently, (x, y) approximates all $z \in A$ such that $x \sqsubseteq z \sqsubseteq y$. We will restrict our attention to *consistent* pairs – those where $x \sqsubseteq y$, that is, the set of approximated elements is nonempty; we denote the set of all consistent pairs over A by A^c. A pair (x, y) with $x = y$ is called *exact* – it "approximates" a single element of the original lattice.

It is natural to order approximating pairs according to their information content. Formally, for $x_1, x_2, y_1, y_2 \in A$ define the *information ordering* $(x_1, y_1) \leq_i (x_2, y_2)$ iff $x_1 \sqsubseteq x_2$ and $y_2 \sqsubseteq y_1$. This ordering and the restriction to consistent pairs leads to a complete partially ordered set $(A^c, \leq_i)$, the *consistent CPO*. For example, the *trivial pair* $(\bot, \top)$ consisting of $\sqsubseteq$-least $\bot$ and $\sqsubseteq$-greatest lattice element $\top$ approximates all lattice elements and thus contains no information – it is the least element of the CPO $(A^c, \leq_i)$; exact pairs (x, x) are the maximal elements of $(A^c, \leq_i)$.

To define an approximation operator $\mathcal{O} : A^c \to A^c$, one essentially has to define two functions: a function $\mathcal{O}' : A^c \to A$ that yields a revised *lower* bound (first component) for a given pair; and a function $\mathcal{O}'' : A^c \to A$ that yields a revised *upper* bound (second component) for a given pair. Accordingly, the overall approximation is then given by $\mathcal{O}(x, y) = (\mathcal{O}'(x, y), \mathcal{O}''(x, y))$ for $(x, y) \in A^c$. The operator $\mathcal{O} : A^c \to A^c$ is *approximating* iff it is $\leq_i$-monotone and it satisfies $\mathcal{O}'(x, x) = \mathcal{O}''(x, x)$ for all $x \in A$, that is, $\mathcal{O}$ assigns exact pairs to exact pairs. Such an $\mathcal{O}$ then *ap-*

Kripke-Kleene semantics	$lfp(\mathcal{O})$	grounded pair
admissible/reliable pair (x, y)	$(x, y) \leq_i \mathcal{O}(x, y)$	admissible pair
three-valued supported model (x, y)	$(x, y) = \mathcal{O}(x, y)$	complete pair
M-supported model (x, y)	$(x, y) \leq_i \mathcal{O}(x, y)$ and (x, y) is $\leq_i$-maximal	preferred pair
two-valued supported model (x, x)	$(x, x) = \mathcal{O}(x, x)$	model
two-valued stable model (x, x)	$x = lfp(\mathcal{O}'(\cdot, x))$	stable model

Table 1: Operator-based semantical notions (and their argumentation names on the right) for a complete lattice $(A, \sqsubseteq)$ and an approximating operator $\mathcal{O} : A^c \to A^c$ on the consistent CPO. While an approximating operator always possesses three-valued (post-)fixpoints, two-valued fixpoints need not exist. Clearly, any two-valued stable model is a two-valued supported model is a preferred pair is a complete pair is an admissible pair; furthermore the grounded semantics is a complete pair.

proximates an operator $O : A \to A$ on the original lattice iff $\mathcal{O}'(x, x) = O(x)$ for all $x \in A$.

The main contribution of Denecker, Marek and Truszczyński (2000) was the association of the *stable operator* to an approximating operator. Their original definition was four-valued; in this paper we are only interested in two-valued stable models and simplified the definitions. For an approximating operator $\mathcal{O}$ on a consistent CPO, a (two-valued) pair $(x, x) \in A^c$ is a (two-valued) *stable model of* $\mathcal{O}$ iff x is the least fixpoint of the operator $\mathcal{O}'(\cdot, x)$ defined by $w \mapsto \mathcal{O}'(w, x)$ for $w \sqsubseteq x$. This general, lattice-theoretic approach yields a uniform treatment of the standard semantics of the major nonmonotonic knowledge representation formalisms – logic programming, default logic and autoepistemic logic (Denecker, Marek, and Truszczyński 2003).

In subsequent work, Denecker, Marek and Truszczyński (2004) presented a general, abstract way to define the most precise – called the *ultimate* – approximation of a given operator O. Most precise here refers to a generalisation of $\leq_i$ to operators, where for $\mathcal{O}_1, \mathcal{O}_2$, they define $\mathcal{O}_1 \leq_i \mathcal{O}_2$ iff for all $(x, y) \in A^c$ it holds that $\mathcal{O}_1(x, y) \leq_i \mathcal{O}_2(x, y)$. Denecker, Marek and Truszczyński (2004) show that the most precise approximation of O is $\mathcal{U}_O : A^c \to A^c$ that maps (x, y) to

$$\left(\bigsqcap \{O(z) \mid x \sqsubseteq z \sqsubseteq y\}, \bigsqcup \{O(z) \mid x \sqsubseteq z \sqsubseteq y\}\right)$$

where $\sqcap$ denotes the greatest lower bound and $\sqcup$ the least upper bound in the complete lattice $(A, \sqsubseteq)$.

In recent work, we defined new operator-based semantics inspired by semantics from logic programming and abstract argumentation (Strass 2013a).[1] An overview is in Table 1.

Abstract Dialectical Frameworks

An abstract dialectical framework (ADF) is a directed graph whose nodes represent statements or positions which can be accepted or not. The links represent dependencies: the status of a node s only depends on the status of its parents (denoted $par(s)$), that is, the nodes with a direct link to s. In addition, each node s has an associated acceptance condition C_s specifying the exact conditions under which s is accepted. C_s is a function assigning to each subset of $par(s)$ one of the truth values $\mathbf{t}$, $\mathbf{f}$. Intuitively, if for some $R \subseteq par(s)$ we have $C_s(R) = \mathbf{t}$, then s will be accepted provided the nodes in R are accepted and those in $par(s) \setminus R$ are not accepted.

Definition 1. An *abstract dialectical framework* is a tuple $\Xi = (S, L, C)$ where

- S is a set of statements (positions, nodes),
- $L \subseteq S \times S$ is a set of links,
- $C = \{C_s\}_{s \in S}$ is a collection of total functions $C_s : 2^{par(s)} \to \{\mathbf{t}, \mathbf{f}\}$, one for each statement s. The function C_s is called *acceptance condition of* s.

It is often convenient to represent acceptance conditions by propositional formulas. In particular, we will do so for the complexity results of this paper. There, each C_s is represented by a propositional formula φ_s over $par(s)$. Then, clearly, $C_s(R \cap par(s)) = \mathbf{t}$ iff $R \models \varphi_s$. Furthermore, throughout the paper we will denote ADFs by Ξ and tacitly assume that $\Xi = (S, L, C)$ unless stated otherwise.

Brewka and Woltran (2010) introduced a useful subclass of ADFs called *bipolar*: Intuitively, in bipolar ADFs (BADFs) each link is supporting or attacking (or both). Formally, a link $(r, s) \in L$ is *supporting in* Ξ iff for all $R \subseteq par(s)$, we have that $C_s(R) = \mathbf{t}$ implies $C_s(R \cup \{r\}) = \mathbf{t}$; symmetrically, a link $(r, s) \in L$ is *attacking in* Ξ iff for all $R \subseteq par(s)$, we have that $C_s(R \cup \{r\}) = \mathbf{t}$ implies $C_s(R) = \mathbf{t}$. An ADF $\Xi = (S, L, C)$ is *bipolar* iff all links in L are supporting or attacking or both; we use L^+ to denote all supporting and L^- to denote all attacking links of L in Ξ. For an $s \in S$ we define $att_\Xi(s) = \{x \mid (x, s) \in L^-\}$ and $supp_\Xi(s) = \{x \mid (x, s) \in L^+\}$.

The semantics of ADFs can be defined using approximating operators. For two-valued semantics of ADFs we are interested in sets of statements, that is, we work in the complete lattice $(A, \sqsubseteq) = (2^S, \subseteq)$. To approximate elements of this lattice, we use consistent pairs of sets of statements and the associated consistent CPO $(A^c, \leq_i)$ – the *consistent CPO over S-subset pairs*. Such a pair $(X, Y) \in A^c$ can be regarded as a three-valued interpretation where all elements in X are true, those in $Y \setminus X$ are unknown and those in $S \setminus Y$ are false. (This allows us to use "pair" and "interpretation" synonymously from now on.) The following definition specifies how to revise a given three-valued interpretation.

Definition 2 ((Strass 2013a, Definition 3.1))**.** Let Ξ be an

[1] To be precise, we used a slightly different technical setting there. The results can however be transferred to the present setting (Denecker, Marek, and Truszczyński 2004, Theorem 4.2).

ADF. Define an operator $\mathcal{G}_\Xi : 2^S \times 2^S \to 2^S \times 2^S$ by

$$\mathcal{G}_\Xi(X,Y) = (\mathcal{G}'_\Xi(X,Y), \mathcal{G}'_\Xi(Y,X))$$
$$\mathcal{G}'_\Xi(X,Y) = \{s \in S \mid B \subseteq par(s), C_s(B) = \mathbf{t}, B \subseteq X, (par(s) \setminus B) \cap Y = \emptyset\}$$

Intuitively, statement s is included in the revised lower bound iff the input pair provides sufficient reason to do so, given acceptance condition C_s. Although the operator is defined for all pairs (including inconsistent ones), its restriction to consistent pairs is well-defined since it maps consistent pairs to consistent pairs. This operator defines the *approximate* family of ADF semantics according to Table 1. Based on the three-valued operator $\mathcal{G}_\Xi$, a two-valued one-step consequence operator for ADFs can be defined by $G_\Xi(X) = \mathcal{G}'_\Xi(X,X)$. The general result of Denecker, Marek and Truszczyński (2004) (Theorem 5.6) then immediately defines the ultimate approximation of G_Ξ as the operator $\mathcal{U}_\Xi$ given by $\mathcal{U}_\Xi(X,Y) = (\mathcal{U}'_\Xi(X,Y), \mathcal{U}''_\Xi(X,Y))$ with

- $\mathcal{U}'_\Xi(X,Y) = \{s \in S \mid \text{ for all } X \subseteq Z \subseteq Y, Z \models \varphi_s\}$ and
- $\mathcal{U}''_\Xi(X,Y) = \{s \in S \mid \text{ for some } X \subseteq Z \subseteq Y, Z \models \varphi_s\}$.

Incidentally, Brewka and Woltran (2010) already defined this operator, which was later used to define the *ultimate* family of ADF semantics according to Table 1 (Brewka et al. 2013).[2] In this paper, we will refer to the two families of three-valued semantics as "approximate σ" and "ultimate σ" for σ among admissible, grounded, complete, preferred and stable. For two-valued supported models (or simply models), approximate and ultimate semantics coincide.

Finally, for a propositional formula φ over vocabulary P and $X \subseteq Y \subseteq P$ we define the *partial valuation of φ by* (X,Y) as $\varphi^{(X,Y)} = \varphi[p/\mathbf{t} : p \in X][p/\mathbf{f} : p \in P \setminus Y]$. This partial evaluation takes the two-valued part of (X,Y) and replaces the evaluated variables by their truth values. Naturally, $\varphi^{(X,Y)}$ is a formula over the vocabulary $Y \setminus X$.

Complexity theory

We assume familiarity with the complexity classes P, NP and coNP, as well as with polynomial reductions and hardness and completeness for these classes. We also make use of the polynomial hierarchy, that can be defined (using oracle Turing machines) as follows: $\Sigma_0^P = \Pi_0^P = \Delta_0^P = \mathsf{P}$, $\Sigma_{i+1}^P = \mathsf{NP}^{\Sigma_i^P}$, $\Pi_{i+1}^P = \mathsf{coNP}^{\Sigma_i^P}$, $\Delta_{i+1}^P = \mathsf{P}^{\Sigma_i^P}$ for $i \geq 0$.

As a somewhat non-standard polynomial hierarchy complexity class, we use D_i^P, a generalisation of the complexity class DP to the polynomial hierarchy. A language is in DP iff it is the intersection of a language in NP and a language in coNP. Generally, a language is in D_i^P iff it is the intersection of a language in Σ_i^P and a language in Π_i^P. The canonical problem of $\mathsf{DP} = \mathrm{D}_1^P$ is SAT-UNSAT, the problem to decide for a given pair (ψ_1, ψ_2) of propositional formulas whether ψ_1 is satisfiable and ψ_2 is unsatisfiable. Obviously, by definition $\Sigma_i^P, \Pi_i^P \subseteq \mathrm{D}_i^P \subseteq \Delta_{i+1}^P$ for all $i \geq 0$.

[2] Technically, Brewka et al. (2013) represented interpretations not by pairs $(X,Y) \in A^c$ but by mappings $v : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ into the set of truth values $\mathbf{t}$ (true), $\mathbf{f}$ (false) and $\mathbf{u}$ (unknown or undecided). Clearly the two representations are interchangeable.

Preparatory Considerations

We first introduce some notation to make precise what decision problems we will analyze. For a set S, let

- $(A^c, \leq_i)$ be the consistent CPO of S-subset pairs,
- $\mathcal{O}$ an approximating operator on $(A^c, \leq_i)$,
- $\sigma \in \{adm, com, grd, pre, 2su, 2st\}$ a semantics among admissible, complete, grounded, preferred, two-valued supported and two-valued stable semantics, respectively.

In the *verification* problem we decide whether $(X,Y) \in A^c$ is a σ-model/pair of $\mathcal{O}$, denoted by $\mathsf{Ver}_\sigma^\mathcal{O}(X,Y)$. In the *existence* problem we ask whether there exists a σ-model/pair of $\mathcal{O}$ which is non-trivial, that is, different from $(\emptyset, S)$, denoted by $\mathsf{Exists}_\sigma^\mathcal{O}$. For query reasoning and $s \in S$ we consider the problem of deciding whether there exists a σ-model/pair (X,Y) of $\mathcal{O}$ such that $s \in X$, denoted by $\mathsf{Cred}_\sigma^\mathcal{O}(s)$ (*credulous* reasoning) and the problem of deciding whether in all σ-models/pairs (X,Y) of $\mathcal{O}$ we have $s \in X$, denoted by $\mathsf{Skept}_\sigma^\mathcal{O}(s)$ (*skeptical* reasoning). Note that it is no restriction to check only for truth of a statement $s \in S$, since checking for falsity can be modeled by introducing a new statement s' that behaves like the logical negation of s, by setting its acceptance condition to $\varphi_{s'} = \neg s$.

Existing results

We briefly survey – to the best of our knowledge – all existing complexity results for abstract dialectical frameworks. For general ADFs Ξ and the ultimate family of semantics, Brewka et al. (2013) have shown the following:

- $\mathsf{Ver}_{\mathsf{2su}}^{\mathcal{U}_\Xi}$ is in P, $\mathsf{Exists}_{\mathsf{2su}}^{\mathcal{U}_\Xi}$ is NP-complete (Proposition 5)
- $\mathsf{Ver}_{\mathsf{adm}}^{\mathcal{U}_\Xi}$ is coNP-complete (Proposition 10)
- $\mathsf{Ver}_{\mathsf{grd}}^{\mathcal{U}_\Xi}$ and $\mathsf{Ver}_{\mathsf{com}}^{\mathcal{U}_\Xi}$ are DP-complete (Theorem 6, Cor. 7)
- $\mathsf{Ver}_{\mathsf{2st}}^{\mathcal{U}_\Xi}$ is in DP (Proposition 8)
- $\mathsf{Exists}_{\mathsf{2st}}^{\mathcal{U}_\Xi}$ is Σ_2^P-complete (Theorem 9)

For bipolar ADFs, Brewka and Woltran (2010) showed that $\mathsf{Ver}_{\mathsf{grd}}^{\mathcal{U}_\Xi}$ is in P (Proposition 15). So particularly for BADFs, this paper will greatly illuminate the complexity landscape.

Relationship between the operators

Since $\mathcal{U}_\Xi$ is the ultimate approximation of G_Ξ it is clear that for any $X \subseteq Y \subseteq S$ we have $\mathcal{G}_\Xi(X,Y) \leq_i \mathcal{U}_\Xi(X,Y)$. In other words, the ultimate revision operator produces new bounds that are at least as tight as those of the approximate operator. More explicitly, the ultimate new lower bound always contains the approximate new lower bound: $\mathcal{G}'_\Xi(X,Y) \subseteq \mathcal{U}'_\Xi(X,Y)$; conversely, the ultimate new upper bound is contained in the approximate new upper bound: $\mathcal{U}''_\Xi(X,Y) \subseteq \mathcal{G}''_\Xi(X,Y)$. Somewhat surprisingly, it turns out that the revision operators for the upper bound coincide.

Lemma 2. *Let $\Xi = (S, L, C)$ be an ADF and $X \subseteq Y \subseteq S$.*

$$\mathcal{G}''_\Xi(X,Y) = \mathcal{U}''_\Xi(X,Y)$$

The operators for computing a new lower bound are demonstrably different, since we can find Ξ and (X,Y) with $\mathcal{U}'_\Xi(X,Y) \not\subseteq \mathcal{G}'_\Xi(X,Y)$, as the following ADF shows.

Example 1. Consider the ADF $D = (\{a\}, \{(a, a)\}, \{\varphi_a\})$ with one self-dependent statement a that has acceptance formula $\varphi_a = a \vee \neg a$. In Figure 1, we show the relevant CPO and the behavior of approximate and ultimate operators: we see that $\mathcal{G}_D(\emptyset, \{a\}) <_i \mathcal{U}_D(\emptyset, \{a\})$, which shows that in some cases the ultimate operator is strictly more precise.

So in a sense the approximate operator cannot see beyond the case distinction $a \vee \neg a$. As we will see shortly, this difference really amounts to the capability of tautology checking.

Example 2. ADF $E = (\{a, b\}, \{(b, a), (b, b)\}, \{\varphi_a, \varphi_b\})$ has acceptance formulas $\varphi_a = b \vee \neg b$ and $\varphi_b = \neg b$. So b is self-attacking and the link from b to a is redundant. In Figure 1 on the next page, we show the relevant CPO and the behavior of the operators $\mathcal{U}_E$ and $\mathcal{G}_E$ on this CPO.

The examples show that the approximate and ultimate families of semantics really are different, save for one straightforward inclusion relation in case of admissible.

Corollary 3. *For any ADF Ξ, we have the following:*

1. *An approximate admissible pair is an ultimate admissible pair, but not vice versa.*
2. *With respect to their sets of pairs, the approximate and ultimate versions of preferred/complete/grounded semantics are $\subseteq$-incomparable.*

Operator complexities

We next analyze the computational complexity of deciding whether a single statement is contained in the lower or upper bound of the revision of a given pair. This then leads to the complexity of checking whether current lower/upper bounds are pre- or postfixpoints of the revision operators for computing new lower/upper bounds, that is, whether the revisions represent improvements in terms of the information ordering. Intuitively, these results describe how hard it is to "use" the operators and lay the foundation for the rest of the complexity results.

Proposition 4. *Let Ξ be an ADF, $s \in S$ and $X \subseteq Y \subseteq S$.*

1. *Deciding $s \in \mathcal{G}'_\Xi(X, Y)$ is in* P.
2. *Deciding $\mathcal{G}'_\Xi(X, Y) \subseteq X$ is in* P.
3. *Deciding $X \subseteq \mathcal{G}'_\Xi(X, Y)$ is in* P.

Now let $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$.

4. *Deciding $s \in \mathcal{O}''(X, Y)$ is* NP*-complete.*
5. *Deciding $\mathcal{O}''(X, Y) \subseteq Y$ is* coNP*-complete.*
6. *Deciding $Y \subseteq \mathcal{O}''(X, Y)$ is* NP*-complete.*

Proof. We only show 1 and 4 as the rest follows suit.

1. Since $X \subseteq Y$, we have that whenever there exists a $B \subseteq X \cap par(s)$ with $C_s(B) = \mathbf{t}$ and $(par(s) \setminus B) \cap Y = \emptyset$, we know that $B = X \cap par(s)$. Thus $s \in \mathcal{G}'_\Xi(X, Y)$ iff $C_s(X \cap par(s)) = \mathbf{t}$ and $(par(s) \setminus X) \cap Y = \emptyset$. For acceptance functions represented by propositional formulas, both conditions can be checked in polynomial time.
4. Deciding $s \in \mathcal{G}''_\Xi(X, Y)$ is NP-complete:
   in NP: By definition, $\mathcal{G}''_\Xi(X, Y) = \mathcal{G}'_\Xi(Y, X)$. To verify $s \in \mathcal{G}'_\Xi(Y, X)$, we can guess a set $M \subseteq S$ and verify that $M \subseteq Y$, $par(s) \setminus M \subseteq S \setminus X$ and $M \models \varphi_s$.
   NP-hard: For hardness, we provide a reduction from SAT. Let ψ be a propositional formula over vocabulary P. Define an ADF $\Xi = (S, L, C)$ with $S = P \cup \{z\}$ where $z \notin P$, $\varphi_z = \psi$ and $\varphi_p = p$ for all $p \in P$. Observe that $par(z) = P$, and set $X = \emptyset$ and $Y = P$. Now $z \in \mathcal{G}''_\Xi(X, Y)$ iff $z \in \mathcal{G}'_\Xi(Y, X)$ iff $z \in \mathcal{G}'_\Xi(P, \emptyset)$ iff there is an $M \subseteq P$ with $P \setminus M \cap \emptyset = \emptyset$ and $M \models \varphi_z$ iff there is an $M \subseteq P$ with $M \models \psi$ iff ψ is satisfiable. □

These results can also be formulated in terms of partial evaluations of acceptance formulas: We have $s \in \mathcal{G}'_\Xi(X, Y)$ iff the partial evaluation $\varphi_s^{(X,Y)}$ is a formula without variables that has truth value $\mathbf{t}$. Similarly, we have $s \in \mathcal{G}''_\Xi(X, Y)$ iff the partial evaluation $\varphi_s^{(X,Y)}$ is satisfiable. Under standard complexity assumptions, computing a new lower bound with the ultimate operator is harder than with the approximate operator. This is because, intuitively, $s \in \mathcal{U}'_\Xi(X, Y)$ iff the partial evaluation $\varphi_s^{(X,Y)}$ is a tautology.

Proposition 5. *Let Ξ be an ADF, $s \in S$ and $X \subseteq Y \subseteq S$.*

1. *Deciding $s \in \mathcal{U}'_\Xi(X, Y)$ is* coNP*-complete.*
2. *Deciding $\mathcal{U}'_\Xi(X, Y) \subseteq X$ is* NP*-complete.*
3. *Deciding $X \subseteq \mathcal{U}'_\Xi(X, Y)$ is* coNP*-complete.*

Proof. We only show the first item since the remaining proofs work along the same lines. The hardness proof uses the ADF from Proposition 4.

in coNP: To decide that $s \notin \mathcal{U}'_\Xi(X, Y)$, we guess a Z with $X \subseteq Z \subseteq Y$ and verify that $Z \not\models \varphi_s$.

coNP-hard: Set $X = \emptyset$ and $Y = P$. Now $z \in \mathcal{U}'_\Xi(X, Y)$ iff $z \in \mathcal{U}'_\Xi(\emptyset, P)$ iff for all $Z \subseteq P$, we have $Z \models \varphi_z$ iff for all $Z \subseteq P$, we have $Z \models \psi$ iff ψ is a tautology. □

The next result considerably simplifies the complexity analysis of deciding the existence of non-trivial pairs.

Lemma 6. *Let $(A, \sqsubseteq)$ be a complete lattice and $\mathcal{O}$ an approximating operator on A^c. The following are equivalent:*

1. *$\mathcal{O}$ has a non-trivial admissible pair.*
2. *$\mathcal{O}$ has a non-trivial preferred pair.*
3. *$\mathcal{O}$ has a non-trivial complete pair.*

Proof. "(1) ⇒ (2)": Let $(\bot, \top) <_i (x, y) \leq_i \mathcal{O}(x, y)$. We show that there is a preferred pair $(p, q) \geq_i (x, y)$. Define $D = \{(a, b) \mid (x, y) \leq_i (a, b)\}$, then the pair $(D, \leq_i)$ is a CPO on which $\mathcal{O}$ is an approximating operator. (Obviously $(a, b) \in D$ implies $(x, y) \leq_i (a, b)$ whence by presumption and $\leq_i$-monotonicity of $\mathcal{O}$ we get $(x, y) \leq_i \mathcal{O}(x, y) \leq_i \mathcal{O}(a, b)$ and $\mathcal{O}(a, b) \in D$.) Now any sequence $(a, b) \leq_i \mathcal{O}(a, b) \leq_i \mathcal{O}(\mathcal{O}(a, b)) \leq_i \ldots$ is a non-empty chain in D and therefore has an upper bound in D. By Zorn's lemma, the set of all $\mathcal{O}$-admissible pairs in A has a maximal element $(p, q) \geq_i (x, y) >_i (\bot, \top)$.

"(2) ⇒ (3)": By (Strass 2013a, Theorem 3.10), every preferred pair is complete.

"(3) ⇒ (1)": Any complete pair is admissible (Table 1). □

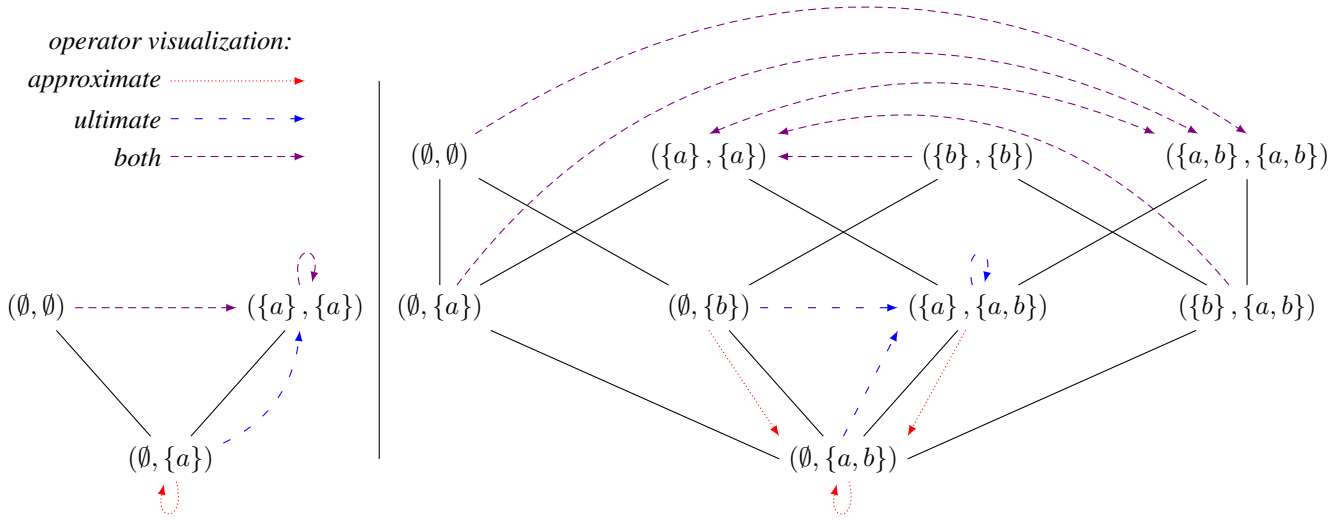


Figure 1: Hasse diagrams of consistent CPOs for the ADFs from Example 1 (left) and Example 2 (right). Solid lines represent the information ordering $\leq_i$. Directed arrows express how revision operators map pairs to other pairs. For pairs where the revisions coincide, the arrows are densely dashed and violet. When the operators revise a pair differently, we use a dotted red arrow for the ultimate and a loosely dashed blue arrow for the approximate operator. Exact (two-valued) pairs are the $\leq_i$-maximal elements. For those pairs, (and any ADF Ξ) it is clear that the operators $\mathcal{U}_\Xi$ and $\mathcal{G}_\Xi$ coincide since they approximate the same two-valued operator G_Ξ. In Example 1 on the left, we can see that the ultimate operator maps all pairs to its only fixpoint $(\{a\},\{a\})$ where a is true. The approximate operator has an additional fixpoint, $(\emptyset,\{a\})$, where a is unknown. In Example 2 on the right, the major difference between the operators is whether statement a can be derived given that b has truth value unknown. This is the case for the ultimate, but not for the approximate operator. Since there is no fixpoint in the upper row (showing the two-valued operator G_E), the ADF E does not have a two-valued model. Each of the revision operators has however exactly one three-valued fixpoint, which thus constitutes the respective grounded, preferred and complete semantics.

This directly shows the equivalence of the respective decision problems, that is, $\mathsf{Exists}^{\mathcal{O}}_{\mathsf{adm}} = \mathsf{Exists}^{\mathcal{O}}_{\mathsf{pre}} = \mathsf{Exists}^{\mathcal{O}}_{\mathsf{com}}$.

Regarding decision problems for querying, skeptical reasoning w.r.t. admissibility is trivial, i.e. $(\emptyset, S)$ is always an admissible pair in any ADF. Further credulous reasoning w.r.t. admissibility, complete and preferred semantics coincides.

Lemma 7. *Let Ξ be an ADF, $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$ and $s \in S$. Then $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{adm}}(s)$ iff $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{com}}(s)$ iff $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{pre}}(s)$.*

Proof. Assume (X, Y) with $s \in X$ is admissible w.r.t. $\mathcal{O}$, then there exists a (X', Y') with $(X, Y) \leq_i (X', Y')$ which is preferred with respect to $\mathcal{O}$ and where $s \in X'$, see proof of Lemma 6. Since any preferred pair is also complete and any complete pair is also admissible the claim follows. □

Generic upper bounds

We now show generic upper bounds for the computational complexity of the considered problems. This kind of analysis is in the spirit of the results of Dimopoulos, Nebel and Toni (2002) (Section 4). The first item is furthermore a straightforward generalization of Theorem 6.13 in (Denecker, Marek, and Truszczyński 2004).

Theorem 8. *Let S be a finite set, define $A = 2^S$ and let $\mathcal{O}$ be an approximating operator on $(A^c, \leq_i)$, the consistent CPO of S-subset pairs. For a pair $(X, Y) \in A^c$ and an $s \in S$, let the problem of deciding whether $s \in \mathcal{O}'(X, Y)$ be in Π^P_i; let the problem of deciding $s \in \mathcal{O}''(X, Y)$ be in Σ^P_i. For $(X, Y) \in A^c$ and a statement $s \in S$, we have:*

1. *The least fixpoint of $\mathcal{O}$ can be computed in polynomial time with a polynomial number of calls to a Σ^P_i-oracle.*
2. *$\mathsf{Ver}^{\mathcal{O}}_{\mathsf{adm}}(X, Y)$ is in Π^P_i; $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{adm}}(s)$ is in Σ^P_{i+1};*
3. *$\mathsf{Ver}^{\mathcal{O}}_{\mathsf{com}}(X, Y)$ is in D^P_i; $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{com}}(s)$ is in Σ^P_{i+1};*
4. *$\mathsf{Ver}^{\mathcal{O}}_{\mathsf{pre}}(X, Y)$ is in Π^P_{i+1}; $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{pre}}(s)$ is in Σ^P_{i+1}; $\mathsf{Skept}^{\mathcal{O}}_{\mathsf{pre}}(s)$ is in Π^P_{i+2}.*

Proof. 1. For any $(V, W) \in A^c$ we can use the oracle to compute an application of $\mathcal{O}'$ by simply asking whether $z \in \mathcal{O}'(V, W)$ for each $z \in S$. This means we can compute with a linear number of oracle calls the sets $\mathcal{O}'(V, W)$ and $\mathcal{O}''(V, W)$, thus the pair $\mathcal{O}(V, W)$. Hence we can compute the sequence $(\emptyset, S) \leq_i \mathcal{O}(\emptyset, S) \leq_i \mathcal{O}(\mathcal{O}(\emptyset, S)) \leq_i \ldots$ which converges to the least fixpoint of $\mathcal{O}$ after a linear number of operator applications.

2. We can decide $\mathcal{O}'(X, Y) \subseteq X$ and $Y \subseteq \mathcal{O}''(X, Y)$ in Σ^P_i, $X \subseteq \mathcal{O}'(X, Y)$ and $\mathcal{O}''(X, Y) \subseteq Y$ in Π^P_i; all by combining independent guesses. Then $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{adm}}(X, Y)$ is in Π^P_i. For $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{adm}}(s)$, we guess a pair (X_1, Y_1) with $s \in X_1$ and check if it is admissible.

3. $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{com}}(X, Y)$ is in D^P_i by the same method as for admissibility. $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{com}}(s) = \mathsf{Cred}^{\mathcal{O}}_{\mathsf{adm}}(s)$ by Lemma 7.

4. For $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{pre}}(X,Y)$, consider the co-problem, i.e. deciding whether (X,Y) is not a preferred pair. We first check if (X,Y) is a complete pair, which is in D^P_i. If this holds, we guess an (X_1,Y_1) with $(X,Y) <_i (X_1,Y_1)$ and check if it is complete. $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{pre}}(X,Y)$: coincides with credulous reasoning w.r.t. admissibility, see Lemma 7. $\mathsf{Skept}^{\mathcal{O}}_{\mathsf{pre}}(s)$: Consider the co-problem, i.e. deciding whether there exists a preferred pair (X_1,Y_1) with $X_1 \cap \{a\} = \emptyset$. We guess such a pair (X_1,Y_1) and check if it is preferred. □

Naturally, the capability of solving the functional problem of *computing* the grounded semantics allows us to solve the associated decision problems.

Corollary 9. *Under the assumptions of Theorem 8, the problems* $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{grd}}$ *and* $\mathsf{Exists}^{\mathcal{O}}_{\mathsf{grd}}$ *are in* Δ^P_{i+1}.

Complexity of General ADFs

Due to the coincidence of $\mathcal{G}''_\Xi$ and $\mathcal{U}''_\Xi$, the computational complexities of decision problems that concern only the upper bound operator also coincide. This will save both work and space in the subsequent developments. Additionally, for all containment results (except for the grounded semantics), we can use Theorem 8 and need only show hardness.

Proposition 10. *Let* Ξ *be an ADF,* $X, Y \subseteq S$ *and consider any* $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$. $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{adm}}(X,Y)$ *is* coNP*-complete.*

Proof. Hardness follows from Proposition 4, item 5. □

Recall that a pair (X,Y) is an approximate/ultimate complete pair iff it is a fixpoint of the corresponding (approximate/ultimate) operator. Given the complexities of operator computation, it is straightforward to show the following.

Proposition 11. *Let* Ξ *be an ADF,* $X \subseteq Y \subseteq S$ *and consider any* $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$. $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{com}}(X,Y)$ *is* DP*-complete.*

Next, we analyze the complexity of verifying that a given pair is the approximate (ultimate) Kripke-Kleene semantics of an ADF Ξ, that is, the least fixpoint of $\mathcal{G}_\Xi$ ($\mathcal{U}_\Xi$). Although interesting, the proof is lengthy and technical, so we unfortunately have to omit it due to space constraints. Interestingly, the membership part is the tricky one, where we encode the steps of the operator computation into propositional logic.

Theorem 12. *Let* Ξ *be an ADF,* $X \subseteq Y \subseteq S$ *and consider any operator* $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$. $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{grd}}(X,Y)$ *is* DP*-complete.*

We next ask whether there exists a *non-trivial* admissible pair, that is, if at least one statement has a truth value other than unknown. Clearly, we can guess a pair and perform the coNP-check to show that it is admissible. The next result shows that this is also the best we can do. Again, the proof is lengthy and technical and we could not include it here.

Theorem 13. *Let* Ξ *be an ADF and consider any operator* $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$. $\mathsf{Exists}^{\mathcal{O}}_{\mathsf{adm}}$ *is* Σ^P_2*-complete.*

Lemma 6 implies the same complexity for the existence of non-trivial complete and preferred pairs.

Corollary 14. *Let* Ξ *be an ADF,* $\sigma \in \{com, pre\}$ *and consider any operator* $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$. $\mathsf{Exists}^{\mathcal{O}}_{\sigma}$ *is* Σ^P_2*-complete.*

By corollary to Theorem 12, the existence of a non-trivial grounded pair can be decided in DP by testing whether the trivial pair $(\emptyset, S)$ is (not) a fixpoint of the relevant operator. The following result shows that this bound can be improved. We cannot present the proof here but can say that intuitively, half of the usual subset checks can be left out due to using the trivial pair.

Proposition 15. *Let* Ξ *be an ADF and consider any operator* $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$. $\mathsf{Exists}^{\mathcal{O}}_{\mathsf{grd}}$ *is* coNP*-complete.*

Using the result for existence of non-trivial admissible pairs, the verification complexity for the preferred semantics is straightforward to obtain, similarly as in the case of AFs (Dimopoulos and Torres 1996).

Proposition 16. *Let* Ξ *be an ADF,* $X \subseteq Y \subseteq S$ *and consider any* $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$. $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{pre}}(X,Y)$ *is* Π^P_2*-complete.*

Considering query reasoning we now show that on general ADFs credulous reasoning with respect to admissibility is harder than on AFs. By Lemma 7, the same lower bound holds for complete and preferred semantics.

Proposition 17. *Let* Ξ *be an ADF,* $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$ *be an operator and* $s \in S$. $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{adm}}(s)$ *is* Σ^P_2*-complete.*

For credulous and skeptical reasoning with respect to the grounded semantics, we first observe that the two coincide since there is always a unique grounded pair. Furthermore, a statement s is true in the approximate grounded pair iff s is true in the least fixpoint (of $\mathcal{G}_\Xi$) iff s is true in all fixpoints iff there is no fixpoint where s is unknown or false. This condition can be encoded in propositional logic and leads to the next result. For the ultimate operator we can use results for the verification problem (Brewka et al. 2013, Theorem 6). Briefly put, the problem is in coNP since the NP-hardness comes from verifying that certain arguments are undefined in the ultimate grounded pair, which is not needed for credulous reasoning. For coNP-hardness the proof of (Brewka and Woltran 2010, Proposition 13) can be easily adapted.

Proposition 18. *Let* Ξ *be an ADF,* $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$, $s \in S$. *Both* $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{grd}}(s)$ *and* $\mathsf{Skept}^{\mathcal{O}}_{\mathsf{grd}}(s)$ *are* coNP*-complete.*

Regarding skeptical reasoning for the remaining semantics we need only show the results for complete and preferred semantics, in all other cases the complexity coincides with credulous reasoning or is trivial. For complete semantics it is easy to see that skeptical reasoning coincides with skeptical reasoning under grounded semantics, since the grounded pair is the $\leq_i$-least complete pair.

Corollary 19. *Let* Ξ *be an ADF,* $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$ *and* $s \in S$. $\mathsf{Skept}^{\mathcal{O}}_{\mathsf{com}}(s)$ *is* coNP*-complete.*

Similar to reasoning on AFs, we step up one level of the polynomial hierarchy by changing from credulous to skeptical reasoning with respect to preferred semantics, which makes skeptical reasoning under preferred semantics particularly hard. We apply proof ideas by (Dunne and Bench-Capon 2002) to prove Π^P_3-hardness.

Theorem 20. *Let* Ξ *be an ADF,* $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$ *and* $s \in S$. $\mathsf{Skept}^{\mathcal{O}}_{\mathsf{pre}}(s)$ *is* Π^P_3*-complete.*

Two-valued semantics

The complexity results we have obtained so far might lead the reader to ask why we bother with the approximate operator $\mathcal{G}_\Xi$ at all: the ultimate operator $\mathcal{U}_\Xi$ is at least as precise and for all the semantics considered up to now, it has the same computational costs. We now show that for the verification of two-valued stable models, the operator for the upper bound plays no role and therefore the complexity difference between the lower bound operators for approximate (in P) and ultimate (coNP-hard) semantics comes to bear.

For the ultimate two-valued stable semantics, Brewka et al. (2013) already have some complexity results: model verification is in DP (Proposition 8), and model existence is Σ_2^P-complete (Theorem 9). We will show next that we can do better for the approximate version.

Proposition 21. *Let Ξ be an ADF and $X \subseteq Y \subseteq S$. Verifying that X is the least fixpoint of $\mathcal{G}'_\Xi(\cdot, Y)$ is in* P.

Proof sketch. Roughly, we construct the sequence defined by $X_0 = \emptyset$ and $X_{i+1} = \mathcal{G}'_\Xi(X_i, Y)$ for $i \geq 0$, as long as $X_i \subseteq Y$. By $\leq_i$-monotonicity of $\mathcal{G}_\Xi$, this sequence is monotonically $\subseteq$-increasing and so the procedure terminates after a linear number of steps. We then check if $X_{i+1} = X_i = X$, that is, the right fixpoint was reached. □

In particular, the procedure can decide whether Y is the least fixpoint of $\mathcal{G}'_\Xi(\cdot, Y)$, that is, whether (Y, Y) is a two-valued stable model of $\mathcal{G}_\Xi$. This yields the next result.

Theorem 22. *Let Ξ be an ADF and $X \subseteq S$. 1.* $\mathsf{Ver}^{\mathcal{G}_\Xi}_{\mathsf{2st}}(X, X)$ *is in* P. *2.* $\mathsf{Exists}^{\mathcal{G}_\Xi}_{\mathsf{2st}}$ *is* NP*-complete.*

The hardness direction of the second part is clear since the respective result from stable semantics of abstract argumentation frameworks carries over.

Brewka et al. (2013) showed that $\mathsf{Ver}^{\mathcal{U}_\Xi}_{\mathsf{2st}}$ is in DP (Proposition 8). As one of the most surprising results of this paper, we can improve that upper bound to coNP. The proof is not at all trivial, but basically the operator for the upper bound (contributing the NP part) is not really needed. Using the complexity of the lower revision operator $\mathcal{U}'_\Xi$, we can even show completeness for coNP.

Proposition 23. *Let Ξ be an ADF and $X \subseteq S$.* $\mathsf{Ver}^{\mathcal{U}_\Xi}_{\mathsf{2st}}(X, X)$ *is* coNP*-complete.*

We now turn to the credulous and skeptical reasoning problems for the two-valued semantics. We first recall that a two-valued pair (X, X) is a supported model (or model for short) of an ADF Ξ iff $\mathcal{G}_\Xi(X, X) = (X, X)$. Thus it could equally well be characterized by the two-valued operator by saying that X is a model iff $G_\Xi(X) = X$. Now since $\mathcal{U}_\Xi$ is the ultimate approximation of G_Ξ, also $\mathcal{U}_\Xi(X, X) = (X, X)$ in this case. Rounding up, this recalls that approximate and ultimate two-valued supported models coincide. Hence we get the following results for reasoning with this semantics.

Corollary 24. *Let Ξ be an ADF, $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$ be an operator and $s \in S$. The problem* $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{2su}}(s)$ *is* NP*-complete;* $\mathsf{Skept}^{\mathcal{O}}_{\mathsf{2su}}(s)$ *is* coNP*-complete.*

For the approximate two-valued stable semantics, the fact that model verification can be decided in polynomial time leads to the next result.

$\mathcal{O}$	$\mathcal{G}_\Xi, \mathcal{U}_\Xi$				$\mathcal{G}_\Xi, \mathcal{U}_\Xi$	$\mathcal{G}_\Xi$	$\mathcal{U}_\Xi$
σ	adm	com	pre	grd	2su	2st	2st
$\mathsf{Ver}^{\mathcal{O}}_\sigma$	coNP-c	DP-c	Π_2^P-c	DP-c	in P		coNP-c
$\mathsf{Exists}^{\mathcal{O}}_\sigma$	Σ_2^P-c	Σ_2^P-c	Σ_2^P-c	coNP-c	NP-c		Σ_2^P-c
$\mathsf{Cred}^{\mathcal{O}}_\sigma$	Σ_2^P-c	Σ_2^P-c	Σ_2^P-c	coNP-c	NP-c		Σ_2^P-c
$\mathsf{Skept}^{\mathcal{O}}_\sigma$	trivial	coNP-c	Π_3^P-c	coNP-c	coNP-c		Π_2^P-c

Table 2: Complexity results for semantics of ADFs.

Corollary 25. *Let Ξ be an ADF and $s \in S$.* $\mathsf{Cred}^{\mathcal{G}_\Xi}_{\mathsf{2st}}(s)$ *is* NP*-complete;* $\mathsf{Skept}^{\mathcal{G}_\Xi}_{\mathsf{2st}}(s)$ *is* coNP*-complete.*

For the ultimate two-valued stable semantics, things are bit more complex. The hardness reduction in the proof of Theorem 9 in (Brewka et al. 2013) makes use of a statement z that is false in any ultimate two-valued stable model. This can be used to show the same hardness for the credulous reasoning problem for this semantics: we introduce a new statement x that behaves just like $\neg z$, then x is true in some model if and only if there exists a model.

Proposition 26. *Let Ξ be an ADF and $s \in S$. The problem* $\mathsf{Cred}^{\mathcal{U}_\Xi}_{\mathsf{2st}}(s)$ *is Σ_2^P-complete.*

A similar argument works for the skeptical reasoning problem: Given a QBF $\forall P \exists Q \psi$, we construct its negation $\exists P \forall Q \neg \psi$, whose associated ADF D has an ultimate two-valued stable model (where z is false) iff $\exists P \forall Q \neg \psi$ is true iff the original QBF $\forall P \exists Q \psi$ is false. Hence $\forall P \exists Q \psi$ is true iff z is true in all ultimate two-valued stable models of D.

Proposition 27. *Let Ξ be an ADF and $s \in S$. The problem* $\mathsf{Skept}^{\mathcal{U}_\Xi}_{\mathsf{2st}}(s)$ *is Π_2^P-complete.*

Complexity of Bipolar ADFs

We first note that since BADFs are a subclass of ADFs, all membership results from the previous section immediately carry over. However, we can show that many problems will in fact become easier. Intuitively, computing the revision operators is now P-easy because the associated satisfiability/tautology problems only have to treat restricted acceptance formulas. In bipolar ADFs, by definition, if in some three-valued pair (X, Y) a statement s is accepted by a revision operator ($s \in \mathcal{O}'(X, Y)$), it will stay so if we set its undecided supporters to true and its undecided attackers to false. Symmetrically, if a statement is rejected by an operator ($s \notin \mathcal{O}''(X, Y)$), it will stay so if we set its undecided supporters to false and its undecided attackers to true. This is the key idea underlying the next result.

Proposition 28. *Let Ξ be a BADF with $L = L^+ \cup L^-$, $\mathcal{O} \in \{\mathcal{G}_\Xi, \mathcal{U}_\Xi\}$, $s \in S$ and $X \subseteq Y \subseteq S$.*

1. *Deciding $s \in \mathcal{O}'(X, Y)$ is in* P.
2. *Deciding $s \in \mathcal{O}''(X, Y)$ is in* P.

Using the generic upper bounds of Theorem 8, it is now straightforward to show membership results for BADFs with known link types.

Corollary 29. *Let Ξ be a BADF with $L = L^+ \cup L^-$, consider any operator $\mathcal{O} \in \{\mathcal{G}_{\Xi}, \mathcal{U}_{\Xi}\}$ and semantics $\sigma \in \{adm, com\}$. For $X \subseteq Y \subseteq S$ and $s \in S$, we find that*

- $\mathsf{Ver}^{\mathcal{O}}_{\sigma}(X, Y)$ *and* $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{grd}}(X, Y)$ *are in* P*;*
- $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{pre}}(X, Y)$ *is in* coNP*;*
- $\mathsf{Exists}^{\mathcal{O}}_{\sigma}$, $\mathsf{Exists}^{\mathcal{O}}_{\mathsf{pre}}$, $\mathsf{Cred}^{\mathcal{O}}_{\sigma}(s)$ *and* $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{pre}}(s)$ *are in* NP*;*
- $\mathsf{Exists}^{\mathcal{O}}_{\mathsf{grd}}$, $\mathsf{Cred}^{\mathcal{O}}_{\mathsf{grd}}(s)$, $\mathsf{Skept}^{\mathcal{O}}_{\mathsf{grd}}(s)$, $\mathsf{Skept}^{\mathcal{O}}_{\mathsf{com}}(s)$ *are in* P*;*
- $\mathsf{Skept}^{\mathcal{O}}_{\mathsf{pre}}(s)$ *is in* Π^P_2*.*

Proof. Membership is due to Theorem 8 and the complexity bounds of the operators in BADFs in Proposition 28, just note that $\Sigma^P_0 = \Pi^P_0 = \mathsf{P}$. $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{grd}}(X, Y)$ is in $\mathsf{P}^{\mathsf{P}} = \mathsf{P}$ by Corollary 9. For the existence of non-trivial pairs we can simply guess and check in polynomial time for admissible pairs and thus also for complete and preferred semantics. □

Hardness results straightforwardly carry over from AFs.

Proposition 30. *Let Ξ be a BADF with $L = L^+ \cup L^-$, consider any operator $\mathcal{O} \in \{\mathcal{G}_{\Xi}, \mathcal{U}_{\Xi}\}$ and semantics $\sigma \in \{adm, com, pre\}$. For $X \subseteq Y \subseteq S$ and $s \in S$:*

- $\mathsf{Ver}^{\mathcal{O}}_{\mathsf{pre}}(X, Y)$ *is* coNP*-hard;*
- $\mathsf{Exists}^{\mathcal{O}}_{\sigma}$ *and* $\mathsf{Cred}^{\mathcal{O}}_{\sigma}(s)$ *are* NP*-hard;*
- $\mathsf{Skept}^{\mathcal{O}}_{\mathsf{pre}}(s)$ *is* Π^P_2*-hard.*

Proof. Hardness results from AFs for these problems carry over to BADFs as for all semantics AFs are a special case of BADFs (Brewka et al. 2013; Strass 2013a). The complexities of the problems on AFs for admissible and preferred semantics are shown by (Dimopoulos and Torres 1996), except for the Π^P_2-completeness result of skeptical preferred semantics, which is shown by (Dunne and Bench-Capon 2002). The complete semantics is studied by (Coste-Marquis, Devred, and Marquis 2005). □

We next show that there is no hope that the existence problems for approximate and ultimate two-valued stable models coincide as there are cases when the semantics differ.

Example 3. Consider the BADF $F = (S, L, C)$ with statements $S = \{a, b, c\}$ and acceptance formulas $\varphi_a = \mathbf{t}$, $\varphi_b = a \vee c$ and $\varphi_c = a \vee b$. The only two-valued supported model is (S, S) where all statements are true. This pair is also an ultimate two-valued stable model, since $\mathcal{U}'_F(\emptyset, S) = \{a\}$, and both $\varphi_b^{(\{a\},S)} = \mathbf{t} \vee c$ and $\varphi_c^{(\{a\},S)} = \mathbf{t} \vee b$ are tautologies, whence $\mathcal{U}'_F(\{a\}, S) = S$. However, (S, S) is not an approximate two-valued stable model: although $\mathcal{G}'_F(\emptyset, S) = \{a\}$, then $\mathcal{G}'_F(\{a\}, S) = \{a\}$ and we thus cannot reconstruct the upper bound S. Thus F has no approximate two-valued stable models.

So approximate and ultimate two-valued stable model semantics are indeed different. However, we can show that the respective existence problems have the same complexity.

Proposition 31. *Let Ξ be a BADF with $L = L^+ \cup L^-$, $\mathcal{O} \in \{\mathcal{G}_{\Xi}, \mathcal{U}_{\Xi}\}$ an operator and semantics $\sigma \in \{2su, 2st\}$. For $X \subseteq S$, $\mathsf{Ver}^{\mathcal{O}}_{\sigma}(X, X)$ is in P; $\mathsf{Exists}^{\mathcal{O}}_{\sigma}$ is NP-complete.*

σ	adm	com	pre	grd	2su	2st
$\mathsf{Ver}^{\mathcal{O}}_{\sigma}$	in P	in P	coNP-c	in P	in P	in P
$\mathsf{Exists}^{\mathcal{O}}_{\sigma}$	NP-c	NP-c	NP-c	in P	NP-c	NP-c
$\mathsf{Cred}^{\mathcal{O}}_{\sigma}$	NP-c	NP-c	NP-c	in P	NP-c	NP-c
$\mathsf{Skept}^{\mathcal{O}}_{\sigma}$	trivial	in P	Π^P_2-c	in P	coNP-c	coNP-c

Table 3: Complexity results for semantics of bipolar Abstract Dialectical Frameworks for $\mathcal{O} \in \{\mathcal{G}_{\Xi}, \mathcal{U}_{\Xi}\}$.

Proof. Membership carries over – for supported models from (Brewka et al. 2013, Proposition 5), for approximate stable models from Theorem 22. For membership for ultimate stable models, we can use Proposition 28 to adapt the decision procedure of Proposition 21. In any case, hardness carries over from AFs (Dimopoulos and Torres 1996). □

For credulous and skeptical reasoning over the two-valued semantics, membership is straightforward and hardness again carries over from argumentation frameworks.

Corollary 32. *Let Ξ be a BADF with $L = L^+ \cup L^-$; consider any operator $\mathcal{O} \in \{\mathcal{G}_{\Xi}, \mathcal{U}_{\Xi}\}$ and semantics $\sigma \in \{2su, 2st\}$. For $s \in S$, $\mathsf{Cred}^{\mathcal{O}}_{\sigma}(s)$ is NP-complete; $\mathsf{Skept}^{\mathcal{O}}_{\sigma}(s)$ is coNP-complete.*

Discussion

In this paper we studied the computational complexity of abstract dialectical frameworks using approximation fixpoint theory. We showed numerous novel results for two families of ADF semantics, the approximate and ultimate semantics, which are themselves inspired by argumentation and AFT. We showed that in most cases the complexity increases by one level of the polynomial hierarchy compared to the corresponding reasoning tasks on AFs. A notable difference between the two families of semantics lies in the stable semantics, where the approximate version is easier than its ultimate counterpart. For the restricted, yet powerful class of bipolar ADFs we proved that for the corresponding reasoning tasks AFs and BADFs have the same complexity, which suggests that many types of relations between arguments can be introduced without increasing the worst-time complexity. On the other hand, our results again emphasize that arbitrary (non-bipolar) ADFs cannot be compiled into equivalent Dung AFs in deterministic polynomial time, unless the polynomial hierarchy collapses to the first level. Under the same assumption, ADFs cannot be implemented directly with methods that are typically applied to AFs, for example answer-set programming (Egly, Gaggl, and Woltran 2010).

Our results lay the foundation for future algorithms and their implementation, for example augmenting the ADF system DIAMOND (Ellmauthaler and Strass 2013) to support also the approximate semantics family, as well as devising efficient methods for the interesting class of BADFs.

For further future work several promising directions are possible. Studying easier fragments of ADFs as well as parameterized complexity analysis can lead to efficient decision

procedures, as is witnessed for AFs (Dvořák et al. 2014; Dvořák, Ordyniak, and Szeider 2012). We also deem it auspicious to use alternative representations of acceptance conditions, for instance by employing techniques from knowledge compilation (Darwiche and Marquis 2002).

A detailed complexity analysis of other useful AF semantics would also reveal further insights, e.g. semi-stable semantics (Caminada, Carnielli, and Dunne 2012), naive-based semantics, such as cf2 (Baroni, Giacomin, and Guida 2005), or a recently proposed extension-based semantics for ADFs (Polberg, Wallner, and Woltran 2013). For semantical analysis, it would be useful to consider principle-based evaluations for ADFs (Baroni and Giacomin 2007). Furthermore it appears natural to compare (ultimate) ADF semantics and ultimate logic programming semantics (Denecker, Marek, and Truszczyński 2004) in approximation fixpoint theory, in particular with respect to computational complexity.

Acknowledgements. Thanks to an anonymous reviewer for pointing out a notational issue in our usage of the complexity classes D_i^P. This research was supported by DFG (project BR 1817/7-1) and FWF (project I1102).

References

Baroni, P., and Giacomin, M. 2007. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence* 171(10-15):675–700.

Baroni, P.; Giacomin, M.; and Guida, G. 2005. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence* 168(1-2):162–210.

Bench-Capon, T. J. M., and Dunne, P. E. 2007. Argumentation in Artificial Intelligence. *Artificial Intelligence* 171(10-15):619–641.

Brewka, G., and Gordon, T. F. 2010. Carneades and Abstract Dialectical Frameworks: A Reconstruction. In *Proceedings of COMMA 2010*, volume 216 of *FAIA*, 3–12. IOS Press.

Brewka, G., and Woltran, S. 2010. Abstract Dialectical Frameworks. In *Proceedings of KR 2010*, 102–111. AAAI Press.

Brewka, G.; Ellmauthaler, S.; Strass, H.; Wallner, J. P.; and Woltran, S. 2013. Abstract Dialectical Frameworks Revisited. In *Proceedings of IJCAI 2013*, 803–809. IJCAI/AAAI.

Brewka, G.; Dunne, P. E.; and Woltran, S. 2011. Relating the Semantics of Abstract Dialectical Frameworks and Standard AFs. In *Proceedings of IJCAI 2011*, 780–785. IJCAI/AAAI.

Brewka, G.; Polberg, S.; and Woltran, S. 2013. Generalizations of Dung frameworks and their role in formal argumentation. *IEEE Intelligent Systems* PP(99). Special Issue on Representation and Reasoning. In press.

Caminada, M. W., and Amgoud, L. 2007. On the evaluation of argumentation formalisms. *Artificial Intelligence* 171(5–6):286–310.

Caminada, M. W.; Carnielli, W. A.; and Dunne, P. E. 2012. Semi-stable Semantics. *Journal of Logic and Computation* 22(5):1207–1254.

Coste-Marquis, S.; Devred, C.; and Marquis, P. 2005. Symmetric Argumentation Frameworks. In *ECSQARU*, volume 3571 of *LNCS*, 317–328. Springer.

Darwiche, A., and Marquis, P. 2002. A Knowledge Compilation Map. *Journal of Artificial Intelligence Research* 17:229–264.

Davey, B., and Priestley, H. 2002. *Introduction to Lattices and Order*. Cambridge University Press, second edition.

Denecker, M.; Marek, V. W.; and Truszczyński, M. 2000. Approximations, Stable Operators, Well-Founded Fixpoints and Applications in Nonmonotonic Reasoning. In *Logic-Based Artificial Intelligence*. Kluwer Academic Publishers. 127–144.

Denecker, M.; Marek, V. W.; and Truszczyński, M. 2003. Uniform Semantic Treatment of Default and Autoepistemic Logics. *Artificial Intelligence* 143(1):79–122.

Denecker, M.; Marek, V. W.; and Truszczyński, M. 2004. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Information and Computation* 192(1):84–121.

Dimopoulos, Y., and Torres, A. 1996. Graph Theoretical Structures in Logic Programs and Default Theories. *Theoretical Computer Science* 170(1-2):209–244.

Dimopoulos, Y.; Nebel, B.; and Toni, F. 2002. On the computational complexity of assumption-based argumentation for default reasoning. *Artificial Intelligence* 141(1/2):57–78.

Dung, P. M. 1995. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence* 77(2):321–358.

Dunne, P. E., and Bench-Capon, T. J. M. 2002. Coherence in Finite Argument Systems. *Artificial Intelligence* 141(1/2):187–203.

Dunne, P. E., and Wooldridge, M. 2009. Complexity of abstract argumentation. In Simari, G., and Rahwan, I., eds., *Argumentation in Artificial Intelligence*. Springer. 85–104.

Dvořák, W.; Järvisalo, M.; Wallner, J. P.; and Woltran, S. 2014. Complexity-Sensitive Decision Procedures for Abstract Argumentation. *Artificial Intelligence* 206:53–78.

Dvořák, W.; Ordyniak, S.; and Szeider, S. 2012. Augmenting tractable fragments of abstract argumentation. *Artificial Intelligence* 186:157–173.

Egly, U.; Gaggl, S. A.; and Woltran, S. 2010. Answer-set programming encodings for argumentation frameworks. *Argument and Computation* 1(2):147–177.

Ellmauthaler, S., and Strass, H. 2013. The DIAMOND System for Argumentation: Preliminary Report. In *Proceedings of ASPOCP*, 97–107.

Gordon, T. F.; Prakken, H.; and Walton, D. 2007. The Carneades model of argument and burden of proof. *Artificial Intelligence* 171(10–15):875–896.

Polberg, S.; Wallner, J. P.; and Woltran, S. 2013. Admissibility in the Abstract Dialectical Framework. In *Proceedings of CLIMA XIV*, volume 8143 of *LNAI*, 102–118. Springer.

Strass, H., and Wallner, J. P. 2013. Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. Technical Report 2, Computer Science Institute, Leipzig University. http://nbn-resolving.de/urn:nbn:de:bsz:15-qucosa-129614.

Strass, H. 2013a. Approximating operators and semantics for abstract dialectical frameworks. *Artificial Intelligence* 205:39–70.

Strass, H. 2013b. Instantiating knowledge bases in abstract dialectical frameworks. In *Proceedings of CLIMA XIV*, volume 8143 of *LNAI*, 86–101. Springer.

Van Gijzel, B., and Prakken, H. 2011. Relating Carneades with abstract argumentation. In *Proceedings of IJCAI 2011*, 1113–1119. IJCAI/AAAI.

Wyner, A.; Bench-Capon, T. J. M.; and Dunne, P. E. 2013. On the instantiation of knowledge bases in abstract argumentation frameworks. In *Proceedings of CLIMA XIV*, volume 8143 of *LNAI*, 34–50. Springer.

Skolemization for Weighted First-Order Model Counting

Guy Van den Broeck
Computer Science Department
University of California, Los Angeles
guyvdb@cs.ucla.edu

Wannes Meert
Computer Science Department
KU Leuven
wannes.meert@cs.kuleuven.be

Adnan Darwiche
Computer Science Department
University of California, Los Angeles
darwiche@cs.ucla.edu

Abstract

First-order model counting emerged recently as a novel reasoning task, at the core of efficient algorithms for probabilistic logics. We present a Skolemization algorithm for model counting problems that eliminates existential quantifiers from a first-order logic theory without changing its weighted model count. For certain subsets of first-order logic, lifted model counters were shown to run in time polynomial in the number of objects in the domain of discourse, where propositional model counters require exponential time. However, these guarantees apply only to Skolem normal form theories (i.e., no existential quantifiers) as the presence of existential quantifiers reduces lifted model counters to propositional ones. Since textbook Skolemization is not sound for model counting, these restrictions precluded efficient model counting for directed models, such as probabilistic logic programs, which rely on existential quantification. Our Skolemization procedure extends the applicability of first-order model counters to these representations. Moreover, it simplifies the design of lifted model counting algorithms.

1 Introduction

Weighted model counting (WMC) is a generalization of model counting (Gomes, Sabharwal, and Selman 2009). In model counting, also known as #SAT, one counts the number of satisfying assignments of a propositional sentence. In WMC, each assignment has an associated weight and the task is to compute the *sum of the weights* of all satisfying assignments. One application of WMC is to probabilistic graphical models. For example, exact inference algorithms for Bayesian networks encode probabilistic inference as a WMC task, which can then be solved by knowledge compilation (Darwiche 2002) or exhaustive DPLL search (Sang, Beame, and Kautz 2005).

WMC also plays an important role in *first-order probabilistic* representations. These became popular in recent years, in statistical relational learning (Getoor and Taskar 2007) and probabilistic logic learning (De Raedt et al. 2008), which are concerned with modeling and learning complex logical and probabilistic interactions between large numbers of objects. Efficient algorithms again reduce exact probabilistic inference to a WMC problem on a propositional knowledge base (Chavira, Darwiche, and Jaeger 2006; Fierens et al. 2011; 2013). Encoding first-order probabilistic models into propositional logic retains a key advantage of the Bayesian network algorithms: WMC naturally exploits determinism and local structure in the probabilistic model (Boutilier et al. 1996; Chavira and Darwiche 2005). A disadvantage is that the high-level first-order structure is lost. Poole (2003) observed that knowing the *symmetries* that are abundant in first-order structure can speed up probabilistic inference. Lifted inference algorithms reason about groups of objects as a whole, similar to the high-level reasoning of first-order resolution. This has lead Van den Broeck et al. (2011) and Gogate and Domingos (2011) to propose *weighted first-order model counting* (WFOMC) as the core reasoning task underlying lifted inference algorithms. WFOMC assigns a weight to interpretations in finite-domain, function-free first-order logic, and computes the sum of the weights of all models.

Counting models at the first-order level has computational advantages. For certain classes of theories, knowing the first-order structure gives exponential speedups (Van den Broeck 2011). For example, counting the models of a first-order universally quantified CNF with up to two logical variables per clause can always be done in time polynomial in the size of the domain of discourse. In contrast, a propositionalization of these CNFs will often have a treewidth polynomial in the domains size, and propositional model counting runs in exponential time.

One major limitation of first-order model counters, however, is that they require input in *Skolem normal form* (i.e., without existential quantifiers). This is a common requirement for first-order automated reasoning algorithms, such as theorem provers. It is usually dealt with by Skolemization, which introduces Skolem constants and functions. However, the introduction of functions is problematic for first-order model counters as they expect a function-free input.

The main contribution of this paper is a *Skolemization procedure* that is specific for weighted first-order model counting. The procedure maps a logical input theory to an output theory that is devoid of existential quantifiers and functions, yet has an identical weighted first-order model count. The procedure is modular, in that it remains sound when extending the input and output theories with a new

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

sentence. Furthermore, it is purely first-order as it is independent of the domain of discourse.

The proposed Skolemization algorithm has a range of implications. First, it opens up new possibilities for lifted inference algorithms. For example, on Markov Logic Networks with quantifiers (Richardson and Domingos 2006), and various forms of *Probabilistic Logic Programs* (e.g., De Raedt, Kimmig, and Toivonen (2007)), lifted algorithms currently provide little or no benefit over propositional ones. The main reason is that the WFOMC form of these representations generally contain existential quantifiers. The proposed Skolemization algorithm allows us, for the first time, to perform lifted inference on these representations. Second, there are *liftability theorems* that define classes of theories for which WFOMC is domain-lifted, meaning that it runs in time polynomial in the domain size (Jaeger and Van den Broeck 2012). These theorems had to assume Skolem normal form for the mentioned reason, but now apply more generally. Finally, the Skolemization algorithm averts the need for special inference rules that deal with existential quantifiers, *simplifying the design* of future WFOMC algorithms.

2 Weighted First-Order Model Counting

We start by formally defining the weighted first-order model counting task. We also compare it to propositional weighted model counting and discuss existing algorithms.

Background

Throughout this paper, we will work with the *function-free finite-domain* fragment of first-order logic (FOL), which we now briefly review. An *atom* $\mathtt{P}(t_1, \ldots, t_n)$ consists of predicate $\mathtt{P}/n$ of arity n followed by n arguments, which are either *constants* from a finite domain $\mathbf{D} = \{\mathsf{A}, \mathsf{B}, \ldots\}$ or *logical variables* $\{x, y, \ldots\}$. We use $\mathbf{y}$ to denote a sequence of logical variables. A *literal* is an atom or its negation. A *formula* combines atoms with logical connectives and quantifiers $\exists$ and $\forall$. A logical variable x is *quantified* if it is enclosed by a $\forall x$ or $\exists x$. A *free variable* is one that is not quantified. A *sentence* is a formula without free variables. A formula is *ground* if it contains no logical variables. A *clause* is a disjunction of literals and a *CNF* is a conjunction of clauses. The *groundings* of a quantifier-free formula is the set of formulas obtained by instantiating the free variables with any possible combination of constants from $\mathbf{D}$. The grounding of $\forall x, \phi$ and $\exists x, \phi$ is the conjunction resp. disjunction of all groundings of ϕ.

We will make use of *Herbrand semantics* (Hinrichs and Genesereth 2006), as is customary in statistical relational learning and probabilistic logic learning. The *Herbrand base* of sentence Δ for domain $\mathbf{D}$ is the set of all ground atoms that can be constructed from predicates and constants in $\mathbf{D}$. A *Herbrand interpretation* is a truth-value assignment to all atoms in the Herbrand base. We will find it convenient to represent interpretations as sets of literals. A Herbrand model of Δ is a Herbrand interpretation ω that satisfies Δ, denoted by $\omega \models_{\mathbf{D}} \Delta$.

Definitions

We first review propositional weighted model counting.

Definition 1 (WMC). Given
- a *sentence* Δ in propositional logic over literals $\mathcal{L}$, and
- a *weight function* $\mathrm{w} : \mathcal{L} \to \mathbb{R}^{\geq 0}$,

the *weighted model count* (WMC) is

$$\mathrm{WMC}(\Delta, \mathrm{w}) = \sum_{\omega \models \Delta} \prod_{l \in \omega} \mathrm{w}(l).$$

WFOMC lifts WMC to the first-order level as follows.

Definition 2 (WFOMC[1]). Given
- a *sentence* Δ in FOL containing predicates $\mathcal{P}$,
- a set of constants $\mathbf{D}$, including the constants in Δ, and
- a pair of *weight functions* $\mathrm{w}, \bar{\mathrm{w}} : \mathcal{P} \to \mathbb{R}$,

the *weighted first-order model count* (WFOMC) is

$$\mathrm{WFOMC}(\Delta, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}}) = \sum_{\omega \models_{\mathbf{D}} \Delta} \prod_{l \in \omega_0} \bar{\mathrm{w}}(\mathrm{pred}(l)) \prod_{l \in \omega_1} \mathrm{w}(\mathrm{pred}(l)),$$

where ω_0 and ω_1 consists of the true, respectively false, literals in ω, and pred maps literals to their predicate.

The weight functions assign a weight to each predicate. The weight of a positive (negative) literal is the weight of its predicate in w ($\bar{\mathrm{w}}$). The weight of a model is the product of its literal weights. Finally, the total count is the sum of the weights of all the Herbrand models of Δ.

Our WFOMC definition deviates from WMC in two ways. First, WMC directly assigns weights to *literals*. WFOMC instead assigns weights to *predicates*, and defines literal weights in terms of predicate weights. This distinguishes WFOMC from probabilistic databases (see Section 6). If for modeling reasons, certain literals need to be assigned unique weights, this can always be achieved by introducing new predicates.

Second, our definition permits predicate weights to be *negative* numbers. Negative weights will turn out to be crucial for our Skolemization algorithm. Historically, the WMC weight function has mostly been used to represent probabilities. This led to the (sometimes implicit) assumption that weights are between zero and one, or at least non-negative. Nevertheless, all exact weighted model counters we are aware of can handle negative weights.[2] It appears that the positive weight assumption is more intrinsic to approximate weighted model counters (Wei and Selman 2005; Gogate and Dechter 2011). Section 6 discusses negative weights in more detail.

Motivation

A WFOMC problem can always be propositionalized into a WMC problem. We can ground Δ for $\mathbf{D}$, turn every atom

[1] This definition is based on Van den Broeck et al. (2011). WFOMC is called *Lifted WMC* in Gogate and Domingos (2011).

[2] In fact, the only underlying requirement of exact model counting approaches is that literal weights are elements from a *commutative semiring* (Kimmig, Van den Broeck, and De Raedt 2012).

in the Herbrand base into a propositional atom, and associate with every propositional literal the weight of its original predicate. One may wonder why we define this task at the first-order level.

Our motivation is computational. Similar to how a single step of first-order resolution can perform a large number of propositional resolution steps, a WFOMC solver can often provide exponential speedups over WMC solvers. First-order quantifiers make statements about groups of symmetric objects, which we can reason about jointly.

Without going into algorithmic details, we will now illustrate this principle on concrete examples. For the sake of simplicity, the examples are non-weighted model counting problems, corresponding to WFOMC problems where $\mathrm{w}(\mathtt{P}) = \bar{\mathrm{w}}(\mathtt{P}) = 1$ for all predicates $\mathtt{P}$. Consider Δ to be

$$\mathtt{Stress}(\mathsf{A}) \Rightarrow \mathtt{Smokes}(\mathsf{A}). \tag{1}$$

Assuming that $\mathbf{D} = \{\mathsf{A}\}$, every interpretation of $\mathtt{Stress}(\mathsf{A})$ and $\mathtt{Smokes}(\mathsf{A})$ satisfies Δ, except when $\mathtt{Stress}(\mathsf{A})$ is true and $\mathtt{Smokes}(\mathsf{A})$ is false. Therefore, the model count is 3. Now let Δ be

$$\forall x,\ \mathtt{Stress}(x) \Rightarrow \mathtt{Smokes}(x). \tag{2}$$

Without changing $\mathbf{D}$, the model count is still 3. When we expand $\mathbf{D}$ to contain n constants, we get n independent copies of Formula 1. For each person x, atoms $\mathtt{Stress}(x)$ and $\mathtt{Smokes}(x)$ can jointly take 3 values, and the total model count becomes 3^n.

This example already demonstrates the benefits of first-order counting. A *propositional* model counter on the groundings of Formula 2 would detect that all n clauses are independent, recompute for every clause that it has 3 models, and multiply these counts n times. Propositional model counters have no elementary operation for exponentiation. A *first-order* model counter reads from the first-order structure that it suffices to compute the model count of a single ground clause, and then knows to exponentiate. It never actually grounds the formula, and given the size of $\mathbf{D}$, it runs in logarithmic time. This gives an exponential speedup over propositional counting, which runs in linear time.

These first-order counting techniques can interplay with propositional ones. Take for example Δ to be

$$\forall y,\ \mathtt{ParentOf}(y) \wedge \mathtt{Female} \Rightarrow \mathtt{MotherOf}(y). \tag{3}$$

This sentence is about a specific individual who may be a female, depending on whether the proposition $\mathtt{Female}$ is true. We can separately count the models in which $\mathtt{Female}$ is true, and those in which it is false (i.e., a Shannon decomposition). When $\mathtt{Female}$ is false, Δ is satisfied, and the $\mathtt{ParentOf}$ and $\mathtt{MotherOf}$ atoms can take on any value. This gives 4^n models. When $\mathtt{Female}$ is true, Δ is structurally identical to Formula 2, and has 3^n models. The total model count is then $3^n + 4^n$.

These concepts can be applied recursively to count more complicated formulas. Take for example

$$\forall x, \forall y,\ \mathtt{ParentOf}(x, y) \wedge \mathtt{Female}(x) \Rightarrow \mathtt{MotherOf}(x, y).$$

There is now a partition of the ground clauses into n independent sets of n clauses. The sets correspond to values of x, and the individual clauses to values of y. The formula for each specific x, that is, each set of clauses, is structurally identical to Formula 3 and has count of $3^n + 4^n$. The total model count is then $(3^n + 4^n)^n$.

The most impressive improvements are attained when propositional model counters run in time *exponential* in n, yet first-order model counters run in *polynomial* time. To consider an example where this comes up, let Δ be

$$\forall x, \forall y,\ \mathtt{Smokes}(x) \wedge \mathtt{Friends}(x, y) \Rightarrow \mathtt{Smokes}(y). \tag{4}$$

This time, the clauses in the grounding of Δ are no longer independent, and it would be wrong to simply exponentiate their counts. Let us first assume that we know a partial interpretation of the $\mathtt{Smokes}$ atoms with k positive literals (i.e., k people smoke). The question is now: how many models extend this partial interpretation? Formula 4 encodes that a smoker cannot be friends with a non smoker. Hence, out of n^2 $\mathtt{Friends}$ atoms, $k(n-k)$ have to be false, and the others can take either truth value. Thus, the number of models is $2^{n^2-k(n-k)}$. Second, we know that there are $\binom{n}{k}$ partial interpretations with k smokers, and k can range from 0 to n. This results in the total model count of

$$\sum_{k=0}^{n} \binom{n}{k} 2^{n^2-k(n-k)}.$$

In fact, the systems discussed in the next section can automatically construct this formula and compute the model count of Formula 4 in time polynomial in n. On the other hand, existing propositional WMC algorithms require time that is exponential in n on this problem. We note here that the *treewidth* of the grounding of Δ is linear in n.

There are space considerations that motivate first-order model counting as well. When converting a WFOMC problem to WMC, the grounding of Δ has size polynomial in the size of $\mathbf{D}$, but the degree of this polynomial can be high. When the grounding does not fit into memory, even approximate WMC becomes a problem.

Algorithms

Several algorithms exist for solving propositional WMC. Exact solvers are based on either exhaustive DPLL search (Sang, Beame, and Kautz 2005), or knowledge compilation to a circuit language that supports efficient model counting, such as `d-DNNF` (Darwiche 2002; Chavira and Darwiche 2008) or `SDD` (Choi, Kisa, and Darwiche 2013). Approximate WMC algorithms use local search (Wei and Selman 2005) or sampling (Gogate and Dechter 2011).

More recently, algorithms were introduced that directly solve the WFOMC task. They take a WFOMC problem and automatically generate and evaluate the types of expressions shown in the previous section. Their elementary operations include exponentiation, summation and binomial coefficients. They are called *lifted inference* algorithms. In particular, two lifted algorithms were proposed for *exact* WFOMC, one based on first-order knowledge compilation (Van den Broeck et al. 2011; Van den Broeck 2011; Van den Broeck 2013), and the other based on first-order DPLL search (Gogate and Domingos 2011). Approximate

algorithms were also proposed, including lifted importance sampling (Gogate and Domingos 2011; Gogate, Jha, and Venugopal 2012). More generally, there is a large literature on exact and approximate *lifted probabilistic inference* in statistical relational models, which can be adapted to solve certain WFOMC tasks. See Kersting (2012) for an overview.

Normal Forms

It is common for logical reasoning algorithms to operate on *normal form* representations instead of arbitrary sentences. For example, propositional SAT solvers and weighted model counters often expect CNF inputs. We distinguish the following first-order normal forms.

- A theory in *prenex normal form* consists of formulas $Q_1 x_1, \ldots, Q_n x_n,\ \phi$, where each Q_i is either a universal or existential quantifier, and ϕ is quantifier-free.
- A theory in *prenex clausal form* is a theory in prenex normal form where ϕ is a clause.
- A theory in *Skolem normal form* is a theory in prenex normal form where all Q_i are universal quantifiers.
- A *first-order CNF* is a theory in Skolem and prenex clausal form. Thus, all sentences take the form $\forall x_1, \ldots, \forall x_n,\ l_1 \vee \cdots \vee l_m$.

Existing WFOMC algorithms require a theory to be in *first-order CNF*. The same requirement is often posed by automated theorem provers, such as first-order resolution.

3 Skolemization for WFOMC

It is well known that one can take any arbitrary formula and convert it to prenex clausal form. This involves pushing negations inside, pushing quantifiers to the front, and distributing disjunctions over conjunctions. The situation for Skolem normal form is different.

Motivation

Not every formula can be transformed into an equivalent Skolem normal form. This problem is typically dealt with by *Skolemization*, which eliminates existential quantifiers from a prenex normal form. This is done by replacing existentially quantified variables by Skolem constants and functions. The result is not logically equivalent to the original formula, but only *equisatisfiable* (i.e., satisfiable precisely when the original formula is satisfiable).

The standard Skolemization algorithm is specific to the satisfiability task and may be unsuitable for other tasks. It is particularly unsuitable for WFOMC as it may produce a result with functions, which are not permitted in the WFOMC task. For example, standard Skolemization would transform the formula

$$\forall x, \exists y,\ \texttt{WorksFor}(x, y) \vee \texttt{Boss}(x) \tag{5}$$

into the following formula with the Skolem function $\mathsf{Sk}()$.

$$\forall x,\ \texttt{WorksFor}(x, \mathsf{Sk}(x)) \vee \texttt{Boss}(x).$$

As soon as we allow functions, the Herbrand base becomes infinite, which makes the model counting task ill-defined, therefore, ruling out standard Skolemization for WFOMC.[3]

Algorithm

This section introduces a Skolemization technique for WFOMC. It takes as input a triple $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ whose Δ is an arbitrary sentence and returns a triple $(\Delta', \mathrm{w}', \bar{\mathrm{w}}')$ whose Δ' is in Skolem normal form (i.e., no existential quantifiers). Such a Δ' can then be turned into first-order CNF using standard transformations. The proposed technique does not introduce functions. It satisfies two properties, one is essential and the other expands the applications of the technique.

The essential property is soundness.

Property 1 (Soundness). *Skolemization of* $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ *to* $(\Delta', \mathrm{w}', \bar{\mathrm{w}}')$ *is sound iff for any* $\mathbf{D}$*, we have that*

$$\mathrm{WFOMC}(\Delta, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}}) = \mathrm{WFOMC}(\Delta', \mathbf{D}, \mathrm{w}', \bar{\mathrm{w}}').$$

To motivate the second property, we note that one may be interested in queries of the form $\mathrm{WFOMC}(\Delta \wedge \phi, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}})$, where Δ, w and $\bar{\mathrm{w}}$ are fixed, but where ϕ is changing. For example, we will see in Section 4 that probabilistic inference can be reduced to these types of queries. Therefore, we want to achieve a stronger form of soundness.

Property 2 (Modularity). *Skolemization of* $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ *to* $(\Delta', \mathrm{w}', \bar{\mathrm{w}}')$ *is modular iff for any* $\mathbf{D}$ *and any sentence* ϕ*,*

$$\mathrm{WFOMC}(\Delta \wedge \phi, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}}) = \mathrm{WFOMC}(\Delta' \wedge \phi, \mathbf{D}, \mathrm{w}', \bar{\mathrm{w}}').$$

That is, by replacing ϕ, one does not invalidate the Skolemization obtained under a different ϕ.

The proposed Skolemization algorithm eliminates existential quantifiers one by one. Its basic building block is the following transformation.

Definition 3. Suppose that Δ contains a subexpression of the form $\exists x, \phi(x, \mathbf{y})$, where $\phi(x, \mathbf{y})$ is an arbitrary sentence containing the free logical variables x and $\mathbf{y}$. Let n be the number of variables in $\mathbf{y}$. First, we introduce two new predicates: the *Tseitin predicate* $\mathtt{Z}/n$ and the *Skolem predicate* $\mathtt{S}/n$. Second, we replace the expression $\exists x, \phi(x, \mathbf{y})$ in Δ by the atom $\mathtt{Z}(\mathbf{y})$, and append the formulas

$$\begin{aligned} \forall \mathbf{y}, \forall x,\ \mathtt{Z}(\mathbf{y}) \vee \neg\phi(x, \mathbf{y}) \\ \forall \mathbf{y},\ \mathtt{S}(\mathbf{y}) \vee \ \ \mathtt{Z}(\mathbf{y}) \\ \forall \mathbf{y}, \forall x,\ \mathtt{S}(\mathbf{y}) \vee \neg\phi(x, \mathbf{y}). \end{aligned}$$

The functions w' and $\bar{\mathrm{w}}'$ are equal to w and $\bar{\mathrm{w}}$, except that $\mathrm{w}'(\mathtt{Z}) = \bar{\mathrm{w}}'(\mathtt{Z}) = \mathrm{w}'(\mathtt{S}) = 1$ and $\bar{\mathrm{w}}'(\mathtt{S}) = -1$.

In the resulting theory Δ', a single existential quantifier is now eliminated. This building block can eliminate single universal quantifiers as well. When Δ contains a subexpression $\forall x, \phi(x, \mathbf{y})$, we replace it by $\neg\exists x, \neg\phi(x, \mathbf{y})$, whose existential quantifier can be eliminated with Definition 3.

We can now show the following.

[3] One could obtain a Skolem normal form by grounding existential quantifiers, replacing them by large, but finite disjunctions. While this may still permit limited runtime improvements on vacuous formulas, it is for all practical purposes equivalent to reducing the WFOMC problem to a WMC problem. Moreover, that transformation is dependent on the domain and leads to large formulas whose conversion to CNF blows up (e.g., when grounding $\exists x \forall y$).

Theorem 3 (Modularity). *Repeated application of Definition 3 comprises a modular Skolemization algorithm.*

The detailed proof can be found in the appendix.

Intuition

Our Skolemization algorithm implicitly tries to enforce an equivalence between the eliminated subexpression and the Tseitin predicate[4], which is explicitly written as

$$\forall \mathbf{y},\ \mathtt{Z}(\mathbf{y}) \Leftrightarrow [\exists x, \phi(x, \mathbf{y})]\,.$$

This equivalence contains an existential quantifier so it cannot be represented explicitly. Instead, the algorithms enforces a relaxed equivalence, represented by the three formulas in Definition 3. The intuition is that by relaxing the equivalence we introduce additional models to the theory, but for every additional model with weight W, there is exactly one additional model with weight $-W$.[5] The WFOMC therefore stays the same.

The interaction between the three relaxed formulas, the intended equivalence, and the model weights becomes more apparent after a case analysis on $\mathtt{Z}(\mathbf{y})$:

1. When $\mathtt{Z}(\mathbf{y})$ is false, it implies that $\exists x, \phi(x, \mathbf{y})$ is false, which is intended. It also implies that $\mathtt{S}(\mathbf{y})$ is true, which does not change the model count, since we multiply by 1.
2. When $\mathtt{Z}(\mathbf{y})$ is true, it implies that only three states of $\mathtt{S}(\mathbf{y})$ and $\exists x, \phi(x, \mathbf{y})$ are allowed:
   (a) $\exists x, \phi(x, \mathbf{y})$ is true and $\mathtt{S}(\mathbf{y})$ is true. This is again intended, because $\mathtt{Z}(\mathbf{y})$ and $\exists x, \phi(x, \mathbf{y})$ are equivalent.
   (b) $\exists x, \phi(x, \mathbf{y})$ is false and $\mathtt{S}(\mathbf{y})$ is true. This is an unintended state with a positive weight W.
   (c) $\exists x, \phi(x, \mathbf{y})$ is true and $\mathtt{S}(\mathbf{y})$ is false. This is an unintended state with a weight $-W$. The negative weight comes from the fact that $\bar{\mathrm{w}}(\mathtt{S}) = -1$.

The weights of the unintended models cancel each other out.

Examples

We will now illustrate our Skolemization algorithm on concrete examples. Suppose that Δ is Formula 5, that is,

$$\forall x, \exists y,\ \mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x).$$

We can apply Definition 3 to the subexpression $\exists y,\ \mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x)$, resulting in a Δ' equal to

$$\begin{aligned}
&\forall x,\ \mathtt{Z}(x)\\
\forall x, \forall y,\ &\mathtt{Z}(x) \vee \neg\left[\mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x)\right]\\
&\forall x,\ \mathtt{Z}(x) \vee \mathtt{S}(x)\\
\forall x, \forall y,\ &\mathtt{S}(x) \vee \neg\left[\mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x)\right].
\end{aligned}$$

The first formulas is the original formula with the subexpression substituted by $\mathtt{Z}(x)$.

To get a better insight into the result, we will simplify it using first-order unit propagation (Van den Broeck et al. 2011) while noting that the first formula is a unit clause. The simplified theory is

$$\begin{aligned}
\forall x, \forall y,\ &\mathtt{S}(x) \vee \neg\mathtt{WorksFor}(x, y)\\
\forall x,\ &\mathtt{S}(x) \vee \neg\mathtt{Boss}(x).
\end{aligned}$$

We verify the correctness of this Skolemization as follows.

- When $\mathtt{Boss}(x)$ is true, the formula is satisfied for x, and the models of Δ' are intended, that is, they correspond to models of Δ. Indeed, $\mathtt{S}(x)$ is entailed to be true and the model weights are multiplied by one.
- When $\mathtt{Boss}(x)$ is false and $\mathtt{WorksFor}(x, y)$ is true for at least one y, then $\mathtt{S}(x)$ is entailed to be true. Again these models are intended, because $\exists y,\ \mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x)$ is now satisfied for x. The model weights are multiplied by one.
- When $\mathtt{Boss}(x)$ is false and $\mathtt{WorksFor}(x, y)$ is false for all y then $\mathtt{S}(x)$ can be either true or false. This is where unintended models appear, once with $\mathtt{S}(x)$ true and once with $\mathtt{S}(x)$ false. Because they have opposing weights, the contributions of these unintended models cancel out.

As a second example, consider Δ to be

$$\forall x, \exists y, \exists \mathtt{z},\ \mathtt{Parents}(x, y, \mathtt{z}) \vee \mathtt{Adam}(x).$$

Skolemization of the inner existential quantifier results in

$$\begin{aligned}
&\forall x, \exists y,\ \mathtt{Z}_1(x, y)\\
\forall x, \forall y, \forall \mathtt{z},\ &\mathtt{Z}_1(x, y) \vee \neg\left[\mathtt{Parents}(x, y, \mathtt{z}) \vee \mathtt{Adam}(x)\right]\\
&\forall x, \forall y,\ \mathtt{Z}_1(x, y) \vee \mathtt{S}_1(x, y)\\
\forall x, \forall y, \forall \mathtt{z},\ &\mathtt{S}_1(x, y) \vee \neg\left[\mathtt{Parents}(x, y, \mathtt{z}) \vee \mathtt{Adam}(x)\right]
\end{aligned}$$

This example shows the need for a Tseitin predicate $\mathtt{Z}_1$. The first sentence still contains an existential quantifier. One more elimination and unit propagation step replaces that sentence by $\forall y, \forall x,\ \mathtt{S}_2(\mathbf{y}) \vee \neg\mathtt{Z}_1(x, y)$ and the result is in Skolem normal form.

Properties

Theorem 3 suggests the repeated application of Definition 3, as long as the sentence contains an existential quantifier, or a universal quantifier not in prenex form. This approach has one caveat: eliminating $\exists x, \phi(x, \mathbf{y})$ adds the expression $\neg\phi(x, \mathbf{y})$ to Δ'. When we eliminate quantifiers from left to right, $\phi(x, \mathbf{y})$ itself can contain quantifiers. This operation will introduce new quantifiers in $\neg\phi(x, \mathbf{y})$ and cause a blow up due to the duplication in the newly added formulas. This can be avoided by eliminating from right to left, that is, from innermost to outermost. We can show the following theorem, whose proof is in the appendix.

Theorem 4 (Termination and Complexity). *Repeated application of Definition 3 will terminate with a sentence in Skolem normal form. Moreover, this can be achieved in time polynomial in the size of* Δ.

The resulting Skolem normal form sentence can subsequently be transformed into first-order CNF. When using Tseitin's transformation (Tseitin 1983), this can even be done in polynomial time.

[4]This equivalence represents a set of propositional Tseitin encodings, in which each $\mathtt{Z}$ atom is a Tseitin variable (Tseitin 1983).

[5]This is not dissimilar to the inclusion-exclusion principle.

In our first example, the Tseitin predicate $\mathtt{Z}$ could be removed from Δ' by unit propagation. The following proposition generalizes that observation.

Proposition 5. *Suppose that we are eliminating a subexpression $\exists x, \phi(x, \mathbf{y})$ from a sentence $\forall \mathbf{y}, \exists \mathbf{x}, \phi(x, \mathbf{y})$ using the procedure of Definition 3. That is, the existential quantifier in this subexpression is only preceded by universal quantifiers. Then, we can avoid adding Tseitin predicate $\mathtt{Z}$ and instead define Δ' to be*

$$\forall \mathbf{y}, \forall x,\ \mathtt{S}(\mathbf{y}) \vee \neg\phi(x, \mathbf{y}).$$

This simplifies the transformation when applicable, in particular when Δ is already in prenex normal form.

4 WFOMC Encodings

We will show in this section how the proposed Skolemization technique can extend the scope of first-order model counters to new situations. We will consider in particular one undirected first-order probabilistic language (Markov Logic) and one directed language (Probabilistic Logic Programs). First-order model counters currently apply to a subset of the first representation, and not to the second representation. With Skolemization, these model counters can now be applied to both. Our treatment is based on providing WFOMC encodings of these representations, to which our Skolemization technique is then applied.[6]

Consider a first-order probabilistic model that induces the distribution $\Pr_{\mathbf{D}}(.)$ for domain $\mathbf{D}$. A *WFOMC encoding* of this model is a triple $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ which guarantees that for any sentence ϕ (usually a conjunction of literals) and domain $\mathbf{D}$, we have that

$$\Pr_{\mathbf{D}}(\phi) = \frac{\mathrm{WFOMC}(\Delta \wedge \phi, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}})}{\mathrm{WFOMC}(\Delta, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}})}.$$

Markov Logic Networks

We will now introduce a WFOMC encoding for *Markov logic networks* (MLN) (Richardson and Domingos 2006).

Representation An MLN is a set of tuples (w, ψ), where w is a real number representing a weight and ψ is a formula in first-order logic. When w is infinite, ψ represents a first-order logic constraint, also called a *hard formula*.

Building further on the example given before, consider the following MLN

$$1.3 \quad \exists y,\ \mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x). \tag{6}$$

This statement softens the logical sentence we saw earlier. Instead of saying that every person either has a boss, or is a boss, it states that worlds with many employed people are more likely. That is, it is now possible to have a world with unemployed people, but the more unemployed people there are, the lower the probability of that world.

The semantics of a first-order MLN Φ is defined in terms of its *grounding* for a given domain of constants $\mathbf{D}$. The grounding of Φ is the MLN obtained by first grounding all its quantifiers and then replacing each formula in Φ with all its groundings (using the same weight). With the domain $\mathbf{D} = \{\mathsf{A}, \mathsf{B}\}$ (e.g., two people, Alice and Bob), the above first-order MLN represents the following grounding.

$$1.3 \quad \mathtt{WorksFor}(\mathsf{A}, \mathsf{A}) \vee \mathtt{WorksFor}(\mathsf{A}, \mathsf{B}) \vee \mathtt{Boss}(\mathsf{A})$$
$$1.3 \quad \mathtt{WorksFor}(\mathsf{B}, \mathsf{A}) \vee \mathtt{WorksFor}(\mathsf{B}, \mathsf{B}) \vee \mathtt{Boss}(\mathsf{B})$$

This ground MLN contains six different random variables, which correspond to all groundings of atoms $\mathtt{WorksFor}(x, y)$ and $\mathtt{Boss}(x)$. This leads to a distribution over 2^6 possible worlds (i.e., interpretation). The weight of each world is simply the product of all weights e^w, where (w, γ) is a ground MLN formula and γ is satisfied by the world. The weights of worlds that do not satisfy a hard formula are set to zero. The probabilities of worlds are obtained by normalizing their weights.

Encoding a Markov Logic Network

Definition 4. The WFOMC encoding $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ of an MLN is constructed as follows. For each MLN formula $(w_i, \phi_i(\mathbf{x}_i))$, where $\mathbf{x}_i$ denotes the free logical variables in ϕ_i, we introduce a parameter predicate $\mathtt{P_i}/|\mathbf{x}_i|$. For each MLN formula, Δ contains the sentence $\forall \mathbf{x}_i,\ \mathtt{P_i}(\mathbf{x}_i) \Leftrightarrow \phi_i(\mathbf{x}_i)$. The weight function sets $\mathrm{w}(\mathtt{P_i}) = e^{w_i}$, $\bar{\mathrm{w}}(\mathtt{P_i}) = 1$, and $\mathrm{w}(\mathtt{Q}) = \bar{\mathrm{w}}(\mathtt{Q}) = 1$ for all other predicates $\mathtt{Q}$.

Each $\mathtt{P_i}$ captures the truth value of ϕ_i and carries its weight. Hard formulas can directly be encoded as constraints.

The encoding of Formula 6 has Δ equal to

$$\forall x,\ \mathtt{P}(x) \Leftrightarrow \exists y,\ \mathtt{WorksFor}(x, y) \vee \mathtt{Boss}(x).$$

Its w maps $\mathtt{P}$ to $e^{1.3}$ and all other predicates to 1. Its $\bar{\mathrm{w}}$ maps all predicates to 1.

As discussed in Section 2, WFOMC algorithms require first-order CNF input. Definition 4 will only yield a Δ in Skolem normal form (and thus rewritable into CNF) if the MLN formulas are quantifier-free. Then, the only quantifiers in Δ are the universal ones introduced by the encoding itself. Therefore, Van den Broeck et al. (2011) and Gogate and Domingos (2011) resort to *grounding all quantifiers* in the MLN formulas so as to obtain a CNF. This makes the WFOMC encoding specific to the domain $\mathbf{D}$, and partly removes first-order structure from the problem.

Our discussion is based on Van den Broeck et al. (2011). It is similar to the encoding of Gogate and Domingos (2011), whose parameter predicates have more arguments. While these encodings are specific to MLNs, it is straightforward to generalize them to other undirected languages, such as *parfactor graphs* (Poole 2003).

Applying Skolemization We can now perform WFOMC inference in MLNs with quantifiers. Skolemization and CNF

[6] These encodings are implemented in the WFOMC system: http://dtai.cs.kuleuven.be/wfomc

conversion for the example above results in a Δ' equal to

$$\begin{aligned}
&\forall x,\ \mathtt{P}(x) \vee \neg \mathtt{Z}(x)\\
&\forall x,\ \neg \mathtt{P}(x) \vee \mathtt{Z}(x)\\
\forall x, \forall y,\ &\mathtt{S}(x) \vee \neg \mathtt{WorksFor}(y, x)\\
&\forall x,\ \mathtt{S}(x) \vee \neg \mathtt{Boss}(x)\\
&\forall x,\ \mathtt{S}(x) \vee \mathtt{Z}(x)\\
\forall x, \forall y,\ &\mathtt{Z}(x) \vee \neg \mathtt{WorksFor}(x, y)\\
\forall x, \forall y,\ &\mathtt{Z}(x) \vee \neg \mathtt{Boss}(x)
\end{aligned}$$

This theory can be used for WFOMC inference.

Probabilistic Logic Programs

We now show a WFOMC encoding for a directed first-order probabilistic language. The encoding is explained for the *ProbLog* language (De Raedt, Kimmig, and Toivonen 2007; Fierens et al. 2013).

ProbLog Representation ProbLog extends logic programs with facts that are annotated with probabilities. A ProbLog program Φ is a set of probabilistic facts F and a regular logic program L. A probabilistic fact $p :: a$ consists of a probability p and an atom a. A logic program is a set of rules, with the form `Head :- Body`, where the head is an atom and the body is a conjunction of literals. For example,

$$\begin{aligned}
&0.1 :: \mathtt{Attends}(x).\\
&0.3 :: \mathtt{ToSeries}(x).\\
&\mathtt{Series} \text{ :- } \mathtt{Attends}(x), \mathtt{ToSeries}(x).
\end{aligned}$$

This program expresses that if more people attend a workshop, it more likely turns into a series of workshops.

The semantics of a ProbLog program Φ are defined by a distribution over the *groundings* of the probabilistic facts for a given domain of constants $\mathbf{D}$ (Sato 1995).[7] The probabilistic facts $p_i :: a_i$ induce a set of possible worlds, one for each possible partition of a_i in positive and negative literals. The set of true a_i literals with the logic program L define a well-founded model (Van Gelder, Ross, and Schlipf 1991). The probability of such a model is the product of p_i for all true a_i literals and $1 - p_i$ for all false a_i literals.

For the domain $\mathbf{D} = \{A, B\}$ (two people), the above first-order ProbLog program represents the following grounding:

$$\begin{aligned}
&0.1 :: \mathtt{Attends}(A).\\
&0.1 :: \mathtt{Attends}(B).\\
&0.3 :: \mathtt{ToSeries}(A).\\
&0.3 :: \mathtt{ToSeries}(B).\\
&\mathtt{Series} \text{ :- } \mathtt{Attends}(A), \mathtt{ToSeries}(A).\\
&\mathtt{Series} \text{ :- } \mathtt{Attends}(B), \mathtt{ToSeries}(B).
\end{aligned}$$

This ground ProbLog program contains 4 probabilistic facts which corresponds to 2^4 possible worlds. The weight of, for example, the world in which $\mathtt{Attends}(A)$ and $\mathtt{ToSeries}(A)$ are true would be $0.1 \cdot (1 - 0.1) \cdot 0.3 \cdot (1 - 0.3) = 0.0189$ and the model would be $\{\mathtt{Attends}(A), \mathtt{ToSeries}(A), \mathtt{Series}\}$.

Encoding a ProbLog Program The transformation from a ProbLog program to a first-order logic theory is based on Clark's completion (Clark 1978). This is a transformation from logic programs to first-order logic. For certain classes of programs, called *tight* logic programs (Fages 1994), it is correct, in the sense that every model of the logic program is a model of the completion, and vice versa. Intuitively, for each predicate `P`, the completion contains a single sentence encoding all its rules. These rules have the form $\mathtt{P}(\mathbf{x}) \text{ :- } b_i(\mathbf{x}, \mathbf{y}_i)$, where b_i is a body and $\mathbf{y}_i$ are the variables that appear in the body b_i but not in the head. The sentence encoding these rules in the completion is $\forall \mathbf{x},\ \mathtt{P}(\mathbf{x}) \Leftrightarrow \bigvee_i \exists \mathbf{y}_i,\ b_i(\mathbf{x}, \mathbf{y}_i)$. If the program contains cyclic rules, the completion is not sound, and, it is necessary to first apply a conversion to remove positive loops (Janhunen 2004).

Definition 5. The WFOMC encoding $(\Delta, \mathrm{w}, \bar{\mathrm{w}})$ of a tight ProbLog program has Δ equal to Clark's completion of L. For each probabilistic fact[8] $p :: a$ we set the weight function to $\mathrm{w}(\mathrm{pred}(a)) = p$ and $\bar{\mathrm{w}}(\mathrm{pred}(a)) = 1 - p$.

Again, a Skolem normal form is required to use WFOMC. However, we get this form only when the variables that appear in the body of a rule also appear in the head of a rule. This is not the case for most Prolog programs though. For example, if we apply Definition 5 to the example above, an existential quantifier appears in the sentence:

$$\mathtt{Series} \Leftrightarrow \exists x, \mathtt{Attends}(x) \wedge \mathtt{ToSeries}(x).$$

Furthermore, w maps `Attends` to 0.1 and `ToSeries` to 0.3, and $\bar{\mathrm{w}}$ maps `Attends` to 0.9 and `ToSeries` to 0.7. Both w and $\bar{\mathrm{w}}$ are 1 for all other predicates. This example is not in Skolem normal form and requires Skolemization before it can be processed by WFOMC algorithms.

Applying Skolemization Skolemization followed by CNF conversion gives a Δ' equal to

$$\begin{aligned}
&\mathtt{Series} \vee \neg \mathtt{Z}\\
&\neg \mathtt{Series} \vee \mathtt{Z}\\
\forall x,\ &\mathtt{Z} \vee \neg \mathtt{Attends}(x) \vee \neg \mathtt{ToSeries}(x)\\
&\mathtt{Z} \vee \mathtt{S}\\
\forall x,\ &\mathtt{S} \vee \neg \mathtt{Attends}(x) \vee \neg \mathtt{ToSeries}(x)
\end{aligned}$$

Sentence Δ' is in Skolem normal form and is now processable by WFOMC algorithms.

A simple ProbLog program as the one above is identical to a noisy-or structure (Cozman 2004), popular in Bayesian network modeling. Skolemization thus offers a fundamental method to lift first-order, directed structures, such as the noisy-or, in a generic manner (see also Section 6).

[7] Our treatment assumes a function-free and finite-domain fragment of ProbLog. Starting from classical ProbLog semantics, one can obtain the a finite function-free domain for a given query by exhaustively executing the Prolog program and keeping track of the goals that are called during resolution.

[8] If multiple probabilistic facts are defined for the same predicate, auxiliary predicates need to be introduced.

5 Liftability Implications

In our motivation for introducing first-order model counting, we touched upon the runtime and complexity improvements that can be attained by first-order counting. These complexity improvements have inspired a particular notion of lifted inference, called *domain-lifted* inference, which says that a WFOMC algorithm is lifted when it runs in time polynomial in the size of $\mathbf{D}$ (Van den Broeck 2011).

While this notion of lifted inference may not capture everyone's perception of lifting, it does provide a clear formal framework. In particular, we can now talk about classes of sentences Δ for which an algorithm is domain-lifted. We say that the algorithm is *complete* for those classes. We can also talk about classes of sentences Δ for which there exists, or cannot exist a domain-lifted algorithm. We call the former classes *liftable* (Jaeger and Van den Broeck 2012).

All existing completeness and liftability theorems require that Δ is in first-order CNF. This requirement carries over from the existing WFOMC algorithms. Given our Skolemization algorithm, we can now restate these theorems to apply more generally. For example, the positive liftability result of Van den Broeck (2011) becomes the following

Corollary 6. *Suppose that Δ is a theory of sentences with up to two logical variables, and otherwise arbitrary structure. The complexity of computing the WFOMC of Δ is polynomial in the size of $\mathbf{D}$. That is, this class is domain-liftable.*

Other notions of liftability also include queries ϕ in the complexity analysis, since they are important for lifted probabilistic inference. Based on Van den Broeck and Davis (2012), and Van den Broeck and Darwiche (2013), we can now claim the following.

Corollary 7. *Suppose that Δ is a theory of sentences with up to two logical variables, and otherwise arbitrary structure. The complexity of computing the WFOMC of $\Delta \wedge \phi$ is polynomial in the size of $\mathbf{D}$ and ϕ, provided that ϕ is a conjunction of only unary literals, and binary literals of bounded Boolean rank.*

These WFOMC liftability theorems have direct implications for all languages with a WFOMC encoding. For example, we can now say that MLNs with up to two logical variables per formula are domain-liftable, regardless of the quantifiers used. Previously, this was only true for quantifier-free MLNs. We can now also show that ProbLog programs with up to two logical variables per clause are guaranteed to be liftable. This is the first such liftability result for probabilistic logic programs.

6 Related Work

In the encodings for MLNs and probabilistic logics, the weight functions (indirectly) represent probabilities and are therefore always positive. Our Skolemization algorithm introduces negative weights. This might appear odd when interpreting the weights as negative probabilities. This issue has been discussed before. For example, Feynman (1987) writes *"Negative probabilities allow an abstract calculation which permits freedom to do mathematical calculations in any order simplifying the analysis enormously"*.

The potential of negative probabilities was already observed by Jha and Suciu (2012) for answering queries in probabilistic databases and served as inspiration for our approach. Probabilistic databases (Suciu et al. 2011) are fundamentally a type of first-order probabilistic model. It can be viewed as a special type of weighted model counting problem (Δ, w), where the weight function encodes the probability $w(t)$ with which a tuple t can be found in the database. A query on such a database is typically a union of conjunctive queries (UCQ), which corresponds to a monotone DNF sentence Δ. A noticeable difference with most WMC solvers (and WFOMC) is that the solvers for probabilistic databases expect the theory Δ to be in DNF instead of CNF. Different from WFOMC is that although the query (i.e., Δ) is first-order, the weight function is defined on the propositional level like in WMC. Weights are thus assigned to ground literals (the tuples) whereas for WFOMC weights are assigned to predicates (the tables). This allows WFOMC to exploit more types of symmetries.

Jha and Suciu (2012) propose to extend probabilistic databases with MarkoViews, a representation similar to MLNs, in which each weighted formula is again a UCQ query, that is, a monotone DNF. To compute the probability of a query, they introduce negative tuple probabilities.

The use of negative probabilities has also come up for optimizing calculations for specific structures in probabilistic graphical models like noisy-or (Díez and Galán 2003). This particular case has been translated to the first-order case by Kisynski and Poole (2011) and resulted in an approach to lift noisy-or structures. In Section 4 we showed how the application of Skolemization leads to lifting noisy-or and both methods turn out to output a similar encoding for this particular case. Therefore, the approach followed by Kisynski and Poole (2011) can be considered a special case of the Skolemization algorithm applied to a noisy-or model.

Jaeger (2012) shows a negative liftability proof that uses *relational Skolemization*. Similar to our approach, subexpressions containing an existential quantifier are transformed and relaxed to eliminate the quantifier. Relational Skolemization, however, does not guarantee a correct model count. It rather guarantees that if the weight of a model is non-zero it will also be non-zero in the Skolemized version.

7 Conclusions

In this paper, we introduced a Skolemization procedure that is sound for weighted first-order model counting. It extends the applicability of first-order model counters to encodings which require an existential quantifier such as Markov logic models with quantifiers and probabilistic logic programs. It also extends the class of first-order sentences whose models we can count efficiently.

Acknowledgments

This work was supported by ONR grant #N00014-12-1-0423, NSF grant #IIS-1118122, NSF grant #IIS-0916161, and the Research Foundation-Flanders (FWO-Vlaanderen). GVdB is also at KU Leuven, Belgium.

Appendix

A Proof of Theorem 3

We will now prove the sequence of steps that leads to the removal of an existential quantifier in Δ to obtain Δ', w' and $\bar{\mathrm{w}}'$ while maintaining modularity. To replace the expression $\exists x, \phi(x, \mathbf{y})$ we perform the following steps.

Isolate the Quantifier Introduce a new *Tseitin* predicate $\mathtt{Z}/n$, whose arity n is the number of $\mathbf{y}$ variables. Set $\mathrm{w}'(\mathtt{Z}) = \bar{\mathrm{w}}'(\mathtt{Z}) = 1$ and for all other predicates $\mathtt{P}$, set $\mathrm{w}'(\mathtt{P}) = \mathrm{w}(\mathtt{P})$ and $\bar{\mathrm{w}}'(\mathtt{P}) = \bar{\mathrm{w}}(\mathtt{P})$. Construct Δ' by replacing the expression $\exists x, \phi(x, \mathbf{y})$ in Δ by the atom $\mathtt{Z}(\mathbf{y})$, and appending the equivalence .

In any grounding of Δ, this step performs a Tseitin encoding of all groundings of $\exists x, \phi(x, \mathbf{y})$. The groundings of $\mathtt{Z}(\mathbf{y})$ play the role of Tseitin variables. This step therefore satisfies Property 2.

Split the Equivalence Rewrite equivalence $\forall \mathbf{y},\ \mathtt{Z}(\mathbf{y}) \Leftrightarrow \exists x, \phi(x, \mathbf{y})$ as two implications, $\forall \mathbf{y},\ \mathtt{Z}(\mathbf{y}) \Rightarrow \exists x, \phi(x, \mathbf{y})$ and $\forall \mathbf{y},\ \mathtt{Z}(\mathbf{y}) \Leftarrow \exists x, \phi(x, \mathbf{y})$. In clausal form, these become

$$\forall \mathbf{y}, \exists x,\ \neg \mathtt{Z}(\mathbf{y}) \vee \quad \phi(x, \mathbf{y})$$
$$\forall \mathbf{y}, \forall x,\quad \mathtt{Z}(\mathbf{y}) \vee \neg \phi(x, \mathbf{y}).$$

This step satisfies Property 2 because it is a logical equivalence.

Convert to a Feature Introduce a new *Skolem predicate* predicate $\mathtt{S}/n$. Set $\mathrm{w}(\mathtt{S}) = 1$ and $\bar{\mathrm{w}}(\mathtt{S}) = 0$ and replace the sentence $\forall \mathbf{y}, \exists x,\ \neg \mathtt{Z}(\mathbf{y}) \vee \phi(x, \mathbf{y})$ by

$$\forall \mathbf{y},\ \mathtt{S}(\mathbf{y}) \Leftrightarrow \exists x, \neg \mathtt{Z}(\mathbf{y}) \vee \phi(x, \mathbf{y}).$$

In all models of the resulting theory where $\forall \mathbf{y}, \exists x,\ \neg \mathtt{Z}(\mathbf{y}) \vee \phi(x, \mathbf{y})$ is not satisfied, there will exist a $\mathbf{y}$ for which $\exists x, \neg \mathtt{Z}(\mathbf{y}) \vee \phi(x, \mathbf{y})$ is not satisfied. This will cause at least one $\mathtt{S}(\mathbf{y})$ atom to be false in those models, which means that the weight of those models is multiplied by 0. The weight of all other models remains the same. This step therefore satisfies Property 2.

Convert to an Implication Set $\bar{\mathrm{w}}(\mathtt{S}) = -1$ and turn the equivalence $\forall \mathbf{y},\ \mathtt{S}(\mathbf{y}) \Leftrightarrow \exists x, \neg \mathtt{Z}(\mathbf{y}) \vee \phi(x, \mathbf{y})$ into an implication $\forall \mathbf{y},\ \mathtt{S}(\mathbf{y}) \Leftarrow \exists x, \neg \mathtt{Z}(\mathbf{y}) \vee \phi(x, \mathbf{y})$, which in clausal form becomes

$$\forall \mathbf{y},\ \mathtt{S}(\mathbf{y}) \vee \quad \mathtt{Z}(\mathbf{y})$$
$$\forall \mathbf{y}, \forall x,\ \mathtt{S}(\mathbf{y}) \vee \neg \phi(x, \mathbf{y}).$$

Replacing the equivalence by an implication and changing $\bar{\mathrm{w}}(\mathtt{S})$ to -1 is correct for the following reason. Let $\mathtt{S}(\mathbf{y}) \Leftrightarrow \Sigma(\mathbf{y})$ be the above equivalence which is in Δ, and let Γ represent all other sentences in Δ (i.e., $\Delta \equiv (\Sigma(\mathbf{y}) \Leftrightarrow \mathtt{S}(\mathbf{y})) \wedge \Gamma$). Our goal is now to construct a triple $(\Delta', \mathrm{w}', \bar{\mathrm{w}}')$, where $\Delta' \equiv (\Sigma(\mathbf{y}) \Rightarrow \mathtt{S}(\mathbf{y})) \wedge \Gamma$, such that $\mathrm{WFOMC}(\Delta \wedge \phi, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}}) = \mathrm{WFOMC}(\Delta' \wedge \phi, \mathbf{D}, \mathrm{w}', \bar{\mathrm{w}}')$ for all domains $\mathbf{D}$ and all sentences ϕ.

Let $\Sigma(\mathtt{A})$ and $\mathtt{S}(\mathtt{A})$ be any arbitrary grounding of $\Sigma(\mathbf{y})$ and $\mathtt{S}(\mathbf{y})$. A case analysis on the values of $\Sigma(\mathtt{A})$ and $\mathtt{S}(\mathtt{A})$ shows that $\mathrm{WFOMC}(\Delta \wedge \phi, \mathbf{D}, \mathrm{w}, \bar{\mathrm{w}})$ and $\mathrm{WFOMC}(\Delta' \wedge \phi, \mathbf{D}, \mathrm{w}', \bar{\mathrm{w}}')$ consist of the following terms (for compactness we drop $\mathbf{D}$ from the notation since it doesn't change).

$\Sigma(\mathbf{A})$	$\mathtt{S}(\mathbf{A})$	$\mathrm{WFOMC}(\Delta \wedge \phi, \mathrm{w}, \bar{\mathrm{w}})$
1	1	$\mathrm{w}(\mathtt{S}) \cdot \mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}, \bar{\mathrm{w}})$
1	0	0
0	1	0
0	0	$\bar{\mathrm{w}}(\mathtt{S}) \cdot \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}, \bar{\mathrm{w}})$

$\Sigma(\mathbf{A})$	$\mathtt{S}(\mathbf{A})$	$\mathrm{WFOMC}(\Delta' \wedge \phi, \mathrm{w}', \bar{\mathrm{w}}')$
1	1	$\mathrm{w}'(\mathtt{S}) \cdot \mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}', \bar{\mathrm{w}}')$
1	0	0
0	1	$\mathrm{w}'(\mathtt{S}) \cdot \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}', \bar{\mathrm{w}}')$
0	0	$\bar{\mathrm{w}}'(\mathtt{S}) \cdot \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}', \bar{\mathrm{w}}')$

Note that $\bar{\mathrm{w}}(\mathtt{S}) = 0$ in the encoding of Δ, and that thus

$$\begin{aligned}
&\mathrm{WFOMC}(\Delta \wedge \phi, \mathrm{w}, \bar{\mathrm{w}}) = \\
&\quad \mathrm{w}(\mathtt{S}) \cdot \mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}, \bar{\mathrm{w}}) \\
&\quad + 0 \cdot \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}, \bar{\mathrm{w}}) \\
&\mathrm{WFOMC}(\Delta' \wedge \phi, \mathrm{w}', \bar{\mathrm{w}}') = \\
&\quad \mathrm{w}'(\mathtt{S}) \cdot \mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}', \bar{\mathrm{w}}') \\
&\quad + [\mathrm{w}'(\mathtt{S}) + \bar{\mathrm{w}}'(\mathtt{S})] \\
&\qquad \cdot \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}', \bar{\mathrm{w}}')
\end{aligned}$$

Setting $\mathrm{w}'(\mathtt{P}) = \mathrm{w}(\mathtt{P})$ for all predicates $\mathtt{P}$ except for $\mathtt{S}$ ensures that $\mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}, \bar{\mathrm{w}}) = \mathrm{WFOMC}(\Gamma \wedge \Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}', \bar{\mathrm{w}}')$, that $\mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}, \bar{\mathrm{w}}) = \mathrm{WFOMC}(\Gamma \wedge \neg\Sigma(\mathbf{A}) \wedge \phi, \mathrm{w}', \bar{\mathrm{w}}')$. Furthermore, set $\mathrm{w}(\mathtt{S}) = \mathrm{w}'(\mathtt{S})$. What remains for $\mathrm{WFOMC}(\Delta \wedge \phi, \mathrm{w}, \bar{\mathrm{w}})$ to equal $\mathrm{WFOMC}(\Delta' \wedge \phi, \mathrm{w}', \bar{\mathrm{w}}')$ is that $\mathrm{w}'(\mathtt{S}) + \bar{\mathrm{w}}'(\mathtt{S}) = \mathrm{w}(\mathtt{S}) + \bar{\mathrm{w}}'(\mathtt{S}) = 0$, which is achieved by setting $\bar{\mathrm{w}}'(\mathtt{S}) = -\mathrm{w}(\mathtt{S}) = -1$.

B Proof of Theorem 4

Proof. We begin by proving termination. Let the *internal quantifier count* of a sentence be the number of quantifiers it contains, excluding the leading universal quantifiers. Suppose that a sentence has an internal quantifier count of m. We can select any subexpression that starts with a quantifier and apply Skolemization to it (potentially converting $\forall$ into $\exists$ first). This reduces the internal quantifier count of Δ to be at most $m - 1$ because at least one quantifier is removed. New sentences are added, however, containing the Tseitin and Skolem predicates, and expressions $\neg\phi(x, \mathbf{y})$. These sentences also have an internal quantifier count of at most $m-1$. Suppose that the sentences in Δ have an internal quantifier count of at most m_{max}. Applying Skolemization to one quantifier in each sentence reduces the maximal internal quantifier count to at most $m_{max} - 1$. Therefore, by repeating this procedure for a finite number of steps, we obtain a theory with an internal quantifier count of zero, which is in Skolem normal form.

Next, we prove polynomial complexity. We can remove the quantifiers in a sentence Δ one by one, starting from the innermost quantifier. The removed subexpression $\phi(x, \mathbf{y})$ does not contain any quantifiers, so the internal quantifier count of the added formulas is zero. They are in Skolem normal form. The innermost subexpression is replaced by a Tseitin predicate, reducing the internal quantifier count by

one. The number of required elimination steps before the entire sentence is in Skolem normal form is thus equal to the number of quantifiers in Δ. Moreover, the number of added formulas, and their size, is polynomial in the size of Δ. □

References

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific independence in Bayesian networks. In *Proceedings of UAI*, 115–123. Morgan Kaufmann Publishers Inc.

Chavira, M., and Darwiche, A. 2005. Compiling Bayesian networks with local structure. In *Proceedings of IJCAI*, volume 19, 1306.

Chavira, M., and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artificial Intelligence* 172(6-7):772–799.

Chavira, M.; Darwiche, A.; and Jaeger, M. 2006. Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning* 42(1-2):4–20.

Choi, A.; Kisa, D.; and Darwiche, A. 2013. Compiling probabilistic graphical models using sentential decision diagrams. In *Proceedings of the 12th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*.

Clark, K. 1978. Negation as failure. In *Readings in nonmonotonic reasoning*, 311–325. Morgan Kaufmann Publishers.

Cozman, F. G. 2004. Axiomatizing noisy-or. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, 979–980.

Darwiche, A. 2002. A logical approach to factoring belief networks. *Proceedings of KR* 409–420.

De Raedt, L.; Frasconi, P.; Kersting, K.; and Muggleton, S., eds. 2008. *Probabilistic inductive logic programming: theory and applications*. Berlin, Heidelberg: Springer-Verlag.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of IJCAI*, volume 7, 2462–2467.

Díez, F. J., and Galán, S. F. 2003. Efficient computation for the noisy max. *International Journal of Intelligent Systems* 18(2):165–177.

Fages, F. 1994. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science* 1:51–60.

Feynman, R. P. 1987. Negative probability. *Quantum implications: essays in honour of David Bohm* 235–248.

Fierens, D.; Van den Broeck, G.; Thon, I.; Gutmann, B.; and De Raedt, L. 2011. Inference in probabilistic logic programs using weighted CNF's. In *Proceedings of UAI*, 211–220.

Fierens, D.; Van den Broeck, G.; Renkens, J.; Shterionov, D.; Gutmann, B.; Thon, I.; Janssens, G.; and De Raedt, L. 2013. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming*.

Getoor, L., and Taskar, B., eds. 2007. *An Introduction to Statistical Relational Learning*. MIT Press.

Gogate, V., and Dechter, R. 2011. Samplesearch: Importance sampling in presence of determinism. *Artificial Intelligence* 175(2):694–729.

Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *Proceedings of UAI*, 256–265.

Gogate, V.; Jha, A. K.; and Venugopal, D. 2012. Advances in lifted importance sampling. In *Proceedings of AAAI*, 1910–1916.

Gomes, C. P.; Sabharwal, A.; and Selman, B. 2009. Model counting. *Handbook of Satisfiability* 185:633–654.

Hinrichs, T., and Genesereth, M. 2006. Herbrand logic. Technical Report LG-2006-02, Stanford University, Stanford, CA. http://logic.stanford.edu/reports/LG-2006-02.pdf.

Jaeger, M., and Van den Broeck, G. 2012. Liftability of probabilistic inference: Upper and lower bounds. In *Proceedings of StarAI*.

Jaeger, M. 2012. Lower complexity bounds for lifted inference. *arXiv preprint arXiv:1204.3255*.

Janhunen, T. 2004. Representing normal programs with clauses. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, volume 16, 358.

Jha, A., and Suciu, D. 2012. Probabilistic databases with MarkoViews. *Proceedings of the VLDB Endowment* 5(11):1160–1171.

Kersting, K. 2012. Lifted probabilistic inference. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*.

Kimmig, A.; Van den Broeck, G.; and De Raedt, L. 2012. Algebraic model counting. *arXiv preprint arXiv:1211.4475*.

Kisynski, J., and Poole, D. 2011. Lifted aggregation in directed first-order probabilistic models. In *Proceedings of IJCAI*, 1922–1929.

Poole, D. 2003. First-order probabilistic inference. In *Proceedings of IJCAI*, 985–991.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine learning* 62(1-2):107–136.

Sang, T.; Beame, P.; and Kautz, H. 2005. Solving Bayesian networks by weighted model counting. In *Proceedings of AAAI*, volume 1, 475–482.

Sato, T. 1995. A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP)*, 715–729.

Suciu, D.; Olteanu, D.; Ré, C.; and Koch, C. 2011. Probabilistic databases. *Synthesis Lectures on Data Management* 3(2):1–180.

Tseitin, G. S. 1983. On the complexity of derivation in propositional calculus. In *Automation of Reasoning*. Springer. 466–483.

Van den Broeck, G., and Darwiche, A. 2013. On the complexity and approximation of binary evidence in lifted inference. In *Advances in Neural Information Processing Systems 26 (NIPS)*.

Van den Broeck, G., and Davis, J. 2012. Conditioning in first-order knowledge compilation and lifted probabilistic inference. In *Proceedings of AAAI*.

Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted Probabilistic Inference by First-Order Knowledge Compilation. In *Proceedings of IJCAI*, 2178–2185.

Van den Broeck, G. 2011. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *Advances in Neural Information Processing Systems 24 (NIPS)*, 1386–1394.

Van den Broeck, G. 2013. *Lifted Inference and Learning in Statistical Relational Models*. Ph.D. Dissertation, KU Leuven.

Van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM (JACM)* 38(3):619–649.

Wei, W., and Selman, B. 2005. A new approach to model counting. In *Theory and Applications of Satisfiability Testing*, 96–97. Springer.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

On Egalitarian Belief Merging

Patricia Everaere
LIFL-CNRS, Université Lille 1, France
patricia.everaere@univ-lille1.fr

Sébastien Konieczny **Pierre Marquis**
CRIL-CNRS, Université d'Artois, France
{*konieczny,marquis*}*@cril.fr*

Abstract

Belief merging aims at defining the beliefs of a group of agents from the beliefs of each member of the group. It is related to more general notions of aggregation from economics (social choice theory). Two main subclasses of belief merging operators exist: majority operators which are related to utilitarianism, and arbitration operators which are related to egalitarianism. Though utilitarian (majority) operators have been extensively studied so far, there is much less work on egalitarian operators. In order to fill the gap, we investigate possible translations in a belief merging framework of some egalitarian properties and concepts coming from social choice theory, such as Sen-Hammond equity, Pigou-Dalton property, median, and Lorenz curves. We study how these properties interact with the standard rationality conditions considered in belief merging. Among other results, we show that the distance-based merging operators satisfying Sen-Hammond equity are mainly those for which $leximax$ is used as the aggregation function.

Introduction

The aim of belief merging is to define a coherent belief base from a set of jointly incoherent belief bases, representing the beliefs of a group of agents. The rationality properties of belief merging operators have been studied in (Revesz 1997; Lin and Mendelzon 1999; Konieczny and Pino Pérez 2002a; Konieczny, Lang, and Marquis 2004; Everaere, Konieczny, and Marquis 2010b). Especially, in (Konieczny and Pino Pérez 2002a), a number of postulates characterizing the so called IC merging operators have been identified. At the same time, many definitions of propositional belief merging operators have been pointed out. Most of these operators are distance-based ones, which means that they can be defined using a distance between interpretations and an aggregation function (Revesz 1997; Lin and Mendelzon 1999; Konieczny, Lang, and Marquis 2004; Everaere, Konieczny, and Marquis 2010a).

Two main subclasses of IC belief merging operators have been defined in (Konieczny and Pino Pérez 2002a): majority operators, which solve conflicts using majority, and arbitration operators, which try to find a consensual result. However, while many distance-based majority merging operators have been defined in the literature, very few arbitration operators have been identified so far. To be more precise, the only arbitration IC merging operators we are aware of are distance-based operators using $leximax$ as aggregation function.

As already discussed in previous merging papers, usual IC merging operators can be used to merge either beliefs or goals. This distinction between belief and goals do not seem to impact any of the usual IC postulates. It is natural to consider egalitarian merging operators when merging goals, if one tries to achieve a "fair" result. This could be important to ensure full adhesion of (autonomous) agents towards the obtained (merged) group goals. As to the belief merging issue, when the aim is to find the correct state of the world, majority methods may seem more appealing. In (Everaere, Konieczny, and Marquis 2010b) we discussed the truth-tracking problem for merging, and we made a distinction between two possible views of belief merging. The one relating to truth-tracking is called the epistemic view, and in this case it is natural to consider only majority operators (this is a consequence of Condorcet Jury Theorem (Everaere, Konieczny, and Marquis 2010b)). But there is a second possible view, called the synthesis view, where the aim is to best represent the opinion of the group (profile). This view has nothing to do with the correct state of the world, the merging process only cares about individual opinions. In this case egalitarian operators prove useful, since they provide in a sense a more robust (i.e. more consensual) view of the opinion of the group than majority operators.

Majority merging operators are closely related to the utilitarian social welfare approaches, where the aim is to determine solutions with the best aggregated utility (Harsanyi 1955; Moulin 1988; Sen 2005). On the other hand, arbitration operators are related to the egalitarian social welfare approaches, where the objective is to find solutions which are as fair as possible; this usually means that they give as much as possible to the poorest agents. To this extent, poverty measures (Rawls 1971; Gini 1921; Sen 1973; Dutta 2002) are relevant to the design of egalitarian approaches.

The aim of this paper is to introduce and study new egal-

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

itarian merging operators, by exhibiting other fairness conditions than arbitration and by pointing out belief merging operators satisfying them. The methodology we follow to reach our goal consists in investigating equity conditions considered in social choice theory (Arrow, Sen, and Suzumura 2002) in order to determine if they can be reasonably imported in the belief merging setting. Thus, in the following, we translate to the belief merging framework two egalitarian conditions coming from social choice theory: Sen-Hammond equity, and Pigou-Dalton property. We show that the distance-based merging operators satisfying Sen-Hammond equity are mainly those for which $leximax$ is used as the aggregation function. We also introduce two new families of belief merging operators, based respectively on the median and on an aggregated sum (Lorenz curves). We identify the rationality properties satisfied by these operators and study in particular their egalitarian behaviour.

The rest of the paper is organized as follows. First we give some preliminaries on propositional belief merging, focusing on IC merging operators and distance-based operators. The egalitarian operators we will define do not satisfy all the usual IC postulates. In fact the usual agregation postulates **(IC5)** and **(IC6)** are not satisfied. So we define a general family of belief merging operators, called pre-IC merging operators; where these Pareto-related conditions are replaced by unanimity conditions. Then we show how Sen-Hammond equity condition, and Pigou-Dalton property can be expressed in the belief merging setting. Then, we define the family of median distance-based merging operators. We show that the operators of this family based on the $leximed^k$ aggregation functions are pre-IC operators, and those for which $k \geq 0.5$ satisfy also the arbitration postulate **(Arb)**, but not the Pigou/Dalton property. Finally we introduce the family of cumulative sum distance-based merging operators; we identify in this family some pre-IC operators, and among them an operator satisfying the Pigou-Dalton property.

On Propositional Belief Merging

We consider a propositional language $\mathcal{L}$ defined from a finite set of propositional variables $\mathcal{P}$ and the usual connectives.

An interpretation (or state of the world) ω is a total function from $\mathcal{P}$ to $\{0,1\}$. Ω is the set of all interpretations. An interpretation is usually denoted by a bit vector whenever a strict total order on $\mathcal{P}$ is specified. An interpretation ω is a model of a formula $\phi \in \mathcal{L}$ if and only if it makes it true in the usual truth functional way. $\models$ and $\equiv$ denote logical entailment and equivalence, respectively. $[\phi]$ denotes the set of models of a formula ϕ, i.e., $[\phi] = \{\omega \in \Omega \mid \omega \models \phi\}$.

A *base* K denotes the set of beliefs of an agent, it is a finite set of propositional formulae, interpreted conjunctively (i.e., viewed as the conjunction of its elements).

A *profile* E denotes a group of n agents that are involved in the merging process; formally E is given by a muti-set $\{K_1, \ldots, K_n\}$ of bases. $\bigwedge E$ denotes the conjunction of all elements of E, and $\sqcup$ denotes the multi-set union. Two multi-sets $E = \{K_1, \ldots, K_n\}$ and $E' = \{K'_1, \ldots, K'_n\}$ are equivalent, noted $E \equiv E'$, iff there exists a permutation π over $\{1, \ldots, n\}$ s.t. for each $i \in 1, \ldots, n$, we have $K_i \equiv K'_{\pi(i)}$.

An *integrity constraint* μ is a formula restricting the possible results of the merging process.

A *merging operator* $\triangle$ is a function which associates with a profile E and an integrity constraint μ a *merged* base $\triangle_\mu(E)$.

The logical properties given in (Konieczny and Pino Pérez 2002a) for characterizing IC belief merging operators are:

Definition 1 *A merging operator $\triangle$ is an* IC merging operator *iff it satisfies the following properties:*

(IC0) $\triangle_\mu(E) \models \mu$
(IC1) *If* $\mu \not\models \bot$ *, then* $\triangle_\mu(E) \not\models \bot$
(IC2) *If* $\bigwedge E \wedge \mu \not\models \bot$, *then* $\triangle_\mu(E) \equiv \bigwedge E \wedge \mu$
(IC3) *If* $E_1 \equiv E_2$ *and* $\mu_1 \equiv \mu_2$, *then* $\triangle_{\mu_1}(E_1) \equiv \triangle_{\mu_2}(E_2)$
(IC4) *If* $K_1 \models \mu$ *and* $K_2 \models \mu$, *then* $\triangle_\mu(\{K_1, K_2\}) \wedge K_1 \not\models \bot$ *if and only if* $\triangle_\mu(\{K_1, K_2\}) \wedge K_2 \not\models \bot$
(IC5) $\triangle_\mu(E_1) \wedge \triangle_\mu(E_2) \models \triangle_\mu(E_1 \sqcup E_2)$
(IC6) *If* $\triangle_\mu(E_1) \wedge \triangle_\mu(E_2) \not\models \bot$, *then* $\triangle_\mu(E_1 \sqcup E_2) \models \triangle_\mu(E_1) \wedge \triangle_\mu(E_2)$
(IC7) $\triangle_{\mu_1}(E) \wedge \mu_2 \models \triangle_{\mu_1 \wedge \mu_2}(E)$
(IC8) *If* $\triangle_{\mu_1}(E) \wedge \mu_2 \not\models \bot$, *then* $\triangle_{\mu_1 \wedge \mu_2}(E) \models \triangle_{\mu_1}(E)$

See (Konieczny and Pino Pérez 2002a) for explanations on these properties. Two subclasses of IC merging operators are also defined in (Konieczny and Pino Pérez 2002a):

Definition 2 *An* IC majority operator *is an IC merging operator which satisfies the following majority property:*

(Maj) $\exists n \ \triangle_\mu (E_1 \sqcup \underbrace{E_2 \sqcup \ldots \sqcup E_2}_{n}) \models \triangle_\mu(E_2)$

An IC arbitration operator *is an IC merging operator which satisfies the following arbitration property:*[1]

(Arb) *If* $\left| \begin{array}{l} \triangle_{\mu_1}(K_1) \equiv \triangle_{\mu_2}(K_2) \\ \triangle_{\mu_1 \leftrightarrow \neg\mu_2}(\{K_1, K_2\}) \equiv (\mu_1 \leftrightarrow \neg\mu_2) \\ \mu_1 \not\models \mu_2 \\ \mu_2 \not\models \mu_1 \end{array} \right.$
then $\triangle_{\mu_1 \vee \mu_2}(\{K_1, K_2\}) \equiv \triangle_{\mu_1}(K_1)$

Majority operators solve conflicts using majority. **(Maj)** says that if one duplicates sufficiently many times a profile E_2, then the result of the merging of E_1 with the E_2 duplications will obey the choices of the profile E_2. Arbitration operators try to find a consensual result. See the corresponding condition 8 in Definition 3 and Example 1 which illustrates how this property gives a preference to consensual ("median") choices of interpretations.

The representation theorems enable to interpret these logical properties as constraints on the choice of interpretations for defining the models of the resulting base:

Definition 3 *A* syncretic assignment *is a function mapping each profile E to a total pre-order $\leq_E$[2] over Ω such that for*

[1] When $E = \{K\}$ we note $\triangle_\mu(K)$ instead of $\triangle_\mu(\{K\})$.

[2] For every pre-order $\leq$, $<$ denotes its strict part and $\simeq$ the corresponding indifference relation. Furthermore, we will use $\leq_K$ as a short for $\leq_{\{K\}}$, and $\leq_\omega$ as a short for any $\leq_{\{K_\omega\}}$ where ω is the unique model of K_ω.

any profiles E, E_1, E_2 and for any bases K, K' the following conditions hold:

1. *If $\omega \models \bigwedge E$ and $\omega' \models \bigwedge E$, then $\omega \simeq_E \omega'$*
2. *If $\omega \models \bigwedge E$ and $\omega' \not\models \bigwedge E$, then $\omega <_E \omega'$*
3. *If $E_1 \equiv E_2$, then $\leq_{E_1} = \leq_{E_2}$*
4. *$\forall \omega \models K \; \exists \omega' \models K' \; \omega' \leq_{\{K,K'\}} \omega$*
5. *If $\omega \leq_{E_1} \omega'$ and $\omega \leq_{E_2} \omega'$, then $\omega \leq_{E_1 \sqcup E_2} \omega'$*
6. *If $\omega <_{E_1} \omega'$ and $\omega \leq_{E_2} \omega'$, then $\omega <_{E_1 \sqcup E_2} \omega'$*

A majority syncretic assignment *is a syncretic assignment which satisfies the following condition:*

7. *If $\omega <_{E_2} \omega'$, then $\exists n \; \omega <_{E_1 \sqcup E_2^n} \omega'$*

A fair syncretic assignment *is a syncretic assignment which satisfies the following condition:*

8. *If $\omega <_{K_1} \omega'$, $\omega <_{K_2} \omega''$, and $\omega' \simeq_{\{K_1,K_2\}} \omega''$, then $\omega <_{\{K_1,K_2\}} \omega'$*

Proposition 1 ((Konieczny and Pino Pérez 2002a)) *A merging operator $\triangle$ is an IC merging operator (resp. an IC majority, an IC arbitration operator) iff there exists a syncretic assignment (resp. a majority syncretic assignment, a fair syncretic assignment) that maps each profile E to a total pre-order $\leq_E$ over Ω such that $[\triangle_\mu(E)] = \min([\mu], \leq_E)$.*

Let us now give some examples of IC merging operators using the family of distance-based merging operators (Konieczny, Lang, and Marquis 2004):

Definition 4 *A* distance[3] *between interpretations is a function $d : \Omega \times \Omega \to \mathbb{R}^+$ such that for any $\omega_1, \omega_2 \in \Omega$:*

- *$d(\omega_1, \omega_2) = d(\omega_2, \omega_1)$*
- *$d(\omega_1, \omega_2) = 0$ iff $\omega_1 = \omega_2$*

Usual distances considered in merging are the Hamming distance d_H: $d_H(\omega_1, \omega_2)$ is the number of propositional letters on which the two interpretations differ (this corresponds to the 1-norm distance, also referred to as the Manhattan distance) and the drastic distance d_D, defined as $d_D(\omega_1, \omega_2) = 0$ if $\omega_1 = \omega_2$, and $= 1$ otherwise (this corresponds to the infinity-norm distance, also known as Chebyshev distance).

Definition 5 *An* aggregation function *is a mapping[4] f from $\mathbb{R}^m$ to $\mathbb{R}$, which satisfies:*

- *if $x_i \geq x'_i$, then* **(non-decreasingness)** *$f(x_1, ..., x_i, ..., x_m) \geq f(x_1, ..., x'_i, ..., x_m)$*
- *$f(x_1, \ldots, x_m) = 0$ if $\forall i, \; x_i = 0$* **(minimality)**
- *$f(x) = x$* **(identity)**
- *If σ is a permutation over $\{1, \ldots, m\}$, then $f(x_1, \ldots, x_m) = f(x_{\sigma(1)}, \ldots, x_{\sigma(m)})$* **(symmetry)**

Some additional properties can be considered for f, especially:

- *if $x_i > x'_i$, then* **(strict non-decreasingness)** *$f(x_1, ..., x_i, ..., x_m) > f(x_1, ..., x'_i, ..., x_m)$*

[3]More exactly pseudo-distances since triangular inequality is not required, but for the sake of simplicity we will abuse words in this way.

[4]Strictly speaking, it is a family of mappings, one for each m.

Definition 6 *Let d and f be respectively a distance between interpretations and an aggregation function. The* distance-based merging operator $\triangle^{d,f}$ *is defined by $[\triangle_\mu^{d,f}(E)] = \min([\mu], \leq_E)$, where the total pre-order $\leq_E$ on Ω is defined in the following way (with $E = \{K_1, \ldots, K_n\}$):*

- *$\omega \leq_E \omega'$ iff $d(\omega, E) \leq d(\omega', E)$*
- *$d(\omega, E) = f(d(\omega, K_1), \ldots, d(\omega, K_n))$*
- *$d(\omega, K) = \min_{\omega' \models K} d(\omega, \omega')$*

For usual aggregation functions, whatever the chosen distance, the corresponding distance-based operators exhibit good logical properties ([(Konieczny and Pino Pérez 2002b; Everaere, Konieczny, and Marquis 2010a)]):

Proposition 2 *For any distance d:*

- *if f is the sum Σ, leximin[5], or Σ^n (the sum of the nth powers), then $\triangle^{d,f}$ is an IC majority operator.*
- *if f is leximax, then $\triangle^{d,f}$ is an IC arbitration operator.*

Pre-IC Operators

We now define a general family of belief merging operators, called pre-IC merging operators, obtained by relaxing the two postulates **(IC5)** and **(IC6)** into two natural conditions used in other aggregation theories contexts.

Definition 7 *A merging operator $\triangle$ is* pre-IC merging operator *iff it satisfies* **(IC0)** *to* **(IC4)**, **(IC7)** *to* **(IC8)** *and the following properties:*

(IC5b) $\triangle_\mu(K_1) \wedge \ldots \wedge \triangle_\mu(K_n) \models \triangle_\mu(\{K_1, \ldots, K_n\})$

(IC6b) *If* $\triangle_\mu(K_1) \wedge \ldots \wedge \triangle_\mu(K_n) \not\models \bot$, *then* $\triangle_\mu(\{K_1, \ldots, K_n\}) \models \triangle_\mu(K_1) \wedge \ldots \wedge \triangle_\mu(K_n)$

Thus, switching from IC operators to pre-IC ones simply consists in replacing the postulates **(IC5)** and **(IC6)** postulates by the weaker postulates **(IC5b)** and **(IC6b)**. Indeed, it is easy to prove that **(IC5b)** (resp. **(IC6b)**) is implied by **(IC5)** (resp. **(IC6)**). As a consequence, we have:

Proposition 3 *Every IC merging operator is a pre-IC merging operator.*

Let us now give a representation theorem suited to the pre-IC family:

Definition 8 *A* pre-syncretic assignment *is a function mapping each profile E to a total pre-order $\leq_E$ over Ω such that for any profiles E, E_1, E_2 and for any bases K, K', the following conditions hold:*

1. *If $\omega \models E$ and $\omega' \models E$ then $\omega \equiv_E \omega'$*

2. *If $\omega \models E$ and $\omega' \not\models E$ then $\omega <_E \omega'$*

3. *If $E_1 \equiv E_2$ then $\leq_{E_1} = \leq_{E_2}$*

4. *$\forall \omega \models K$, $\exists \omega' \models K' \; \omega' \leq_{\{K,K'\}} \omega$*

5b. *If $\forall i \; \omega \leq_{K_i} \omega'$ then $\omega \leq_{\{K_1,\ldots,K_n\}} \omega'$*

6b. *If $\forall i \; \omega \leq_{K_i} \omega'$ and $\exists k \; \omega <_{K_k} \omega'$ then $\omega <_{\{K_1,\ldots,K_n\}} \omega'$*

[5]The leximin (resp. leximax) aggregation function selects the interpretations that are minimal for the lexicographic order, once the distances are sorted into the increasing (resp. decreasing) order.

Conditions 5b and 6b are direct translations of Pareto conditions, which are usual conditions in social choice, multicriteria decision making, etc. So they should be considered as minimal aggregation conditions to be satisfied. Conditions 5 and 6 (Definition 3) are much more demanding, since they constraint all unions of two profiles.

Proposition 4 *A merging operator Δ is a pre-IC merging operator iff there exists a pre-syncretic assignment that maps each profile E to a total pre-order $\leq_E$ over Ω such that*

$$[\Delta_\mu(E)] = min([\mu], \leq_E).$$

Proof: **Only If** Let Δ be a pre-IC merging operator. We associate with it a pre-syncretic assignment as follows: for any profile $E = \{K_1, \ldots, K_n\}$, we define $\leq_E$ by $\omega \leq_E \omega'$ if and only if $\omega \models \Delta_\mu(E)$ where $[\mu] = \{\omega, \omega'\}$. From (Konieczny and Pino Pérez 2002a), we know that $\leq_E$ is a total pre-order; furthermore, as Δ satisfies (IC0-IC4) and (IC7-IC8), conditions 1 to 4 are satisfied by the assignment. We have then only to check that conditions $5b$ and $6b$ are satisfied. Consider two interpretations ω and ω', and let μ be such that $[\mu] = \{\omega, \omega'\}$:

Condition $5b$: suppose that $\forall i \in \{1, \ldots, n\}, \omega \leq_{K_i} \omega'$. Then $\forall i \in \{1, \ldots, n\}$, $\omega \models \Delta_\mu(K_i)$. Stated otherwise, $\omega \models \Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$. From (IC5b), we know that $\Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n) \models \Delta_\mu(\{K_1, \ldots, K_n\})$, so $\omega \models \Delta_\mu(\{K_1, \ldots, K_n\})$. Therefore, $\omega \leq_E \omega'$, hence condition $5b$ is satisfied.

Condition $6b$: suppose that $\forall i \in \{1, \ldots, n\}$, $\omega \leq_{K_i} \omega'$ and $\exists k \in \{1, \ldots, n\}$, $\omega <_{K_k} \omega'$. Then $\forall i \in \{1, \ldots, n\}$, $\omega \models \Delta_\mu(K_i)$ and $\omega' \not\models \Delta_\mu(K_k)$. Thus we have $\omega \models \Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$ and $\omega' \not\models \Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$. Hence $[\Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)] = \{\omega\}$. From (IC6b), since $\Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$ is consistent, it must be the case that $\Delta_\mu(\{K_1, \ldots, K_n\}) \models \Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$. Since μ is consistent and Δ satisfies (IC1), $\Delta_\mu(\{K_1, \ldots, K_n\})$ must be consistent as well. As a consequence, we must have $[\Delta_\mu(\{K_1, \ldots, K_n\})] = \{\omega\}$. Therefore, $\omega \models \Delta_\mu(\{K_1, \ldots, K_n\})$ and $\omega' \not\models \Delta_\mu(\{K_1, \ldots, K_n\})$, showing that $\omega <_E \omega'$. Hence condition $6b$ is satisfied.

If Consider a pre-syncretic assignment mapping each belief set E to a total pre-order $\leq_E$ over interpretations. Consider the merging operator Δ defined as follows: $[\Delta_\mu(E)] = min([\mu], \leq_E)$. From (Konieczny and Pino Pérez 2002a), we know that Δ satisfies (IC0-IC4) and (IC7-IC8). Let us now show that Δ satisfies (IC5b-IC6b). Suppose that $\Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$ is consistent (if not, (IC5b-IC6b) is trivially satisfied).

(IC5b) let ω be any model of $\Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$. So $\forall \omega' \models \mu$, $\omega \leq_{K_i} \omega'$. From condition $5b$, we know that $\forall \omega' \models \mu$, $\omega \leq_{\{K_1, \ldots, K_n\}} \omega'$. Hence by definition of Δ, we have $\omega \models \Delta_\mu(\{K_1, \ldots, K_n\})$. Since this holds for any model ω of $\Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$, we conclude that $\Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n) \models \Delta_\mu(\{K_1, \ldots, K_n\})$ and (IC5b) is satisfied.

(IC6b) towards a contradiction, suppose that $\Delta_\mu(\{K_1, \ldots, K_n\}) \not\models \Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$. So $\exists \omega$ s.t. $\omega \models \Delta_\mu(\{K_1, \ldots, K_n\})$ and $\omega \not\models \Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$. Since $\Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$ is consistent, there is an interpretation ω' which is a model of $\Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$. Hence $\forall i \in \{1, \ldots, n\}, \omega' \leq_{K_i} \omega$. Since $\omega \not\models \Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$, $\exists k \in \{1, \ldots, n\}$ s.t. $\omega \not\models \Delta_\mu(K_k)$, showing that $\omega' <_{K_k} \omega$. From condition $6b$, we conclude that $\omega' <_{\{K_1, \ldots, K_n\}} \omega$. This contradicts the fact that $\omega \models \Delta_\mu(\{K_1, \ldots, K_n\})$. Hence, $\Delta_\mu(\{K_1, \ldots, K_n\}) \models \Delta_\mu(K_1) \wedge \ldots \wedge \Delta_\mu(K_n)$, and (IC6b) is satisfied. □

As one can expect, it is much easier to satisfy the pre-IC merging conditions than IC merging ones. We can for instance show that:

Proposition 5 *If d is any distance and f is any aggregation function satisfying strict non-decreasingness, then the merging operator $\Delta^{d,f}$ is a pre-IC merging operator.*

One can compare this result with a corresponding one about distance-based IC merging operators, reported in (Konieczny, Lang, and Marquis 2004), and showing that the aggregation function f has to satisfy two additional conditions in order to guarantee that the distance-based merging operator given by d and f is an IC merging operator.

Conditions for Egalitarian Merging

The only egalitarian property that has been proposed so far for belief merging is the *arbitration* property, represented by the **(Arb)** postulate (or the corresponding semantic condition 8 on syncretic assignments). So a key issue we would like to address is to determine whether other egalitarian properties are possible in the belief merging framework, and, if so, how they relate with arbitration.

If one looks closely at condition 8 in Definition 3, it is clear that the arbitration property only imposes some constraints on the merging of profiles consisting of two bases. Then the IC properties **(IC5)** and **(IC6)** ensure the propagation of the constraints on profiles of any size.

We now propose a first alternative condition, coming from social choice theory, for characterizing egalitarian behaviour in belief merging. This condition, proposed by Hammond (1976) is known in the literature as the Sen-Hammond equity condition (Sen 1997; Suzumura 1983). This condition can be expressed in the following way:

Definition 9 (Sen97) *If person i is worse off than person j both in x and in y, and if i is better off himself in x than in y, while j is better off in y than in x, and if furthermore all others are just as well off in x as in y, then x is socially at least as good as y.*

It is translated in the belief merging setting as constraints on the total pre-orders associated with the input profiles. These constraints concern profiles of arbitrary size, and not only those consisting of two bases. Before doing it, we first need to define a notion of the respective "satisfaction" of two bases given an interpretation:

Definition 10 *Given a merging operator $\triangle$ defining an assignment which maps every profile E to a total pre-order $\leq_E$ over Ω, given an interpretation ω, and two bases K_1*

and K_2, we say that K_1 is better than K_2 *given ω, denoted $K_1 \prec_\omega K_2$, iff $\exists \omega_1 \models K_1, \forall \omega_2 \models K_2, \omega_1 <_\omega \omega_2$.*

We can now give the translation of the Sen-Hammond Equity (SHe) condition:

Definition 11 (Condition (SHe)) *Let $E = \{K_1, \ldots, K_n\}$.*

$$\left.\begin{array}{l} \omega <_{K_1} \omega' \\ \omega' <_{K_2} \omega \\ \forall i \neq 1,2\ \omega \simeq_{K_i} \omega' \\ K_1 \prec_\omega K_2 \\ K_1 \prec_{\omega'} K_2 \end{array}\right\} \Longrightarrow\ \omega' \leq_E \omega$$

When distance-based merging operators are considered, this condition is equivalent to:

Definition 12 (Condition (SHE))

$$\left.\begin{array}{l} d(\omega, K_1) < d(\omega', K_1) < d(\omega', K_2) < d(\omega, K_2) \\ \forall i \neq 1,2\ d(\omega, K_i) = d(\omega', K_i) \end{array}\right\} \Longrightarrow$$

$$f(d(\omega', K_1), d(\omega', K_2), \ldots, d(\omega', K_n)) \leq$$
$$f(d(\omega, K_1), d(\omega, K_2), \ldots, d(\omega, K_n))$$

Proposition 6 *A distance-based merging operator $\Delta^{d,f}$ satisfies* **(SHE)** *if and only if it satisfies* **(SHe)**.

Proof: We prove the equivalence between **(SHe)** and **(SHE)** by showing that those two implications have equivalent premises and equivalent conclusions.

By definition $K_1 \prec_\omega K_2$ iff $\exists \omega_1 \models K_1, \forall \omega_2 \models K_2, \omega_1 <_\omega \omega_2$. Then $\exists \omega_1 \models K_1, \forall \omega_2 \models K_2$, $d(\omega, \omega_1) < d(\omega, \omega_2)$: $\exists \omega_1 \models K_1$, $d(\omega, \omega_1) < min_{\omega_2 \models K_2} d(\omega, \omega_2) = d(\omega, K_2)$. Hence we have $d(\omega, K_1) < d(\omega, K_2)$.

$\omega <_{K_1} \omega' \Leftrightarrow d(\omega, K_1) < d(\omega', K_1)$
$\omega' <_{K_2} \omega \Leftrightarrow d(\omega', K_2) < d(\omega, K_2)$
$\forall i \neq 1,2\ \omega \simeq_{K_i} \omega' \Leftrightarrow \forall i \neq 1,2\ d(\omega, K_i) = d(\omega', K_i)$
$K_1 \prec_\omega K_2 \Leftrightarrow d(\omega, K_1) < d(\omega, K_2)$
$K_1 \prec_{\omega'} K_2 \Leftrightarrow d(\omega', K_1) < d(\omega', K_2)$

This shows the equivalence of the premises. Now, using transitivity, we get: $d(\omega, K_1) < d(\omega', K_1) < d(\omega', K_2) < d(\omega, K_2)$ and $\forall i \neq 1,2\ \omega \simeq_{K_i} \omega' \Leftrightarrow \forall i \neq 1,2\ d(\omega, K_i) = d(\omega', K_i)$. Hence $\omega' \leq_E \omega \Leftrightarrow f(d(\omega', K_1), d(\omega', K_2), \ldots, d(\omega', K_n)) \leq f(d(\omega, K_1), d(\omega, K_2), \ldots, d(\omega, K_n))$. □

This Sen-Hammond Equity condition expresses the following idea: compare two "situations" ω_1 and ω_2 that are equally good for all the agents except two of them. For these two agents one of them (K_1) is in the two situations better than the other agent (K_2). Then the fairer situation is the one that gives the more to the less satisfied agent (K_2).

This condition looks close to the arbitration condition (compare **(SHe)** with condition 8 of Definition 3), but the two conditions are logically independent. Let us now illustrate on a simple example how they differ:

Example 1 *Consider a propositional language over two variables and a distance-based merging operator $\triangle^{d,f}$. Suppose that the distance d on which $\triangle^{d,f}$ is built is the shortest path distance on the following graph:*

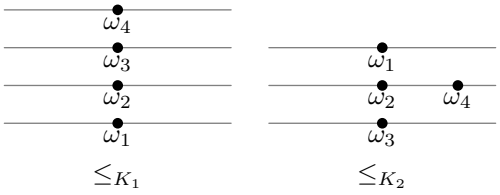


Figure 1: Egalitarian behaviour - (Arb)

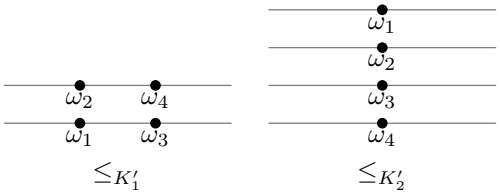


Figure 2: Egalitarian behaviour - (SHE)

$$\omega_1 \overset{1}{\text{——}} \omega_2 \overset{1}{\text{——}} \omega_3 \overset{1}{\text{——}} \omega_4.$$

Figure 1 illustrates the case when the unique model of K_1 is ω_1 and the unique model of K_2 is ω_3. It is clear here that ω_1 and ω_3 play symmetrical roles, so that they cannot be distinguished when K_1 and K_2 are merged. However, on this example, ω_2 appears as a more consensual choice than ω_1 and ω_3 for this merging. Accordingly, **(Arb)** *imposes that ω_2 is strictly preferred to ω_1 and to ω_3 w.r.t. $\leq_{\{K_1,K_2\}}$.*

Suppose now that we want to merge $E' = \{K'_1, K'_2\}$ where $[K'_1]$=$\{\omega_1, \omega_3\}$ and $[K'_2]$ =$\{\omega_4\}$, and that the choice to be made is between ω_1 and ω_2 (i.e., the integrity constraint μ satisfies $[\mu] = \{\omega_1, \omega_2\}$). This is illustrated by Figure 2. **(Arb)** *imposes no constraint on this choice. But clearly, whatever the choice (between $\{\omega_1\}$, $\{\omega_2\}$, or $\{\omega_1, \omega_2\}$), the result will be closer to K'_1 than to K'_2. So a simple equity argument is to commit to the choice that is the best for the furthest base K'_2, so to consider that ω_2 is strictly preferred to ω_1 in $\leq_{E'}$. This is what* **(SHe)** *gives.*

Unfortunately, the family of distance-based IC operators satisfying condition **(SHE)** looks rather limited:

Proposition 7 *Let d be any distance and f be any aggregation function satisfying strict non-decreasingness. The IC merging operator $\Delta^{d,f}$ satisfies condition* **(SHE)** *if and only if $f = leximax$.*

Proof: We know that $\Delta^{d,leximax}$ is an IC operator and that $leximax$ satisfies strict-decreasingness. It is easy to check that $\Delta^{d,leximax}$ satisfies **(SHE)**, so we have mainly to prove the converse implication.

Consider a distance d and an aggregation function f satisfying strict non-decreasingness, and suppose that $\Delta^{d,f}$ satisfies **(SHE)**.

Let $X = (x_1, \ldots, x_n)$ and $X' = (x'_1, \ldots, x'_n)$ be two vectors of distances to a profile E, ordered in the descending way ($\forall i$, $x_i = d(\omega, K_i)$ and $x_i \geq x_{i+1}$; $x'_i = d(\omega', K_i)$ and $x'_i \geq x'_{i+1}$). We have to show that:

$$X <_{leximax} X' \Leftrightarrow f(X) < f(X') \qquad (1)$$

and

$$X \simeq_{leximax} X'^{6} \Leftrightarrow f(X) = f(X') \quad (2)$$

We start with statement (1). Suppose that $X <_{leximax} X'$. We know that $\exists l$ s.t. $\forall i < l, x_i = x'_i$ and $x_l < x'_l$.

Case 1: $l = n$ or $\forall i > l$, $x_i = x'_i$: using strict non-decreasingness we get $f(X) < f(X')$.

Case 2: Suppose that $\exists k > l, x_k \neq x'_k$ and $\forall i \neq k, l, x_i = x'_i$.

- If $x_k \leq x'_k$, then using strict-decreasingness (as $x_l < x'_l$), we get $f(X) < f(X')$.
- If $x_k > x'_k$, then we have $x'_l > x_l \geq x_k > x'_k$. Consider a vector $Y = (y_1, \ldots, y_n)$ s.t. $\forall i \neq k, l$, $y_i = x'_i$ and y_k, y_l such that $x'_l > y_l > y_k > x_l$. As we have $\forall i \neq k, l$, $y_i = x'_i$ and $x'_l > y_l > y_k > x'_k$, because $\Delta^{d,f}$ satisfies **(SHE)**, we can conclude that $f(Y) \leq f(X')$. Furthermore $\forall i \neq k, l$, $y_i = x_i$ and $y_l > y_k > x_l \geq x_k$, using strict decreasingness, we get $f(Y) > f(X)$. By transitivity, we obtain $f(X) < f(X')$.

Case 3: Let us consider now the general case. $\forall i < l, x_i = x'_i$, $x_l < x'_l$ and $l < n$. We define $n - l + 1$ vectors $Y^r = (y^r_1, \ldots, y^r_n)$, for $r = 0$ to $n - l$:

$$y^r_i = \begin{cases} x_i \text{ if } i < l \\ x_l + \frac{r}{n-l}(x'_l - x_l) \text{ if } i = l \\ x_i \text{ if } l+1 \leq i \leq n-r \\ x'_i \text{ if } n-r+1 \leq i \leq n \end{cases}$$

We have $Y^0 = X$ and $Y^{n-l} = X'$, and any two consecutive vectors Y^r and Y^{r+1} differ only on two components, namely y^r_l and y^{r+1}_l on the one hand, and y^r_{n-r} and y^{r+1}_{n-r} on the other hand. Furthermore, $Y^r <_{leximax} Y^{r+1}$, because $\forall i < l, y^r_i = y^{r+1}_i$, $\forall i \leq l, y^r_i \geq y^r_{i+1}$, $\forall i \leq l, y^{r+1}_i \geq y^{r+1}_{i+1}$ and $y^r_l < y^{r+1}_l$. Using Case 2, we can conclude that $f(Y^r) < f(Y^{r+1})$. By transitivity we get $f(Y^0) = f(X) < f(Y^{n-l}) = f(X')$, and the conclusion follows.

Suppose now that $f(X) < f(X')$. If $X \simeq_{leximax} X'$, then $X = X'$ and $f(X) = f(X')$: contradiction. If $X >_{leximax} X'$, then taking advantage of the first part of the proof, we get $f(X) > f(X')$: contradiction. So $X <_{leximax} X'$.

Consider now statement (2).

If $X \simeq_{leximax} X'$, then $X = X'$ and $f(X) = f(X')$.

Suppose now $f(X) = f(X')$. If $X >_{leximax} X'$, then taking advantage of the first part of the proof, we get $f(X) > f(X')$: contradiction. If $X <_{leximax} X'$, then taking advantage of the first part of the proof, we get $f(X) < f(X')$: contradiction. So $X \simeq_{leximax} X'$. □

Let us stress here that the condition of strict non-decreasingness is quite natural and not very demanding. Actually all the aggregation functions giving rise to IC merging operators we are aware of (including Σ, $leximax$, $leximin$, Σ^n, etc.) satisfy non-decreasingness.

Given this proposition, defining other egalitarian distance-based merging operators requires to focus on other equity principles, or to weaken some IC postulates. We explore both ways in the following.

Thus, we first focus on another egalitarian condition from the social choice literature, namely Pigou-Dalton transfer principle (Dalton 1920). The idea underlying it is that every transfer from the most satisfied agent to the least satisfied one decreases the inequalities:

Definition 13 *Let f be an aggregation function. f satisfies* the Pigou-Dalton condition *if for all vectors $X = (x_1, \ldots, x_n)$ and $X' = (x'_1, \ldots, x'_n)$, if $x_1 < x'_1 \leq x'_2 < x_2$ and $x'_1 - x_1 = x_2 - x'_2$ and $\forall i \neq 1, 2$, $x_i = x'_i$ then $f(X) < f(X')$.*

This principle states that if X' can be obtained from X by just making some satisfaction transfer from a well-satisfied agent to a less satisfied one, without changing the fact that the first one is still more satisfied than the second one, then X' is fairer than X.

This principle can be translated as follows for distance-based merging:

Definition 14 (Condition (PD)) *If $\exists k$ and l s.t. $d(\omega, K_k) < d(\omega', K_k) \leq d(\omega', K_l) < d(\omega, K_l)$ and $d(\omega', K_k) - d(\omega, K_k) = d(\omega, K_l) - d(\omega', K_l)$ and $\forall i \neq k$ and $i \neq l, d(\omega, K_i) = d(\omega', K_i)$ then $\omega' <_E \omega$.*

Of course not all distance-based IC operators satisfy the **(PD)** condition. However, it is satisfied by the well-known arbitration operators based on $leximax$:

Proposition 8 *Let d be any distance.*

- $\Delta^{d,leximax}$ *satisfies the* **(PD)** *condition.*
- $\Delta^{d,\Sigma}$ *and* $\Delta^{d,leximin}$ *do not satisfy the* **(PD)** *condition.*

Median Operators

In this section we define a new family of merging operators using generalized median aggregation functions. Interestingly, some operators of this family are pre-IC merging operators and they satisfy **(Arb)**. The idea of using the median value is very motivated by trying to be as fair as possible. Instead of focusing on a unique aggregation function, we study a full family of k-median aggregation functions, inspired by the phantom voters voting rules of (Moulin 1988).

Definition 15 *Let $k \in]0, 1]$ be a real number, the $k-$*median $med^k(\{x_1, \ldots, x_n\})$ *of a multi-set $X = \{x_1, \ldots, x_n\}$ of values from a totally ordered set, is the value $m^k = x_{\sigma(\lceil n*k \rceil)}$ of X, where σ is a permutation of X where the x_i are sorted in ascending order ($\lceil . \rceil$ denotes the ceiling function).*

For $k = 0.5$, the usual notion of median is retrieved.

In many cases these $k-$median functions med^k are not discriminative enough, just like min and max functions. So, we can define k-leximedian operators, noted k-$leximed$, which are to med^k what $leximin$ (resp. $leximax$) is to min (resp. max).

[6] If $X \simeq_{leximax} X'$, then $\forall i, x_i = x'_i$, so $X = X'$.

	K_1	K_2	K_3	distance vector	$med^{0.5}$
000	0	1	1	(0, 1, 1)	1
001	1	0	2	(0, 1, 2)	1
010	1	2	2	(1, 2, 2)	2
011	2	1	3	(1, 2, 3)	2
100	**0**	**2**	**0**	**(0, 0, 2)**	**0**
101	1	1	1	(1, 1, 1)	1
110	1	3	1	(1, 1, 3)	1
111	2	2	2	(2, 2, 2)	2

Table 1: Merging with $\Delta^{d_H, med^{0.5}}$

Definition 16 *Let L_1 and L_2 be two multi-sets consisting of n elements from a totally ordered set:*
$L_1 \leq^k_{leximed} L_2$ *iff*

- $med^k(L_1) < med^k(L_2)$ *or*
- $med^k(L_1) = med^k(L_2)$ *and*
$L_1 \setminus \{med^k(L_1)\} \leq^k_{leximed} L_2 \setminus \{med^k(L_2)\}$

Let us define the k-median merging operators, and later the k-leximedian ones:

Definition 17 *Let $E = \{K_1, ..., K_n\}$ be a profile, d a distance between interpretations and $k \in]0, 1]$ a real number. Let $d^{d,k}_{med}(\omega, E) = med^k(d(\omega, K_1), d(\omega, K_2), \ldots, d(\omega, K_n))$. We define $[\Delta^{d,med^k}_\mu(E)]$ by*

$$[\Delta^{d,med^k}_\mu(E)] = \{\omega \models \mu \mid d^k_{med}(\omega, E) \text{ is minimal}\}.$$

Here is an example illustrating the behaviour of k-median operators:

Example 2 *We consider a profile E of three bases such that $[K_1] = \{000, 100\}$, $[K_2] = \{001\}$ and $[K_3] = \{100\}$. There is no integrity constraint ($\mu = \top$), we use the Hamming distance d_H and the value $k = 0.5$ for the "standard" median. The computations are presented in Table 1.*

We get $[\Delta^{d_H, med^{0.5}}_\top(E)] = \{100\}$. The only selected interpretation is 100, because the best vector is $(0, 0, 2)$, with a median value of 0.

Some specific values of k lead to well-known merging operators: given a profile E of n bases, Δ^{med^ϵ}, with $\epsilon \in]0, \frac{1}{n}]$ corresponds to the min operator, and Δ^{med^α}, with $\alpha \in]\frac{n-1}{n}, 1]$ to the max operator.

Let us now make precise the rationality postulates satisfied by these operators:

Proposition 9 *For any real number $k \in]0, 1]$ and any distance d, Δ^{d,med^k} satisfies* **(IC0)**, **(IC1)**, **(IC3)**, **(IC4)**, **(IC7)**, **(IC8)** *and* **(IC5b)**. **(IC2)**, **(IC5)**, **(IC6)**, **(IC6b)**, **(Maj)** *and* **(Arb)** *are not satisfied in general.*

Proof: For space reasons, we provide only the less obvious proofs.

(IC2) Suppose $k = 0.5$, $d = d_H$ and $[K_1] = \{00\}$, $[K_2] = \{00, 01\}$ and $[K_3] = \{00, 01\}$. We have $d^{d_H,0.5}_m(00, \{K_1, K_2, K_3\}) = 0$ and $d^{d_H,0.5}_m(01, \{K_1, K_2, K_3\}) = 0$: **(IC2)** is not satisfied. Note that for $k \in]\frac{n-1}{n}, 1]$ and any distance d, **(IC2)** is satisfied (the aggregation function max is retrieved).

(IC4) Suppose $K_1 \models \mu$, $K_2 \models \mu$ and $\Delta^{d,med^k}_\mu(\{K_1, K_2\}) \wedge K_1$ consistent.

Let $\omega_1 \models \Delta^{d,med^k}_\mu(\{K_1, K_2\}) \wedge K_1$. Then $d^{d,k}_m(\omega_1, \{K_1, K_2\}) = med^k(d(\omega_1, K_1), d(\omega_1, K_2)) = med^k(0, d(\omega_1, K_2)) = med^k(0, d(\omega_1, \omega_2))$, where $\omega_2 \models K_2$ s.t. $d(\omega_1, K_2) = d(\omega_1, \omega_2)$. Then $d(\omega_2, K_1) \leq d(\omega_1, \omega_2)$: $d^{d,k}_m(\omega_2, \{K_1, K_2\}) = med^k(d(\omega_2, K_1), 0) \leq d^{d,k}_m(\omega_1, \{K_1, K_2\})$. As a consequence ω_2 is selected.

(IC5) and **(IC6)** Suppose $k = 0.5$, and ω_1 s.t. $d^{d,0.5}_m(\omega_1, E_1) = med^{0.5}(0, 0) = 0$, $d^{d,0.5}_m(\omega_1, E_2) = med^{0.5}(3, 4, 4) = 4$ and accordingly $d^{d,0.5}_m(\omega_1, E_1 \sqcup E_2) = med^{0.5}(0, 0, 3, 4, 4) = 3$. Suppose ω_2 s.t. $d^{d,0.5}_m(\omega_2, E_1) = med^{0.5}(1, 1) = 1$, $d^{d,0.5}_m(\omega_2, E_2) = med^{0.5}(2, 6, 7) = 6$ and $d^{d,0.5}_m(\omega_2, E_1 \sqcup E_2) = med^{0.5}(1, 1, 2, 6, 7) = 2$. Then with $[\mu] = \{\omega_1, \omega_2\}$, $[\Delta^{d,med^{0.5}}_\mu(E_1)] = \{\omega_1\}$ and $[\Delta^{d,med^{0.5}}_\mu(E_2)] = \{\omega_1\}$ (and then $\Delta^{d,med^{0.5}}_\mu(E_1)$ and $\Delta^{d,med^{0.5}}_\mu(E_2)$ are jointly consistent), whereas $[\Delta^{d,med^{0.5}}_\mu(E_1 \sqcup E_2)] = \{\omega_2\}$. This example shows that neither **(IC5)** nor **(IC6)** is satisfied.

(Arb) Suppose $0 < k < 0.5$ (so that Δ^{d,med^k} is equivalent to $\Delta^{d,min}$). The following example shows that **(Arb)** is not satisfied. Let ω_1, ω_2, ω_3 be three interpretations s.t. $d(\omega_1, \omega_3) = 1$, $d(\omega_1, \omega_2) = 1$ and $d(\omega_2, \omega_3) = 2$. The bases K_1 and K_2 are defined by $K_1 = \{\omega_2\}$ and $K_2 = \{\omega_3\}$; and the constraints μ_1 and μ_2 are defined by $\mu_1 = \{\omega_1, \omega_3\}$, $\mu_2 = \{\omega_1, \omega_2\}$. Then we have:

$$\left.\begin{array}{c} [\Delta_{\mu_1}(K_1)] = [\Delta_{\mu_2}(K_2)] = \{\omega_1\} \\ [\Delta_{\mu_1 \leftrightarrow \neg\mu_2}(\{K_1, K_2\})] = [\mu_1 \leftrightarrow \neg\mu_2] = \{\omega_2, \omega_3\} \\ \mu_1 \not\models \mu_2 \text{ and } \mu_1 \not\models \mu_2 \end{array}\right\}$$

but $[\Delta_{\mu_1 \vee \mu_2}(\{K_1, K_2\}))] = \{\omega_2, \omega_3\}$ and $[\Delta_{\mu_1}(K_1)] = \{\omega_1\}$: contradiction . □

It turns out that k-median operators satisfy some but not all the expected rationality properties. In particular the very natural postulate **(IC2)**, asking that the result of the merging is just the conjunction of the bases when this conjunction is consistent, is not satisfied (except for values of k making the k-median identical to max). We also have:

Proposition 10 *If $k \geq 0.5$, then Δ^{d,med^k} satisfies* **(Arb)**.

No other equity condition is satisfied by such operators:

Proposition 11 *Whatever k, Δ^{d,med^k} does not satisfy* **(PD)** *or* **(SHE)**.

Proof: Consider the following counter-example, for $k < 0.5$: $med^k(1, 3) = 1$ and $med^k(2, 2) = 2$: $(1, 3) <_{med^k} (2, 2)$: none of **(PD)** or **(SHE)** is satisfied.

Consider the following counter-example, for $0.5 \leq k < 1$: $med^k(0, 4, 4, 7) = 4$ and $med^k(0, 4, 5, 6) = 5$ $(0, 4, 4, 7) <_{med^k} (0, 4, 5, 6)$: none of **(PD)** or **(SHE)** is satisfied.

Consider the following counter-example, for $k = 1$: $med^k(0, 4, 4, 7) = 7$ and $med^k(0, 3, 4, 8) = 8$

$(0,4,4,7) <_{med^k} (0,3,4,8)$: none of **(PD)** or **(SHE)** is satisfied. □

Let us now turn to the k-leximedian operators, that satisfy more expected properties:

Definition 18 *Let $E = \{K_1, ..., K_n\}$ be a profile, μ an integrity constraint, d a distance between interpretations and $k \in]0,1]$ a real number. Define $d^k_{leximed}(\omega, E) = leximed^k(d(\omega, K_1), d(\omega, K_2), \ldots, d(\omega, K_n))$. Then*

$[\Delta_\mu^{d,leximed^k}(E)] = \{\omega \models \mu \mid d_{leximed^k}(\omega, E)$ *is minimal*[7]$\}$

Again, some standard operators are recovered by considering specific values of k: $\Delta^{d,leximed^k}$ with $k \in]0, \frac{1}{n}[$ corresponds to the $leximin$ operator $\Delta^{d,leximin}$, and $\Delta^{d,leximed^1}$ to the $leximax$ operator $\Delta^{d,leximax}$.

As expected $\Delta^{d,leximed^k}$ operators satisfy more interesting properties than Δ^{d,med^k} ones:

Proposition 12 *For any distance d and any $k \in]0,1]$, $\Delta^{d,leximed^k}$ is a pre-IC merging operator.*

We do not have better than that. In particular, these operators are not IC merging ones in the general case:

Proposition 13 *$\Delta^{d,leximed^k}$ does not satisfy any of* **(IC5)**, **(IC6)**, **(Maj)** *and* **(Arb)** *in general.*

Concerning egalitarian properties, we reach the arbitration property, but not the other ones:

Proposition 14 *If $k \geq 0.5$, then $\Delta^{d,leximed^k}$ satisfies* **(Arb)** *but does not satisfy* **(PD)** *or* **(SHE)** *in general.*

It is easy to explain why the condition $k \geq 0.5$ is needed. Indeed, we know that when k goes towards 0 the $leximed^k$ function goes towards $leximin$, and that when k goes towards 1 $leximed^k$ goes towards $leximax$. So it is natural to obtain an egalitarian behaviour for values between "classical" median ($k = 0.5$) and $leximax$.

Considering Proposition 7, Proposition 14 shows that relaxing some IC postulates is a way for escaping from the $leximax$-based operators while satisfying some egalitarian condition.

Cumulative Sum Merging Operators

A very convenient representation of the inequalities of a distribution of income is the Lorenz curve (Lorenz 1905). The principle is to focus on the poorest (least satisfied) agents, by looking first as the utility of the poorest one, then at the sum of the utilities of the two poorest ones, etc. To be more precise, on a Lorenz curve, each element k of the x-axis corresponds to the k poorest agents and the value associated with it on the y-axis is the sum of the utilities of those agents.

This curve can be interpreted in different ways to measure how much a distribution is fair. In particular, the fairest distribution is when for any n, the $n\%$ poorest agents own $n\%$ of the income. The well-known Gini coefficient (Gini 1921; Sen 1973; Dutta 2002), one of the main inequality measures, is the (double of the) area between the Lorenz curve and the fairest distribution.

[7] w.r.t. the lexicographic order.

	K_1	K_2	K_3	Cum. Sat.	Σ
000	**3**	**2**	**2**	**(2, 4, 7)**	**13**
001	2	3	1	(1, 3, 6)	10
010	2	1	1	(1, 2, 4)	7
011	1	2	0	(0, 1, 3)	4
100	3	1	3	(1, 4, 7)	12
101	2	2	2	(2, 4, 6)	12
110	2	0	2	(0, 2, 4)	6
111	1	1	1	(1, 2, 3)	6

Table 2: Merging with $\Delta^{CS(d_H,\Sigma)}$

In the following, we adhere to the notion of cumulative sum for defining a new family of merging operators. We translate the distance between a base and an interpretation into a satisfaction value by reversing the scale. Then we compute the cumulative satisfaction vector and take advantage of an aggregation function on it.

Let us define formally the cumulative sum merging operators:

Definition 19 *Let d be a distance between interpretations, f an aggregation function, E a profile and μ an integrity constraint. Let $M = max(\{d(\omega, \omega') \mid \omega, \omega' \in \Omega\})$. For an interpretation ω, we consider the vector $(l_{\sigma(1)}, \ldots, l_{\sigma(n)})$ where $l_i = M - d(\omega, K_i)$ is the satisfaction value of agent i for the interpretation ω, and σ is the permutation of $\{1, \ldots, n\}$ sorting the l_i in ascending order (the less the least satisfied). Then we define the vector of cumulated satisfaction of μ, $W_d(\omega, E) = (W_1, \ldots, W_n)$, where $W_i = \Sigma_{k=1}^{i} l_{\sigma(k)}$. Finally, the selected interpretations are the ones which maximize the cumulated satisfaction $W_d(\omega, E)$:*

$$[\Delta_\mu^{CS(d,f)}(E)] = \{\omega \models \mu \mid f(W_d(\omega, E)) \text{ is maximal } \}.$$

Let us illustrate the behavior of a cumulative sum merging operator on a simple example:

Example 3 *We step back to Example 2. The computations are presented in Table 2, in which the values represent satisfaction values of the bases (the more the best), and not anymore distances (the less the best).*

We get $[\Delta_\top^{CS(d_H,\Sigma)}(E)] = \{000\}$. The selected interpretation is 000, because the sum of the cumulative satisfaction vector gives the maximal value of 13.

We recover some existing operators as cumulative sum ones:

Proposition 15 *Let d be any distance,*

- $\Delta^{CS(d,leximin)} = \Delta^{d,leximax}$.
- $\Delta^{CS(d,leximax)} = \Delta^{d,leximin}$.

Let us finally turn to the logical properties:

Proposition 16 • *If d is any distance, and f is an aggregation function satisfying strict non-decreasingness, then $\Delta^{CS(d,f)}$ is a pre-IC merging operator.*

$rank:$	$w(\omega, K_i):$	$w(\omega', K_i):$
1	x_1	x_1
$\dots$	$\dots$	$\dots$
$k-1$	x_{k-1}	x_{k-1}
k	x_k	x_{k+1}
$k+1$	x_{k+1}	x_{k+2}
$\dots$	$\dots$	$\dots$
$k'-1$	$x_{k'-1}$	$x_{k'}$
k'	$x_{k'}$	$x'_k = x_k + g$
$k'+1$	$x_{k'+1}$	$x_{k'+1}$
$\dots$	$\dots$	$\dots$
$l'-1$	$x_{l'-1}$	$x_{l'-1}$
l'	$x_{l'}$	$x'_l = x_l - g$
$l'+1$	$x_{l'+1}$	$x_{l'}$
$l-1$	x_{l-1}	x_{l-2}
l	x_l	x_{l-1}
$l+1$	x_{l+1}	x_{l+1}
$\dots$	$\dots$	$\dots$
n	x_n	x_n

- $\Delta^{CS(d,f)}$ *does not satisfy* **(Arb)** *and* **(Maj)** *in the general case.*
- $\Delta^{CS(d,f)}$ *does not satisfy* **(IC5)** *and* **(IC6)** *in the general case.*

While **(Arb)** is not satisfied in general, the Pigou-Dalton condition is ensured for instance when any sum of the m-powers, m varying, is used as aggregation function:

Proposition 17 *For any integer m, $\Delta^{CS(d,\Sigma^m)}$ satisfies* **(PD)***, but* **(SHE)** *is not satisfied.*

Proof:

$\Delta^{CS(d,\Sigma^m)}$ satifies **(PD)**.

Consider ω and ω' s.t. $\exists k$ and l s.t. $d(\omega, K_k) < d(\omega', K_k) \leq d(\omega', K_l) < d(\omega, K_l)$ and $d(\omega', K_k) - d(\omega, K_k) = d(\omega, K_l) - d(\omega', K_l)$ and $\forall i \neq k$ and $i \neq l, d(\omega, K_i) = d(\omega', K_i)$. We note g the value $d(\omega', K_k) - d(\omega, K_k)$, so we have $d(\omega', K_k) = d(\omega, K_k) + g$ and $d(\omega', K_l) = d(\omega, K_l) - g$. g is strictly positive.

We note $\forall i, x_i = n - d(\omega, K_i)$, $x'_k = n - d(\omega', K_k)$ and $x'_l = n - d(\omega', K_l)$. We suppose that the x_i are ordered in the ascending way: $x_1 \leq x_2 \leq \ldots \leq x_n$.

Let l' and k' be the relative ranks of x'_l and x'_k into the x_i-th. We suppose $l' \neq l$ and $k' \neq k$. As $x_l > x'_l \geq x'_k > x_k$, we know that if $l' \neq l$, $l' < l$ and if $k' \neq k$, $k' > k$. We have then the following table, if $k' \neq k$ and $l' \neq l$:

From rank 1 to rank $k-1$, the x_i-th are identical so the W_i^m also.

From rank k to rank $k'-1$, we have in $\Sigma^m(W_d(\omega, E))$:

$$\Sigma^m_{k\ldots(k'-1)}(W_d(\omega, E)) = (x_1 + \ldots + x_{k-1} + x_k)^m + \ldots + (x_1 + \ldots + x_{k-1} + x_k + \ldots + x_{k'-1})^m$$

and in $\Sigma^m(W_d(\omega', E))$:

$$\Sigma^m_{k\ldots(k'-1)}(W_d(\omega', E)) = (x_1 + \ldots + x_{k-1} + x_{k+1})^m + \ldots + (x_1 + \ldots + x_{k-1} + x_{k+1}) + (x_1 + \ldots + x_{k-1} + x'_k)^m \quad (3)$$

In Equation 3, the values from x_k to $x_{k'-1}$ are replaced by the values from x_{k+1} to $x_{k'}$. Since the x_i-th are increasing, we have $\Sigma^m_{k\ldots(k'-1)}(W_d(\omega, E)) \leq \Sigma^m_{k\ldots(k'-1)}(W_d(\omega', E))$.

From rank k' to rank $l'-1$, we have in $\Sigma^m(W_d(\omega, E))$:

$$\Sigma^m_{k'\ldots(l'-1)}(W_d(\omega, E)) = (x_1 + \ldots + x_{k-1} + x_k + \ldots + x_{k'})^m + \ldots + (x_1 + \ldots + x_{k-1} + x_k + \ldots + x_{l'-1})^m$$

and in $\Sigma^m(W_d(\omega', E))$:

$$\Sigma^m_{k'\ldots(l'-1)}(W_d(\omega', E)) = (x_1 + \ldots + x_{k-1} + x_{k+1} + \ldots + x_{k'-1} + x'_k)^m + \ldots + (x_1 + \ldots + x_{k-1} + x_{k+1} + \ldots + x'_k + x_{k'+1} + \ldots + x_{l'-1})^m = (x_1 + \ldots + x_{k-1} + x_{k+1} + \ldots + x_{k'-1} + x_k + g)^m + \ldots + (x_1 + \ldots + x_{k-1} + x_{k+1} + \ldots + x_k + g + x_{k'+1} + \ldots + x_{l'-1})^m$$

(because $x'_k = x_k + g$). Each term of $\Sigma^m(W_d(\omega', E))$ is equal to one term of $\Sigma^m(W_d(\omega, E))$ plus g. As $g > 0$, we have: $\Sigma^m_{k'\ldots(l'-1)}(W_d(\omega, E)) < \Sigma^m_{k'\ldots(l'-1)}(W_d(\omega', E))$.

From rank l' to rank l, we have in $\Sigma^m(W_d(\omega, E))$:

$$\Sigma^m_{l'\ldots l}(W_d(\omega, E)) = (x_1 + \ldots + x_k + x_{k+1} + \ldots + x_{l'-1} + x_{l'})^m + \ldots + (x_1 + \ldots + x_{l'-1} + x_{l'} + \ldots + x_{l-1} + x_l)^m$$

and in $\Sigma^m(W_d(\omega', E))$:

$$\Sigma^m_{l'\ldots l}(W_d(\omega', E)) = (x_1 + \ldots + x_k + g + x_{k+1} + \ldots + x_{l'-1} + x'_l)^m + \ldots + (x_1 + \ldots + x_k + g + x_{k+1} + \ldots + x'_l + x_{l'} + \ldots + x_{l-2} + x_{l-1})^m = (x_1 + \ldots + x_k + g + x_{k+1} + \ldots + x_{l'-1} + x_l - g)^m + \ldots + (x_1 + \ldots + x_k + g + x_{k+1} + \ldots + x_l - g + x_{l'} + \ldots + x_{l-1})^m$$

because $x'_l = x_l - g$, so:

$$\Sigma^m_{l'\ldots l}(W_d(\omega', E)) = (x_1 + \ldots + x_k + x_{k+1} + \ldots + x_{l'-1} + x_l)^m + \ldots + (x_1 + \ldots + x_k + x_{k+1} + \ldots + x_l + x_{l'} \ldots + x_{l-1})^m$$

In each term of $\Sigma^m_{l'\ldots l}(W_d(\omega', E))$, compared with $\Sigma^m_{l'\ldots l}(W_d(\omega, E))$, one element $x_{l'+i}$ with i varying from 0 to $l - l'$ of the sum W_j is remplaced by x_l. Since the x_i-th are increasing, $x_{l'+i} \leq x_l$ for $0 \leq i \leq l - l'$, so $\Sigma^m_{l'\ldots l}(W_d(\omega, E)) \leq \Sigma^m_{l'\ldots l}(W_d(\omega', E))$.

From rank $l+1$ to rank n, we have in $\Sigma^m(W_d(\omega, E))$:

$$\Sigma^m_{(l+1)\ldots n}(W_d(\omega, E)) = (x_1 + \ldots + x_{l'-1} + x_{l'} + \ldots + x_{l-1} + x_{l+1})^m + \ldots + (x_1 + \ldots + x_{l'-1} + x_{l'} + \ldots + x_{l-1} + x_l + x_{l+1} + \ldots + x_n)^m$$

and in $\Sigma^m(W_d(\omega', E))$:

$$\Sigma^m_{(l+1)\ldots n}(W_d(\omega', E)) = (x_1 + \ldots + x_k + x_{k+1} + \ldots + x_{l'-1} + x_{l+1})^m + \ldots + (x_1 + \ldots + x_k + x_{k+1} + \ldots + x'_l + x_{l'} + \ldots + x_n)^m$$

So $\Sigma^m_{(l+1)\ldots n}(W_d(\omega', E)) = \Sigma^m_{(l+1)\ldots n}(W_d(\omega, E))$.

Finally, we obtain $\Sigma^m(W_d(\omega, E)) < \Sigma^m(W_d(\omega', E))$, for all integer m, and ω' is selected : **(PD)** is satisfied.

For the other cases, when x'_l (resp. x'_k) has the same rank as x_l (resp. x_k), there are less cases to be studied, the table is simpler, but the result still holds. □

Conclusion

In this paper, we have investigated alternative egalitarian conditions to the arbitration postulate. Especially, we have translated to the belief merging framework two egalitarian conditions: Sen-Hammond equity, and Pigou-Dalton property.

We have shown that the distance-based merging operators satisfying Sen-Hammond equity are mainly those for which $leximax$ is the aggregation function. This led us to introduce a new family of belief merging operators, the pre-IC operators, which includes the family of IC merging operators as a specific case. We have pointed out a representation theorem for this family, which allows us to define easily some distance-based pre-IC operators. In order to enrich the family of egalitarian merging operators, we have considered two new families of belief merging operators, based respectively on the median and on an aggregated sum (Lorenz curves). We have shown that the operators based on the $leximed^k$ aggregation functions (with $k \geq 0.5$) are pre-IC operators which satisfy the arbitration postulate **(Arb)**. We have also proved that the $\Delta^{CS(d,\Sigma^n)}$ operators from the cumulative sum family are pre-IC operators satisfying **(PD)**.

Besides theory-oriented results, this work produced two interesting families of egalitarian operators: cumulative sum ones and $leximed^k$ ones. Before this work the only known egalitarian merging operators were the $leximax$-based ones. Egalitarian operators are significant for all applications where consensual results are expected, i.e., all agents are supposed to be satisfied in the best way, contrariwise to utilitarian/majority operators. Whereas utilitarian operators can be used when the information sources are sensors or databases, egalitarian operators are particularly important in applications where such sources are agents which are autonomous enough to reject the result of the merging if they consider that it is too far from their position.

Acknowledgments

This work benefited from the support of the AMANDE project of the French National Research Agency (ANR-13-BS02-0004).

References

Arrow, K.; Sen, A. K.; and Suzumura, K., eds. 2002. *Handbook of Social Choice and Welfare*, volume 1. North-Holland.

Dalton, H. 1920. The measurement of the inequality of incomes. *Economic Journal* 30:pp. 348–461.

Dutta, B. 2002. Inequality, Poverty and Welfare. In Arrow, K. J.; Sen, A. K.; and Suzumura, K., eds., *Handbook of Social Choice and Welfare*. Elsevier. 597–633.

Everaere, P.; Konieczny, S.; and Marquis, P. 2010a. Disjunctive merging: Quota and gmin merging operators. *Artificial Intelligence* 174(12-13):824–849.

Everaere, P.; Konieczny, S.; and Marquis, P. 2010b. The epistemic view of belief merging: Can we track the truth? In *Proceedings of the Nineteenth European Conference on Artificial Intelligence (ECAI'10)*, 621–626.

Gini, C. 1921. Measurement of inequality of incomes. *The Economic Journal* 31(121):pp. 124–126.

Hammond, P. J. 1976. Equity, Arrow's conditions, and Rawls' difference principle. *Econometrica* 44(4):pp. 793–804.

Harsanyi, J. C. 1955. Cardinal welfare, individualistic ethics, and interpersonal comparisons of utility. *Journal of Political Economy* 63:309.

Konieczny, S., and Pino Pérez, R. 2002a. Merging information under constraints: a logical framework. *Journal of Logic and Computation* 12(5):773–808.

Konieczny, S., and Pino Pérez, R. 2002b. On the frontier between arbitration and majority. In *Proceedings of KR'02*, 109–118.

Konieczny, S.; Lang, J.; and Marquis, P. 2004. DA^2 merging operators. *Artificial Intelligence* 157:49–79.

Lin, J., and Mendelzon, A. O. 1999. Knowledge base merging by majority. In *Dynamic Worlds: From the Frame Problem to Knowledge Management*, volume 12 of *Applied Logic Series*, 195–218. Kluwer.

Lorenz, M. O. 1905. Methods of measuring the concentration of wealth. *Publications of the American Statistical Association* 9(70):pp. 209–219.

Moulin, H. 1988. *Axioms of Cooperative Decision Making*. Econometric society monographs. Cambridge University Press. chapter 9.

Rawls, J. 1971. *A Theory of Justice*. Harvard University Press.

Revesz, P. Z. 1997. On the semantics of arbitration. *International Journal of Algebra and Computation* 7(2):133–160.

Sen, A. 1973. *On Economic Inequality*. Oxford University Press.

Sen, A. 1997. *Choice, Welfare and Measurement*. Harvard University Press.

Sen, A. 2005. Social choice theory. In Arrow, K. J., and Intriligator, M., eds., *Handbook of Mathematical Economics*, volume 3. Elsevier. 1073–1181.

Suzumura, K. 1983. *Rational choice, collective decisions and social welfare*. Cambridge University Press.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Justified Beliefs by Justified Arguments

Davide Grossi and Wiebe van der Hoek
University of Liverpool

Abstract

The paper addresses how the information state of an agent relates to the arguments that the agent endorses. Information states are modeled in doxastic logic and arguments by recasting abstract argumentation theory in a modal logic format. The two perspectives are combined by an application of the theory of product logics, delivering sound and complete systems in which the interaction of arguments and beliefs is investigated.

1 Introduction

Leaving probabilistic approaches aside, one can recognize in the KR tradition two main paradigms in the formal representation of the knowledge or, more generally, of the information state of an agent. One describes knowledge as what holds in all the situations the agent considers possible—the epistemic alternatives. That is the perspective of epistemic logic (Meyer and van der Hoek 1995; Fagin *et al.* 1995). The other describes knowledge as some form of 'stable' theory about the world the agent endorses in the face of conflicting information. This is the tradition of non-monotonic reasoning and of its most successful recent development, abstract argumentation theory (Dung 1995).

The paper combines these two dimensions in a unified framework where questions concerning the interaction of arguments and beliefs can be systematically addressed, such as: Is the set of arguments supporting an agent's doxastic state 'justifiable' from the point of view of abstract argumentation (e.g., conflict-free, admissible, stable, etc.)?

Technically, the combination of the two perspectives—epistemic and argumentation-theoretic—is achieved by deploying techniques and results from the theory of product modal logics (Gabbay *et al.* 2003). The key idea consists in taking states in a doxastic (Kripke) model and arguments in a (Dung) attack graph as two orthogonal dimensions for the description of the information state of an agent. This intuition suggests the use of bi-dimensional structures for the study of argument-based beliefs. The logics obtained are studied in their completeness and finite model properties.

More generally, the paper lays a bridge between epistemic logic and argumentation theory. We hope that the results presented can foster further interaction between the two fields.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Related work To the best of our knowledge, the only works to date attempting to interface argumentation with epistemic logic are (Grossi 2012) and (Schwarzentruber *et al.* 2012). The first is concerned with the analysis, in dynamic epistemic logic (van Ditmarsch *et al.* 2007), of the fixpoint behavior of some argumentation theoretic notions, and the second enriches the standard framework of abstract argumentation by enabling arguers to hold beliefs about other arguers' available arguments. In its broad purposes, the present paper can be related to recent work (in particular, (Artemov 2008; van Benthem and Pacuit 2011)) aiming at explicitly modeling the 'justifications' or 'reasons' upon which agents base their information state. Our paper shows the viability of using product logics for this type of analysis.

Outline of the paper The following section prepares the ground recapitulating some basic notions from both epistemic logic and abstract argumentation. We then proceed in a modular fashion. First (in Section 3) we introduce a logic that combines the simplest modal logic of argumentation (logic K, cf. (Grossi 2010)) and the simplest modal doxastic logic ($\mathsf{KD45}$). Argument-belief interaction properties are then formalized and discussed in Section 4. Then, we allow the agent to have preferences among the arguments. Finally, Section 6 then introduces and studies a more expressive logic, able to formalize ample fragments of Dung's argumentation theory as well as a rich set of doxastic attitudes based on the set of arguments that an agent endorses. We conclude by pointing at future research directions at the interface of abstract argumentation and epistemic logic. All proofs (or proof sketches) are provided in the appendix.

2 Preliminaries

We start by introducing the basic building blocks of our analysis: simple structures for representing beliefs and for representing arguments and their attacks. We then move to motivate a specific way of combining the two: products. Although we give all necessary definitions, space limitations demand we do this succinctly: the reader may wish to consult (Meyer and van der Hoek 1995) for more background on doxastic logic, (Baroni and Giacomin 2009;

Baroni *et al.* 2011) for abstract argumentation, and (Gabbay and Shehtman 1998; Gabbay *et al.* 2003) for products in modal logic.

Doxastic structures

Definition 1 (Doxastic frame). *A doxastic frame is a tuple $\mathcal{D} = \langle S, B\rangle$ where: i) S is a non-empty set of states; ii) $B \subseteq S$, s.t., $B \neq \varnothing$. The class of all doxastic frames is denoted $\mathfrak{D}$.*

A doxastic frame represents the non-empty set B of states that an agent holds as most plausible among the set of all states S. We will refer to elements of B as *doxastic alternatives* or *doxastically accessible states*.

Given a doxastic frame, a Kripke model is obtained by adding a valuation function $\mathcal{V} : \mathbf{P} \longrightarrow \wp(S)$ interpreting a set of atoms $\mathbf{P}$. A doxastic modality $\Diamond\!\!\!\!-_{\mathsf{B}}$ can then be interpreted as follows:

$$\langle \mathcal{D}, \mathcal{V}\rangle, s \vDash \Diamond\!\!\!\!-_{\mathsf{B}}\varphi \quad \text{iff} \quad \exists s' \in B : \langle \mathcal{D}, \mathcal{V}\rangle, s' \vDash \varphi \qquad (1)$$

For any diamond $\Diamond$ in this paper, we define an associated $\Box$ as $\Box\varphi = \neg \Diamond \neg\varphi$. So $\boxminus_{\mathsf{B}}\varphi$ means that the agent believes φ.

Doxastic frames are somewhat simpler than the structures typically used to study beliefs, namely relational frames where the accessibility relation is transitive and euclidean (cf. (Meyer and van der Hoek 1995)). The two, however, can be proven equivalent for the purpose of this paper, using some standard modal logic arguments:

Theorem 1. *On the basic modal language, the class of doxastic frames is modally equivalent to the class of serial, transitive and euclidean relational frames.*

So the logic of the class of doxastic frames is completely axiomatized by the standard axiom system for logic $\mathsf{KD45}$ containing: the rules Modus Ponens ($\mathbf{MP}$: form φ and $\varphi \rightarrow \psi$, infer ψ) and Necessitation ($\mathbf{Nec}$: from φ, infer $\boxminus_{\mathsf{B}}\varphi$); the axioms $\mathbf{K} : \boxminus_{\mathsf{B}}(\varphi \rightarrow \psi) \rightarrow (\boxminus_{\mathsf{B}}\varphi \rightarrow \boxminus_{\mathsf{B}}\psi)$ representing the agent's ability to reason propositionally, $\mathbf{D} : \neg \boxminus_{\mathsf{B}} \bot$ (beliefs are consistent) and the axioms $\mathbf{4} : \boxminus_{\mathsf{B}}\varphi \rightarrow \boxminus_{\mathsf{B}} \boxminus_{\mathsf{B}} \varphi$ and $\mathbf{5} : \neg\boxminus_{\mathsf{B}} \neg\varphi \rightarrow \boxminus_{\mathsf{B}}\neg\boxminus_{\mathsf{B}} \neg\varphi$ representing positive and negative introspection, respectively. For later reference, recall that $\mathsf{S5}$ is the logic $\mathsf{KD45} + \mathbf{T} : \boxminus_{\mathsf{B}}\varphi \rightarrow \varphi$.

Argumentative structures

We start by the key structure of abstract argumentation:

Definition 2 (Attack graphs (Dung 1995)). *An attack graph is a tuple $\mathcal{A} = \langle A, \rightarrow\rangle$ where: i) A is a non-empty set of arguments; ii) $\rightarrow \subseteq A^2$ is a binary relation ($a \rightarrow b$ stands for a attacks b). The class of all attack graphs is denoted $\mathfrak{A}$.*

These relational structures are the building blocks of abstract argumentation theory. Once A is taken to represent a set of arguments, and $\rightarrow$ an 'attack' relation between arguments, the study of these structures provides general insights on how competing arguments interact, and structural properties of subsets of A can be taken to formalize how collections of arguments form 'justifiable' positions in an argumentation ((Baroni and Giacomin 2009; Baroni *et al.* 2011)).

Conflict-free	$X \subseteq \{x \mid \nexists y \in X : x \leftarrow y\}$
Self-defended	$X \subseteq \{x \mid \forall y : x \leftarrow y \Rightarrow \exists z \in X, y \leftarrow z\}$
Admissible	X is conflict-free and self-defended
Complete extension	X is conflict-free &
	$X = \{x \mid \forall y : x \leftarrow y \Rightarrow \exists z \in X, y \leftarrow z\}$
Stable extension	$X = \{x \mid \nexists y \in X : x \leftarrow y\}$

Table 1: Properties of a set of arguments X in a given attack graph $\mathcal{A}$.

In this paper we will touch upon the argumentation-theoretic notions of conflict-freeness, self-defense, admissibility, complete and stable extensions. Table 1 recapitulates these notions for the ease of the reader.

A Kripke model $\mathcal{M} = \langle A, \leftarrow, \mathcal{V}\rangle$ can be obtained from an attack graph by inverting the attack relation ($a \leftarrow b$ denotes that a is attacked by b) and by adding a valuation function $\mathcal{V} : \mathbf{P} \longrightarrow \wp(A)$ interpreting a set of propositional atoms $\mathbf{P}$. Consider now a modality $\Diamond\!\!\!\!-_{\mathsf{A}}$ with the following semantics:

$$\langle \mathcal{A}, \mathcal{V}\rangle, a \vDash \Diamond\!\!\!\!-_{\mathsf{A}}\varphi \text{ iff } \exists b \in A : a \leftarrow b \;\&\; \langle \mathcal{A}, \mathcal{V}\rangle, b \vDash \varphi \quad (2)$$

An argument a satisfies $\Diamond\!\!\!\!-_{\mathsf{A}}\varphi$ iff some attacker b of a satisfies φ. The logic of $\Diamond\!\!\!\!-_{\mathsf{A}}$ defined by the class of attack graphs is, obviously, K. (Grossi 2010) shows that modal logic K can express a number of argumentation theoretic notions from (Dung 1995), such as: $\neg \Diamond\!\!\!\!-_{\mathsf{A}} p$, expressing that the current argument is not attacked by p; or $\boxdot_{\mathsf{A}} \Diamond\!\!\!\!-_{\mathsf{A}} p$ expressing that the current argument is 'defended' by p-arguments (i.e., its attackers are attacked by p-arguments). The logic K is axiomatized by rules $\mathbf{MP}$ and $\mathbf{Nec}$, and axiom $\mathbf{K}$.

Doxo-argumentative structures

Let us start with a simple motivating example:

Example 1 (After (Modgil 2009)). *Consider two individuals exchanging arguments about the weather forecast. Argument a: "Today will be dry in London since the BBC forecasts sunshine". And argument a': "Today will be wet in London since CNN forecasts rain". We have two arguments (a and a') concerned with whether the 'real' situation is a state s where the sun shines in London or in a state s' where it rains in London.*

In general, starting from a set of doxastic alternatives S and a set of 'arguments' A, we are after structures that can support the analysis of how elements of S interact with elements of A. We want to be able to express properties of state-argument pairs (s, a) such as, in particular, that "argument a supports state s" in the sense that a is ground for s or increases its likeliness.

Example 2 (After (Modgil 2009), continued). *Let then $S = \{s, s'\}$ and $A = \{a, a'\}$. We can represent the simple scenario of Example 1 by the model on the left of Figure 1, where the dark circles indicate that the pairs at issue have a property of interest (in this case the property of 'supporting'): a supports s and does not support s'; vice versa a' supports s' and does not support s.*

So our domain becomes the Cartesian product $S \times A$. Now,

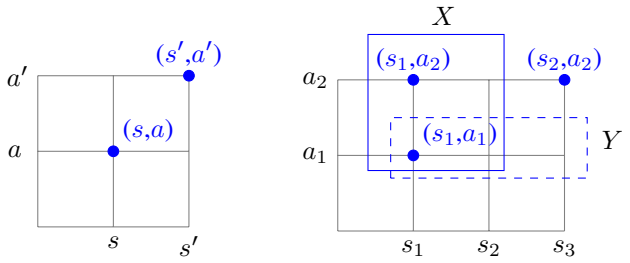


Figure 1: Two doxo-argumentative structures.

if S and A also come equipped with accessibility relations—the doxastic one in case of S and the attack one in case of A—then studying how the logics of these relations interact in $S \times A$ would allow one to talk about properties of state-argument pairs that are of a doxastic and argumentation-theoretic type in the same language. The paper takes this perspective and sets out to develop a formal theory of how arguments and their attacks relate to the doxastic state of an agent. The key tool in accomplishing this, is that of product logics.

Properties of states, arguments and their pairs

The above set up allows one to represent any property of state-argument pairs. In this paper, we will focus on one in particular, namely the property of *support* of an argument for a doxastic state. This will be denoted by symbol σ. Notice that the same argument can support several states and the same state can be supported by several arguments. If a set of states B is supported by a same argument a we say that a *supports* B.

Critically, the set up allows one to represent properties of states, or arguments alone. For instance, to express that a state s has a property X (e.g., 'the sun shines in London') it suffices for X to be true of all the state-argument pairs whose state is s, i.e., to be a set of columns in the Cartesian plane. Similarly, properties Y of arguments alone (e.g., 'being upheld by BBC') can be represented in the same fashion, i.e., by sets of rows of the Cartesian plane.[1]

Figure 1 (right) illustrates all these different properties. Dark circles indicate a relation (e.g., support) between arguments and states: a_2 supports both s_1 and s_3; s_1 is supported by both a_1 and a_2. Rectangle X represents a state-only property, of states s_1 and s_2, and rectangle Y (dashed line) represents an argument-only property, of argument a_1.

Product logics

The product of two (uni-)modal logics[2] is defined as follows (Gabbay and Shehtman 1998). The product $\mathcal{F} \times \mathcal{F}'$ between two frames $\mathcal{F} = \langle S, R\rangle$ and $\mathcal{F}' = \langle S', R'\rangle$ is the frame $\langle S \times S', H, V\rangle$ where:

$$\begin{array}{lll}(s,s')H(t,t') & \text{iff} & sRt \text{ AND } s' = t' \\ (s,s')V(t,t') & \text{iff} & s'R't' \text{ AND } s = t\end{array}$$

Intuitively, the product of two frames can be depicted as a Cartesian plane where H is the relation on the 'horizontal' dimension consisting of the set S and V is the relation on the the 'vertical' one consisting of set S'. Following (Marx 1999), we will use $\Diamond$ to denote the modality interpreted over H—'horizontal' modality—and $\Diamond$ the modality interpreted over V—'vertical' modality.

The product of two classes of frames $\mathfrak{F}$ and $\mathfrak{F}'$ is $\{\mathcal{F} \times \mathcal{F}' \mid \mathcal{F} \in \mathfrak{F} \text{ AND } \mathcal{F}' \in \mathfrak{F}'\}$. Now, given two logics L and L' the product $\mathsf{L} \times \mathsf{L}'$ is the logic of the class of frames defined by the product of the two largest classes $\mathfrak{F}$ and $\mathfrak{F}'$ for which the two logics are complete. For instance, $\mathsf{K} \times \mathsf{K}$ is the logic of the class of all frames consisting of the product of two frames. Here we study products between logics L and L' where L—the 'horizontal' logic—is a doxastic logic and L'—the 'vertical' logic—is a modal logic for argumentation.

3 A simple product logic: $\mathsf{KD45} \times \mathsf{K}$

As a first framework in which to investigate interaction principles between arguments and doxastic states we consider the product of the simplest doxastic logic, namely $\mathsf{KD45}$, with the simplest modal logic of attack graphs, i.e., K.

Syntax and semantics

The language $\mathcal{L}(\Diamond_{\mathsf{B}}, \Diamond_{\mathsf{A}})$, has the following BNF:

$$p \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond_{\mathsf{B}}\varphi \mid \Diamond_{\mathsf{A}}\varphi$$

where p belongs to the set of atoms $\mathbf{P}$. For any language $\mathcal{L}$ that we consider, the variant $\mathcal{L}^{\sigma}$ adds an atom σ to it, where σ intuitively says of (s,a) that s is supported by a. Semantics is given as follows. Let $\mathcal{D}$ be a doxastic frame on S and $\mathcal{A}$ an attack graph on A. A $\mathsf{KD45} \times \mathsf{K}$ model is a structure $\mathcal{M} = \langle \mathcal{D} \times \mathcal{A}, \mathcal{V}\rangle$ where $\mathcal{V} : \mathbf{P} \longrightarrow \wp(S \times A)$. The satisfaction relation is defined by the standard Boolean clauses plus the following clauses (cf. the expressions (1) and (2)):

$$\begin{array}{lll}\mathcal{M},(s,a) \vDash \Diamond_{\mathsf{A}}\varphi & \text{iff} & \exists a' \in A : a \leftarrow a' \;\&\; \mathcal{M},(s,a') \vDash \varphi \\ \mathcal{M},(s,a) \vDash \Diamond_{\mathsf{B}}\varphi & \text{iff} & \exists s' \in B : \mathcal{M},(s',a) \vDash \varphi\end{array}$$

As usual, we denote the truth-set of a formula φ by $[\![\varphi]\!]$.

The expression $\mathcal{M},(s,a) \vDash \varphi$ can be interpreted as: given the 'actual' state is s and the 'currently entertained' argument is a, φ holds. So $\boxminus_{\mathsf{B}}\varphi$ expresses the property that, by keeping fixed the current argument, all pairs consisting of the current argument and a doxastically accessible state, satisfy φ. Intuitively: it is believed that φ holds of the current argument. Similarly, modalities $\Diamond_{\mathsf{A}}$ and $\boxminus_{\mathsf{A}}$ express properties of the attack relation. So $\Diamond_{\mathsf{A}}\varphi$ expresses the property that, by keeping the current state fixed, there exists a pair consisting of the current state and an argument that attacks the current argument, and this pair satisfies φ.

Remark 1 (Satisfaction in products). *As customary in modal logic, formulae are interpreted on pointed models*

[1] Cf. (Sano 2011).

[2] The multi-modal case is a straightforward generalization. Cf. (Gabbay and Shehtman 1998).

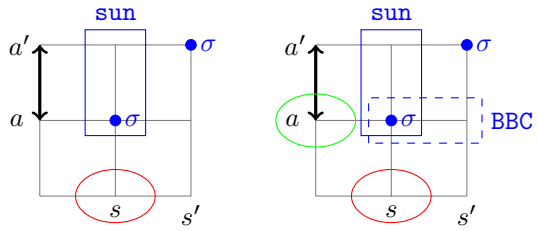


Figure 2: Rendering of Example 1 as a KD45×K model (left) and a DA model (right). Universal relations are not depicted.

"$\mathcal{M}, (s, a)$". So, when we interpret a formula we fix both an argument and a state and $\mathcal{M}, (s, a) \vDash \varphi$ can be interpreted as: given the 'actual' state is s and the 'currently entertained' argument is a, φ holds.

Example 3 (After (Modgil 2009), continued). *We extend Example 1 by making explicit that the two arguments a and a' attack one another, and that the agent believes the actual state is s (the only one supported by argument a), so that the set of doxastic alternatives B is $\{s\}$ (left of Figure 2, where the ellipsis encloses the set of doxastic alternatives). Dark circles denote the truth set of atom σ (representing 'support') and the rectangle denotes the truth set of an atom* sun *(for 'sunshine'). Notice that* sun *is here a 'column' property (Remark 2). Arrows on the vertical dimension denote attack.*

Here are some formulae true at (s, a): (i) $\boxminus_{\mathsf{B}}\sigma$; (ii) $\boxminus_{\mathsf{B}}(\sigma \wedge \boxplus_{\mathsf{A}}\neg\sigma)$. Intuitively: (i) says that my beliefs are supported by the current argument, in the sense that all my doxastic alternatives are supported by the current argument; (ii) says that all my doxastic alternatives are supported by the current argument and all its attackers are not supported by my doxastic alternatives. Here are some of the formulae true at (s', a'): (i) $\neg$sun $\wedge \boxminus_{\mathsf{B}}$sun; (ii) $\boxminus_{\mathsf{B}}\neg\sigma$; (iii) $\boxminus_{\mathsf{B}} \Diamond_{\mathsf{A}} \sigma$. Intuitively: (i) expresses a standard false belief property 'I believe sun *of the current argument but* sun *is false'; (ii) expresses that no doxastic alternative is supported by the current argument; (iii) states that for all doxastic alternatives, the current argument is attacked by an argument supporting that alternative.*

Metalogical results

Axiomatization Logic KD45 × K is the logic on $\mathcal{L}$ of the class of frames consisting of the product of one single-agent doxastic frame and one attack frame. It is axiomatized by taking KD45 for $\Diamond_{\mathsf{B}}$, and K for $\Diamond_{\mathsf{A}}$, plus the two following axioms:

Com	$\Diamond_{\mathsf{A}} \Diamond_{\mathsf{B}} \varphi \leftrightarrow \Diamond_{\mathsf{B}} \Diamond_{\mathsf{A}} \varphi$
Con	$\Diamond_{\mathsf{B}} \boxplus_{\mathsf{A}} \varphi \rightarrow \boxplus_{\mathsf{A}} \Diamond_{\mathsf{B}} \varphi$

We will come back later to the intuitive meaning of these.

The completeness of this axiom system is established as a corollary of known general theorems (Gabbay and Shehtman 1998, Theorem 7.12) or (Gabbay *et al.* 2003, Theorem 5.9): we only need to check that the axioms for K and KD45 are either without atoms (i.e., frame formulae), or else have a specific syntactic form (called *pseudo-transitivity*), which is the case.[3]

Decidability & Complexity A logic has the *product finite model property* w.r.t. class $\mathfrak{F}\times\mathfrak{F}'$ iff every satisfiable formula on that class can be satisfied on a model built on the product of finite frames in $\mathfrak{F}$ and $\mathfrak{F}'$.

Logic KD45 × K has the (strong) product finite model property as every φ can be satisfied on a finite model of size exponential in the length of φ (Gabbay *et al.* 2003, Theorem 6.56). Logic KD45 × K is therefore decidable and its satisfiability problem is NEXPTIME-complete (Gabbay *et al.* 2003, Theorem 6.57).

4 Interaction of attacks and beliefs

We now turn to the sort of insights that we gain by modeling the interaction of doxastic structures (Kripke frames) and argument structures (attack graphs) as a product, and what logic KD45 × K allows us to say about such interaction.

Beliefs

Let us first define formally the classes of state- and argument-properties we have informally introduced in an earlier section. A state-property (column property) is expressed by a formula $\sigma \in \mathcal{L}(\Diamond_{\mathsf{B}}, \Diamond_{\mathsf{A}})$ such that the following equation holds:

$$[\![\sigma]\!] = \{(s, a) \in S \times A \mid \exists a' \in A : (s, a') \in [\![\sigma]\!]\}\,.$$

Similarly, an argument-property (row property) is expressed by a formula $\gamma \in \mathcal{L}(\Diamond_{\mathsf{B}}, \Diamond_{\mathsf{A}})$ such that the following equation holds:

$$[\![\alpha]\!] = \{(s, a) \in S \times A \mid \exists s' \in A : (s', a) \in [\![\alpha]\!]\}\,.^4$$

In the remaining of the paper we will denote state-properties by a variable σ and argument properties by a variable α.

Logic KD45 × K supports the representation of two types of beliefs: the beliefs an agent can entertain about an 'external' state-of-affairs, and the beliefs an agent can entertain about how the states it considers possible interact with available arguments.

Beliefs of the first type concern state-properties: $\boxminus_{\mathsf{B}}\sigma$. Notice that beliefs of state-properties are themselves state-properties. That is, formulae $\boxminus_{\mathsf{B}}\sigma$ are independent of the vertical coordinate, so we have $\mathcal{M}, (s, a) \vDash \boxminus_{\mathsf{B}}\sigma$ iff $\mathcal{M}, (s, a') \vDash \boxminus_{\mathsf{B}}\sigma$ for any $a' \in A$. Put differently: the equivalence $\boxminus_{\mathsf{B}}\sigma \leftrightarrow \boxplus_U \boxminus_{\mathsf{B}} \sigma$ is valid.

Beliefs of the second type concern properties φ expressing some kind of relation between doxastic states and arguments. Both kinds of beliefs behave according to logic

[3] More precisely, the theorem states that the product $\mathsf{L}_H \times \mathsf{L}_V$ of two logics L_H and L_V whose axioms are either formulae from the frame language (i.e., without atoms, like $\Diamond\top$) or have the form $\bigtriangledown \Box p \rightarrow \bigtriangleup p$ where $\bigtriangledown$ is a sequence of possibly different diamonds and $\bigtriangleup$ a sequence of possibly different boxes (so-called *pseudo-transitive formulae*), is completely axiomatized by the axioms of L_H, the axioms of L_V, plus the **Com** and **Con** axioms for each pairs of modalities in the combined language. Notice that pseudo-transitive formulae are Sahlqvist formulae.

[4] Cf. Remark 2 below.

KD45, therefore validating the standard doxastic principles (e.g., positive and negative introspection). To this type of beliefs we turn in the next section.

Interaction properties

Example 3 has already shown interaction properties expressible in KD45 × K. Here are a few more examples, which we illustrate using the designated atom σ:

$$(a)\quad \Box_{\mathsf{A}} \Diamond_{\mathsf{A}} \top \qquad (b)\quad \sigma \wedge \Diamond_{\mathsf{A}} \top \wedge \Box_{\mathsf{A}} \sigma$$
$$(c)\quad \Diamond\!\!\!\!-_{\mathsf{B}} \sigma \qquad (d)\quad \sigma \wedge \Box_{\mathsf{A}} \neg \Diamond\!\!\!\!-_{\mathsf{B}} \sigma$$

To understand these properties, let us see what they express once evaluated at a pointed model $\mathcal{M}, (s, a)$, where s denotes the current state and a the argument currently entertained. Formula (a) expresses that all attackers of a have an attacker, i.e., a is *defended* by some argument. Formula (b) states that the current argument a supports the current alternative s, but a has attackers and indeed all attackers of a support s. Intuitively, a is therefore a 'weak' argument for s. Formula (c) expresses that some doxastically accessible alternatives are supported by the current argument a and hence the beliefs of the agent are supported by that argument. Formula (d) states again that the current argument is supported by the current alternative and that all attackers of the current argument do not support any alternative. This property expresses a form of *safety* of a state-argument pair: the argument supports the alternative but there is no other alternative which is supported by an argument attacking the current one.

It is worth observing that that the latter property generalises a purely argumentative notion of safety as non-existence of attackers ($\Box_{\mathsf{A}} \bot$), by requiring that if there are attackers, then these are not effective as they rule out any doxastic alternative. So it makes the requirement of not having attackers relative to whether they support any further doxastic alternatives. The formula $\boxminus_{\mathsf{B}}(\sigma \rightarrow \Box_{\mathsf{A}} \neg \Diamond\!\!\!\!-_{\mathsf{B}} \sigma)$ would then express the agent's belief of safety of the current argument, across the set of doxastic alternatives.

Validities of KD45 × K and their interpretation

Let us move to the interpretation of **Com** and **Con**.[5]

Intuitively, $\Diamond_{\mathsf{A}} \Diamond\!\!\!\!-_{\mathsf{B}} \varphi \leftrightarrow \Diamond\!\!\!\!-_{\mathsf{B}} \Diamond_{\mathsf{A}} \varphi$ states that the property 'there is an attacker of the argument I'm currently entertaining, and a state I consider doxastically possible, such that the first is in a relation φ with the latter' can be formulated independently of the order of the diamonds involved. The order of quantification over attacks and over doxastic alternatives does not matter.

As to $\Diamond\!\!\!\!-_{\mathsf{B}} \Box_{\mathsf{A}} \varphi \rightarrow \Box_{\mathsf{A}} \Diamond\!\!\!\!-_{\mathsf{B}} \varphi$, it expresses that, if I hold a state as doxastically possible which is in relation φ with all the attackers of an argument I currently entertain, then all the attackers of that argument I currently entertain are in relation φ with one of my doxastic alternatives.

We conclude this section by listing some further validities of KD45 × K. They concern the interaction properties of arguments and doxastic states and can all be proven by applications of axioms **Com** and **Con**.

[5] **Con** is also known as the Church-Rosser property.

Proposition 1. *The following are validities of* KD45 × K*:*

$$\boxminus_{\mathsf{B}} \Diamond_{\mathsf{A}} \varphi \rightarrow \Diamond\!\!\!\!-_{\mathsf{B}} \Diamond_{\mathsf{A}} \boxminus_{\mathsf{B}} \varphi \tag{3}$$
$$\Diamond_{\mathsf{A}} \Diamond\!\!\!\!-_{\mathsf{B}} \varphi \rightarrow \boxminus_{\mathsf{B}} \Diamond_{\mathsf{A}} \Diamond\!\!\!\!-_{\mathsf{B}} \varphi \tag{4}$$

Again, to interpret these formulae it is useful to remember that they are evaluated at a given state and argument. Let us comment on the Formula (3): if all the states I hold possible are such that they are in relation φ (e.g., 'support') with an attacker of the argument I currently entertain, then I hold it possible that some attacker is in relation φ (e.g., 'support') with all the states I consider possible.

Neighborhoods

We conclude this section with a comment on related work. Evidence and justification for doxastic attitudes have been studied within neighborhood semantics (Chellas 1980) in (van Benthem and Pacuit 2011). It is worth, albeit only in passing, to notice a straightforward formal link between our two-dimensional models and neighborhood frames. The intuition is that the set of states supported by each argument can be viewed as the element of a neighborhood. So, any doxo-argumentative model $\mathcal{M} = \langle \mathcal{D} \times \mathcal{A}, \mathcal{V} \rangle$, where $\mathcal{V}$ evaluates a designated constant σ, induces a static neighborhood frame $\langle S, \mathcal{N} \rangle$ where: $\mathcal{N} = \{\{s \in S \mid \exists a \in A : \mathcal{M}, (s, a) \vDash \sigma\}\}_{a \in A}$ Vice versa, any static neighborhood frame $\langle S, \mathcal{N} \rangle$ induces a class of doxo-argumentative models such $\mathcal{M} = \langle \mathcal{D} \times \mathcal{A}, \mathcal{V} \rangle$ such that: the support of $\mathcal{D}$ is S; the support of $\mathcal{A}$ is $\mathcal{N}$; $\mathcal{V}$ is such that $\mathcal{M}, (s, a) \vDash \sigma$ iff $s \in a \in \mathcal{N}$. With respect to neighborhood frames, doxo-argumentative models contain extra structural information encoded in the attack relation.

5 Believing & endorsing

We now want to express properties about the interaction of a given set of doxastic alternatives and a given set of arguments representing the arguments the agent *endorses*. Logic KD45 × K lacks this expressivity. So, to do this, we enrich the K component of the logic. We move from simple attack graphs $\langle A, \rightarrow \rangle$ to attack graphs designating a non-empty set E of 'endorsed arguments': $\mathcal{E} = \langle A, \rightarrow, E \rangle$. We call these frames *enriched attack graphs* and we denote their class by $\mathfrak{E}$. In this section we study the logic of the products of doxastic frames with enriched attack graphs and discuss the sort of insights the logic enables on the interaction between arguments and beliefs.

Syntax and semantics

The language $\mathcal{L}^E$ is defined by the following BNF:

$$p \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\!\!\!\!-_{\mathsf{B}} \varphi \mid \Diamond_{\mathsf{E}} \varphi \mid \Diamond_{\mathsf{A}} \varphi \mid \Diamond\!\!\!\!-_{\mathsf{U}} \varphi \mid \Diamond_{\mathsf{U}} \varphi$$

with $p \in \mathbf{P}$. As before, $\mathcal{L}^{E,\sigma}$ extends $\mathcal{L}^E$ with the designated atom σ. Modalities $\Diamond\!\!\!\!-_{\mathsf{B}}$ and $\Diamond_{\mathsf{A}}$ are as above. As to the others: $\Diamond_{\mathsf{E}}$ means 'for some endorsed argument by keeping fixed the current state'; $\Diamond\!\!\!\!-_{\mathsf{U}}$ means 'for some state by keeping fixed the current argument'; $\Diamond_{\mathsf{U}}$ means 'for some argument by keeping fixed the current state'. Notice that $\Diamond\!\!\!\!-_{\mathsf{U}}$ and $\Diamond_{\mathsf{U}}$ are nothing but universal modalities for the horizontal and, vertical dimensions, respectively. We refer to the fragment

of $\mathcal{L}^E$ containing only $\Diamond\!\!\!\!-_{\mathsf{B}}$, $\Diamond\!\!\!\!-_{\mathsf{U}}$ modalities as its *horizontal fragment*, and to the fragment of $\mathcal{L}^E$ containing only $\Diamond_{\mathsf{E}}$, $\Diamond_{\mathsf{A}}$ and $\Diamond_{\mathsf{U}}$ modalities as its *vertical fragment*.

The semantics for $\mathcal{L}^E$ is defined as follows. Let $\mathcal{F}$ be a doxastic frame on S and $\mathcal{A}$ an enriched attack graph on A. A model is a structure $\mathcal{M} = \langle \mathcal{D} \times \mathcal{E}, \mathcal{V} \rangle$ where $\mathcal{V} : \mathbf{P} \longrightarrow \wp(S \times A)$. The satisfaction relation is defined as follows (clauses are limited to the newly introduced operators):

$$\begin{array}{lll} \mathcal{M}, (s,a) \vDash \Diamond_{\mathsf{E}}\varphi & \text{iff} & \exists a' \in E : \mathcal{M}, (s,a') \vDash \varphi \\ \mathcal{M}, (s,a) \vDash \Diamond\!\!\!\!-_{\mathsf{U}}\varphi & \text{iff} & \exists s' \in S : \mathcal{M}, (s',a) \vDash \varphi \\ \mathcal{M}, (s,a) \vDash \Diamond_{\mathsf{U}}\varphi & \text{iff} & \exists a' \in A : \mathcal{M}, (s,a') \vDash \varphi \end{array}$$

We call the logic on $\mathcal{L}^E$ defined by the class of the above models DA *doxastic argument logic*.

Remark 2 ('Column' and 'row' properties). *Modalities $\Diamond\!\!\!\!-_{\mathsf{U}}$ and $\Diamond_{\mathsf{U}}$ make it possible to express that given properties are state- or argument-properties. A state- or 'column' property is a formula σ that satisfies (in the given model) the equation: $\sigma \leftrightarrow \Box\!\!\!\!-_{\mathsf{U}}\sigma$ (i.e., σ holds of the current pair independently of the argument). Similarly, an argument- or 'row' property is expressed by a formula α that satisfies the equation: $\alpha \leftrightarrow \boxminus_{\mathsf{U}}\alpha$ (i.e., α holds of the current pair independently of the state). Examples are, in the right model of Figure 2: $\Box\!\!\!\!-_{\mathsf{U}}\mathtt{sun}$ of state s, and $\boxminus_{\mathsf{U}}\mathtt{BBC}$ of argument a. We can then express that the agent* believes *φ, in the sense that all its doxastc alternatives have column property φ by $\boxminus_{\mathsf{B}} \Box\!\!\!\!-_{\mathsf{U}} \varphi$ (in the example: $\boxminus_{\mathsf{B}} \Box\!\!\!\!-_{\mathsf{U}} \mathtt{sun}$). Similarly, we can express that all the arguments endorsed by the agent have row property φ by $\Box_{\mathsf{E}} \boxminus_{\mathsf{U}} \varphi$ (in the example: $\Box_{\mathsf{E}} \boxminus_{\mathsf{U}} \mathtt{BBC}$).*

Example 4 (After (Modgil 2009), continued). *We expand Example 3 by recasting it as a* DA *model and making thus explicit that the agent endorses argument a, that is: $E = \{a\}$. The new model is depicted on the right of Figure 2 where the ellipsis on the vertical axis encloses the set of endorsed arguments. The following are validities of the above model (thus independent of the point of evaluation):*

$$\begin{array}{ll} (a) & \boxminus_{\mathsf{B}} \Diamond_{\mathsf{E}} \sigma \\ (b) & \Box_{\mathsf{E}} \boxminus_{\mathsf{B}} (\sigma \rightarrow \neg \Diamond_{\mathsf{A}} \Diamond\!\!\!\!-_{\mathsf{B}} \sigma) \\ (c) & \boxminus_{\mathsf{B}} \Box\!\!\!\!-_{\mathsf{U}} \mathtt{sun} \\ (d) & \Box\!\!\!\!-_{\mathsf{U}} \boxminus_{\mathsf{U}} ((\sigma \wedge \mathtt{sun}) \rightarrow \neg \Diamond_{\mathsf{A}} \Diamond\!\!\!\!-_{\mathsf{U}} (\sigma \wedge \mathtt{sun})) \end{array}$$

Intuitively: (a) expresses that the agent's beliefs are supported by arguments it endorses; (b) that for all pairs of endorsed arguments and doxastic alternatives, if the argument supports the alternative, then there is no attacker of that argument which supports some other alternative; (c) that all doxastic alternatives satisfy sun*—i.e., the agent believes* sun *(cf. Remark 2); (d) that the state-argument pairs satisfying $\sigma \wedge$ sun do not contain any attack between their arguments, that is, their arguments ar* conflict-free.[6] *Notice that none of these properties was expressible in the simpler language $\mathcal{L}(\Diamond\!\!\!\!-_{\mathsf{B}}, \Diamond_{\mathsf{A}})$.*

[6] We come back to a general formalization of conflict-freeness and the other properties of Table 1 below.

Remark 3 (On the 'support' relation between states and arguments). *We have taken a liberal view on the notion of support of a state by an argument. We have treated support as just one of the possible relations between arguments and states (others can be 'incompatibility', 'weak support', etc. depending on the application domain) and modeled it through a dedicated atom σ, whose interpretation has not been constrained. It is important to notice that meaningful classes of* DA *models can be isolated by strengthening our axiom system with axioms enforcing desirable properties on σ, such as: $\Diamond\!\!\!\!-_{\mathsf{U}}\sigma$ (every argument supports some state); $\boxminus_{\mathsf{B}} \Diamond_{\mathsf{E}} \sigma$ (the agent considers possible only states that are supported by some argument).*

Metalogical results

We provide here a sound and complete axiom system for DA. The axiomatization is built from axiom systems for the horizontal and vertical components of the logic. We then establish the decidability of the logic and sample a few results on its expressivity.

Axiomatization: horizontal logic Let us first concern ourselves with axiomatizing the logic determined by the class of doxastic frames on the horizontal fragment of $\mathcal{L}^E$. Consider the logic—call it DA^H—defined by the rules and axioms of KD45 for modality $\Diamond\!\!\!\!-_{\mathsf{B}}$, the rules and axioms of S5 for modality $\Diamond\!\!\!\!-_{\mathsf{U}}$ plus:

$\mathbf{Inc}_{\mathsf{BU}}$	$\Diamond\!\!\!\!-_{\mathsf{B}}\varphi \rightarrow \Diamond\!\!\!\!-_{\mathsf{U}}\varphi$	$\mathbf{4}_{\mathsf{BU}}$	$\boxminus_{\mathsf{B}}\varphi \rightarrow \boxminus_{\mathsf{U}} \boxminus_{\mathsf{B}} \varphi$
$\mathbf{5}_{\mathsf{BU}}$	$\Diamond\!\!\!\!-_{\mathsf{B}}\varphi \rightarrow \boxminus_{\mathsf{U}} \Diamond\!\!\!\!-_{\mathsf{B}} \varphi$		

Lemma 1. *DA^H is sound and complete for class $\mathfrak{D}$.*

Axiomatization: vertical logic As to the logic determined by the class of enriched attack graphs on the vertical fragment of $\mathcal{L}^E$ we can proceed in a similar fashion. Notice that an enriched attack graph $\mathcal{E} = \langle A, \rightarrow, E \rangle$ can be viewed as a doxastic frame to which a binary relation $\rightarrow$ is added. So, consider the logic—call it DA^V—defined by: the rules and axioms of KD45 for modality $\Diamond_{\mathsf{E}}$, the rules and axioms of K for modality $\Diamond_{\mathsf{A}}$ and the rules and axioms of S5 for modality $\Diamond\!\!\!\!-_{\mathsf{U}}$, plus:

$\mathbf{Inc}_{\mathsf{AU}}$	$\Diamond_{\mathsf{A}}\varphi \rightarrow \Diamond_{\mathsf{U}}\varphi$	$\mathbf{Inc}_{\mathsf{EU}}$	$\Diamond_{\mathsf{E}}\varphi \rightarrow \Diamond_{\mathsf{U}}\varphi$
$\mathbf{4}_{\mathsf{EU}}$	$\Box_{\mathsf{E}}\varphi \rightarrow \Box_{\mathsf{U}} \Box_{\mathsf{E}} \varphi$	$\mathbf{5}_{\mathsf{EU}}$	$\Diamond_{\mathsf{E}}\varphi \rightarrow \Box_{\mathsf{U}} \Diamond_{\mathsf{E}} \varphi$

Lemma 2. *DA^V is sound and complete for class $\mathfrak{E}$.*

The proof proceeds as for Lemma 1.

An axiom system for DA Everything is now in place to prove the following result:

Theorem 2 (Completeness of DA**).** *The logic defined by the axioms and rules of DA^H and of DA^V plus the following instances of* **Com** *and* **Con***:*

Com	$\Diamond \Diamond \varphi \leftrightarrow \Diamond \Diamond \varphi$
Con	$\Diamond \Box \varphi \rightarrow \Box \Diamond \varphi$

where $\Diamond \in \{\Diamond\!\!\!\!-_{\mathsf{B}}, \Diamond\!\!\!\!-_{\mathsf{U}}\}$, and $\Diamond \in \{\Diamond_{\mathsf{E}}, \Diamond_{\mathsf{A}}, \Diamond\!\!\!\!-_{\mathsf{U}}\}$, is sound and complete for the class $\mathfrak{D} \times \mathfrak{E}$ consisting of products of doxastic frames and enriched attack graphs.

(Product) Finite model property & decidability We finally establish the two following results concerning the decidability of DA:

Theorem 3. *Logic* DA *does not have the product finite model property.*

Theorem 4. *Logic* DA *is decidable.*

Expressivity on frames We conclude with some technical considerations about the expressivity of DA. The previous subsection has shown how basic notions from argumentation theory can be characterized at the level of models in DA. We now look at the feasibility of characterizations at frame level: are there formulae which characterize whether the set of endorsed arguments E is conflict-free and self-defended? This is a novel application of frame correspondence theory (van Benthem 1983) to abstract argumentation. We start by the following:

Proposition 2. *Let $\mathcal{F} = \mathcal{D} \times \mathcal{E}$. If $\mathcal{F} \vDash \Box_{\mathsf{E}}(\boxminus_{\mathsf{U}} p \rightarrow \neg \Diamond_{\mathsf{A}} \boxminus_{\mathsf{U}} p)$ then E in $\mathcal{E}$ is conflict free.*

We next show that Proposition 2 cannot be strengthened to a *characterization* of conflict-freeness:

Theorem 5. *There exists no formula φ of* DA *s.t.: $\mathcal{F} \vDash \varphi$ iff E is conflict-free, for $\mathcal{F} \in \mathfrak{D} \times \mathfrak{E}$.*

However, a characterization *does* exists for self-defense:

Proposition 3 (Characterization of self-defense of E). *Let $\mathcal{F} = \mathcal{D} \times \mathcal{E}$. Then $\mathcal{F} \vDash (\Box_{\mathsf{E}} p \wedge \Diamond_{\mathsf{E}} \Diamond_{\mathsf{A}} q) \rightarrow \Diamond_{\mathsf{U}}(q \wedge \Diamond_{\mathsf{A}} p)$ if and only if E in $\mathcal{E}$ is self-defended.*

6 Justified beliefs & arguments in DA

This section concludes by showcasing DA as a rich framework for the study of argument-based beliefs.

From beliefs to argument-based beliefs

In standard doxastic logic beliefs are properties that are true of all (doxastically) accessible states. In DA, beliefs are properties that are true of all (doxastically) accessible state-argument pairs, *independently of the argument*. So, "I believe that φ" is formalized by $\boxminus_{\mathsf{B}} \Box_{\mathsf{U}} \varphi$ (cf. Remark 2). But now, the argumentative structure available in DA allows us to differentiate between beliefs based on how they relate to underlying arguments, for instance by being supported by some such arguments. The following definitions illustrate the interaction between beliefs and endorsed arguments:

$$SB\varphi := \boxminus_{\mathsf{B}}(\Box_{\mathsf{U}}\varphi \wedge \Diamond_{\mathsf{U}}\sigma) \quad (5)$$

$$EB\varphi := \boxminus_{\mathsf{B}}(\Box_{\mathsf{U}}\varphi \wedge \Diamond_{\mathsf{E}}\sigma) \quad (6)$$

$$JB(\varphi,\psi) := \boxminus_{\mathsf{B}}(\Box_{\mathsf{U}}\varphi \wedge \Diamond_{\mathsf{E}}(\sigma \wedge \boxminus_{\mathsf{U}}\psi)) \quad (7)$$

The formulae appear in the order from logically weaker to logically stronger and may be taken as definitions of different types of argument-based beliefs in φ: supported belief (SB), endorsed supported belief (EB), and justified belief (JB) where the type of justification is expressed by ψ. Formula (5) expresses that the agent believes φ and its beliefs are supported by arguments—each doxastic alternative is supported by some argument. Formula (6) expresses that the agent believes φ and its beliefs are supported by arguments it endorses. Finally, Formula (7) states that the agent believes φ and its beliefs are supported by arguments it endorses and which have property ψ (notice again the use of $\boxminus_{\mathsf{U}}$ to express that such ψ is a 'row' property, *independent of the doxastic state*). In particular, such ψ can be chosen to express properties such as: "the argument belongs to a given conflict-free set", "the argument belongs to a given stable extension" and the like, to which we now turn.

Dung's argumentation theory in DA We show now how DA can capture some fundamental argumentation-theoretic properties (Table 1). These properties are properties of sets of arguments, that is, row properties (cf. Remark 2). For instance, we want to formalize the property that says that a given row property $\boxminus_{\mathsf{U}}\varphi$ identifies a set of admissible arguments. The strategy for obtaining these formalizations is based on (Grossi 2010). Let $\blacksquare := \boxminus_{\mathsf{U}}\Box_{\mathsf{U}}$:

(i)	φ is a conflict free set	$\blacksquare(\boxminus_{\mathsf{U}}\varphi \rightarrow \neg \Diamond_{\mathsf{A}} \boxminus_{\mathsf{U}}\varphi)$
(ii)	φ is a self-defended set	$\blacksquare(\boxminus_{\mathsf{U}}\varphi \rightarrow \Box_{\mathsf{A}} \Diamond_{\mathsf{A}} \boxminus_{\mathsf{U}}\varphi)$
(iii)	φ is a fixpoint of $\Box_{\mathsf{A}}\Diamond_{\mathsf{A}}$	$\blacksquare(\boxminus_{\mathsf{U}}\varphi \leftrightarrow \Box_{\mathsf{A}} \Diamond_{\mathsf{A}} \boxminus_{\mathsf{U}}\varphi)$
(iv)	φ is an admissible set	$(i) \wedge (ii)$
(v)	φ is a complete extension	$(i) \wedge (iii)$
(vi)	φ is a stable extension	$\blacksquare(\boxminus_{\mathsf{U}}\varphi \leftrightarrow \neg \Diamond_{\mathsf{A}} \boxminus_{\mathsf{U}}\varphi)$

We comment on (i) and (ii). Formula (i) says that if an argument satisfies row property $\boxminus_{\mathsf{U}}\varphi$, then no attacker of that argument exists which also satisfies the same property. Formula (ii) states that all arguments satisfying row property $\boxminus_{\mathsf{U}}\varphi$ are such that all their attackers are defended by some argument satisfying the same property.

Example 5 (After (Modgil 2009), continued). *Let us go back to the model on the right of Figure 2. It is a validity of the model that: the agent believes that* sun *is the case; that it has arguments for that belief, which it endorses; and that those arguments also have the property of belonging to the conflict-free set of arguments specified by property* BBC*. That is, the agent holds a belief to the effect that* sun *is the case and such belief is justified by the set of conflict-free* BBC *arguments. Formally: $JB(\mathtt{sun}, \psi)$ where $\psi := \mathtt{BBC} \wedge \blacksquare(\boxminus_{\mathsf{U}}\mathtt{BBC} \rightarrow \neg \Diamond_{\mathsf{A}} \boxminus_{\mathsf{U}}\mathtt{BBC})$. It is also a validity of the model that the set of arguments supporting the agent's doxastic alternatives is an admissible set, that is:*

$$\blacksquare(\Diamond\!\!\!\!-_{\mathsf{B}}\sigma \rightarrow \neg \Diamond_{\mathsf{A}} \Diamond\!\!\!\!-_{\mathsf{B}}\sigma) \wedge \blacksquare(\Diamond\!\!\!\!-_{\mathsf{B}}\sigma \rightarrow \Box_{\mathsf{A}} \Diamond_{\mathsf{A}} \Diamond\!\!\!\!-_{\mathsf{B}}\sigma)$$

where, notice, $\Diamond\!\!\!\!-_{\mathsf{B}}\sigma$ is a row property ($\Diamond\!\!\!\!-_{\mathsf{B}}\sigma \leftrightarrow \boxminus_{\mathsf{U}} \Diamond\!\!\!\!-_{\mathsf{B}} \sigma$).

Properties of argument-based beliefs

Having interfaced beliefs with properties of supporting arguments, the natural question arising is: how do the argument-based beliefs we have introduced in Formulae (5)-(7) behave with respect to standard principles of doxastic logic? In this final subsection we set out to provide a first answer to this question, and show that argument-based beliefs offer interesting insights into the failure of controversial principles such as, in particular, negative introspection.

All the above types of argument-based beliefs—viz., supported beliefs, endorsed beliefs and justified beliefs—obey the standard distribution and positive introspection principles, while they fail the negative introspection principle.

Proposition 4. *The following formulae are valid on* $\mathfrak{D} \times \mathfrak{E}$*:*

$$SB(\varphi \to \psi) \to (SB\varphi \to SB\psi) \quad (8)$$
$$EB(\varphi \to \psi) \to (EB\varphi \to EB\psi) \quad (9)$$
$$JB(\varphi \to \psi, \chi) \to (JB(\varphi, \chi) \to JB(\psi, \chi)) \quad (10)$$
$$SB\varphi \to SBSB\varphi \quad (11)$$
$$EB\varphi \to EBEB\varphi \quad (12)$$
$$JB(\varphi, \psi) \to JB(JB(\varphi, \psi), \psi) \quad (13)$$

Proposition 5 (Negative introspection failure). *The following formulae are* not *valid on* $\mathfrak{D} \times \mathfrak{E}$*:*

$$\neg SB\varphi \to SB\neg SB\varphi \quad (14)$$
$$\neg EB\varphi \to EB\neg EB\varphi \quad (15)$$
$$\neg JB(\varphi, \psi) \to JB(\neg JB(\varphi, \psi), \psi) \quad (16)$$

A simple inspection of the proof (in the appendix) shows that negative introspection fails only in one specific case, namely when it is not the case that all doxastic states are supported by some argument (resp., supported by endorsed arguments, or supported by endorsed arguments with property ψ). That is, negative introspection for argument-based beliefs does not go through only when the to-be-introspected belief itself is not supported by arguments.

This suggests that, although negative introspection fails in its full form, it still holds in a restricted one:

Proposition 6. *The following formulae are valid on* $\mathfrak{D} \times \mathfrak{E}$*:*

$$(\neg SB\varphi \wedge \boxminus_{\mathsf{B}} \Diamond_{\mathsf{U}} \sigma) \to SB\neg SB\varphi \quad (17)$$
$$(\neg EB\varphi \wedge \boxminus_{\mathsf{B}} \Diamond_{\mathsf{E}} \sigma) \to EB\neg EB\varphi \quad (18)$$
$$(\neg JB\varphi \wedge \boxminus_{\mathsf{B}} \Diamond_{\mathsf{E}} (\sigma \wedge \boxminus_{\mathsf{U}} \psi)) \to JB\neg JB\varphi \quad (19)$$

In other words, if I do not believe (with justification) that φ, but my beliefs are justified then I believe (with justification) that I do not believe (with justification) that φ.

7 Concluding remarks

Summary We proposed an approach based on product logics to study the interaction between the information state of an agent and the arguments the agent endorses. The approach has been illustrated by two product logics, which have been studied from a logical point of view, and have been used to model simple scenarios and formalize a number of interaction properties between beliefs and arguments.

Justification logic A related framework to the one we presented is *justification logic* (Artemov 2008), which formalizes the logic of statements of the type "t is a justification for φ", where t is called a, possibly complex, term. Justification logic provides an articulated solution to the logical omniscience problem of standard epistemic logic. In our framework arguments can well be seen as (atomic) terms but, unlike in justification logic, they are reified in the semantics of the logic and obtain the same 'ontological' status as doxastic states. While on the one hand this reintroduces logical omniscience, on the other hand it enables a classical modal approach for the study of (atomic) justifications viewed as arguments, which was our primary aim with this paper. Whether the reification of arguments/justifications in the semantics can be reconciled with a solution to the logical omniscience problem is an interesting open question.

Future work We point to three more research directions we consider especially worth pursuing. First, in our analysis we have assumed a primitive relation of 'support' between doxastic alternatives and arguments. This relation enabled us to say that a set of states (e.g., the truth-set of a column formula) was supported by a set of arguments whenever each state had an argument which was ground for it. Hence an argument can support a set of states only if it supports each alternative taken in isolation. This is clearly a strong requirement and is in part related to the logical omniscience issue just mentioned. Arguments might be thought of supporting sets of states without needing to support any of the states in the set taken in isolation. How to achieve this weakening while remaining within the set up of product logics is an interesting open question.

Second, we believe that studying our proposal in a multi-agent context would be another prominent venue for further research. For the beliefs, this would require that the alternatives for agents are modelled through a relation, rather than just identifying one set of states that are deemed doxastically possible. For the dimension representing the arguments, there is a range of possibilities one can consider. Are the arguments commonly known by all agents? Note that in our set-up, it is valid that $\blacklozenge \boxminus_{\mathsf{U}} \varphi \leftrightarrow \boxminus_{\mathsf{B}} \blacklozenge \boxminus_{\mathsf{U}} \varphi$ (with $\blacklozenge := \Diamond_{\mathsf{U}} \Diamond_{\mathsf{U}}$): there is an argument with row property φ if and only if the agent believes that. For the single agent-case this seems reasonable, but when reasoning about other agents' beliefs and justifications, there may be reasons to assume (we order them from weak to stronger) that (1) not all agents endorse the same arguments; (2) not all agents agree on the attack relation between arguments, and (3) not all agents need to be aware of the same arguments.[7] Once one agrees on some of those assumptions, one also needs to take into account that agents make such assumptions about each other (e.g., agent a believes that b's beliefs are justified by b's arguments, but not by arguments that a endorses).

Third, the paper has started from the simplest possible doxastic logic, KD45. Incorporating more sophisticated logics of belief constitutes another natural avenue of research. Apart from our interest to look at ways to cover the multi-agent case, other options that present themselves are incorporating richer order-theoretic models for doxastic logic (Baltag and Smets 2008); and studying the dyanmics supported by the horizontal and vertical dimensions with tools from dynamic epistemic logic (van Ditmarsch *et al.* 2007).

A Appendix: proofs

Proof of Theorem 1. Let $\mathfrak{F}$ the class of transitive and euclidian frames and $\mathfrak{F}_g$ the class of point-generated transitive and euclidean frames. We know that $\mathfrak{F}$ and $\mathfrak{F}_g$ are modally equivalent (Blackburn *et al.* 2001). [LEFT TO RIGHT] Assume $\mathfrak{D} \vDash \varphi$. Let $\mathcal{F}_s = \langle S_s, R_s \rangle \in \mathfrak{F}_g$ and $s \in S_s$. Define $B = \{x \mid sR_sx\}$ and notice that $\mathcal{D}_s = \langle S_s, B \rangle \in \mathfrak{D}$ and hence $\mathcal{D}_s \vDash \varphi$. An easy induction shows that $\mathcal{D}_s$ and $\mathcal{F}_s$ are modally equivalent and thus $\mathfrak{F} \vDash \varphi$ [RIGHT TO LEFT] Assume $\mathfrak{F} \vDash \varphi$. Let $\mathcal{D} = \langle S, B \rangle \in \mathfrak{D}$. Define a relation R on S

[7] Cf. (Schwarzentruber *et al.* 2012).

by xRy iff $y \in B$. It is easy to see that R is transitive and euclidean and hence $\mathcal{F} = \langle S, R\rangle \in \mathfrak{F}$. An easy induction proves that $\mathcal{D}$ and $\mathcal{F}$ are modally equivalent, entailing $\mathfrak{D} \vDash \varphi$. □

Proof sketch of Theorem 2. The result follows from Lemmata 1 and 2 by (Gabbay and Shehtman 1998, Theorem 7.12) (cf. our comment on completeness of $\mathsf{KD45} \times \mathsf{K}$). □

Proof sketch of Theorem 3. We exhibit a satisfiable formula from $\mathcal{L}^E$ which cannot be satisfied on a finite product model. Consider the following formula ψ from $\mathcal{L}^E$:

$$\Box_{\mathsf{U}} \Diamond\!\!\!-_{\mathsf{B}} p \wedge \Box_{\mathsf{U}} \boxminus_{\mathsf{B}} (p \rightarrow \Diamond_{\mathsf{A}} \neg p) \wedge \Box_{\mathsf{U}} \boxminus_{\mathsf{B}} (\neg p \rightarrow \Box_{\mathsf{A}} \neg p).$$

Let $\mathcal{E} = \langle A, E, \rightarrow\rangle$ be such that $\langle A, \rightarrow\rangle$ consists of an infinite $\leftarrow$-ascending chain of elements $\langle x_n \mid 0 \leq n < \omega\rangle$ such that $a_i \not\rightarrow a_j$ for $i < j$. For any choice of E, this is clearly a frame in $\mathfrak{E}$. Let then $\mathcal{D}$ consist of a point s_0 accessing all other states (except itself) in the frame. Take an enumeration $s_0, s_1, \ldots$ of these states. This is clearly a frame in $\mathfrak{D}$. Set $\mathcal{V}(p) = \{(s_{n+1}, a_n) \mid 0 \leq n < \omega\}$. We have $\langle \mathcal{D} \times \mathcal{E}, \mathcal{V}\rangle, (s_0, a_0) \vDash \psi$. Clearly, no model on a finite frame in $\mathfrak{D} \times \mathfrak{E}$ satisfies ψ.[8] □

Proof sketch of Lemma 1. Soundness is straightforward. As to completeness, the axiom system can be shown to be complete with respect to the class of frames $\mathcal{F} = \langle S, R_{\mathsf{B}}, R_{\mathsf{U}}\rangle$ consisting of one equivalence relation R_{U} (axiomatized by S5) which, within each of its equivalence classes, contains (containment is enforced by $\mathbf{Inc}_{\mathsf{BU}}$) a transitive and euclidean relation R_{B} (axiomatized by KD45) with the additional property (enforced by axioms $\mathbf{4}_{\mathsf{BU}}$ and $\mathbf{5}_{\mathsf{BU}}$) that within each equivalence class all states have access to the same set of states: $\forall x, y, z$ (in each equivalence class): $xR_{\mathsf{B}}y$ iff $zR_{\mathsf{B}}y$ (in other words, there exists a set of states B all of which elements are R_{B}-accessible by all states in the class, cf. Definition 1). The latter property is a consequence of the fact that R_{B} is a subrelation of R_{U} (by $\mathbf{Inc}_{\mathsf{BU}}$) and that axioms $\mathbf{4}_{\mathsf{BU}}$ and $\mathbf{5}_{\mathsf{BU}}$—which, notice, are Sahlqvist—correspond to the following properties: $\forall x, y, z$ if $xR_{\mathsf{U}}y$ and $yR_{\mathsf{B}}z$ then $xR_{\mathsf{B}}z$; $\forall x, y, z$ if $xR_{\mathsf{U}}y$ and $xR_{\mathsf{B}}z$ then $yR_{\mathsf{B}}z$. Call now this class $\mathfrak{F}$ and consider the class $\mathfrak{F}_g$ of frames in $\mathfrak{F}$ which are point-generated by the equivalence relation R_{U}. Any $\mathcal{F} \in \mathfrak{F}_g$ is thus such that R_{U} is the universal relation on $\mathcal{F}$ and $\mathcal{F}$ contains one unique set B R_{B}-accessible by all elements of the frame. $\mathcal{F}$ is therefore modally equivalent (on the horizontal fragment of $\mathcal{L}^E$) to a $\mathcal{D} \in \mathfrak{D}$. Vice versa, any $\mathcal{D} \in \mathfrak{D}$ is modally equivalent to a $\mathcal{F} \in \mathfrak{F}_g$ (cf. proof of Theorem 1). So $\mathfrak{D}$ is modally equivalent to $\mathfrak{F}_g$ which is, by general results (Blackburn *et al.* 2001), modally equivalent to $\mathfrak{F}$. Therefore, the axiom system is complete for $\mathfrak{D}$. □

Proof sketch of Theorem 4. We provide a reduction of DA^H to S5 and of DA^V to K^{U}, proving that the satisfiability of DA is reducible to the satisfiability of $\mathsf{S5} \times \mathsf{K}^{\mathsf{U}}$, which is decidable (Gabbay *et al.* 2003, Theorem 6.58). [DA^H to S5]. Take a fresh atom p and define a translation (Boolean clauses omitted): $t(\Diamond\!\!\!-_{\mathsf{U}}\varphi) = \Diamond t(\varphi)$; $t(\Diamond\!\!\!-_{\mathsf{B}}\varphi) = \Diamond(p \wedge t(\varphi))$. We prove: φ is satisfiable in $\mathfrak{D}$ iff $t(\varphi)$ is satisfiable in the class

[8] The proof goes through also for the weaker logic $\mathsf{KD45} \times \mathsf{K}^{\mathsf{U}}$.

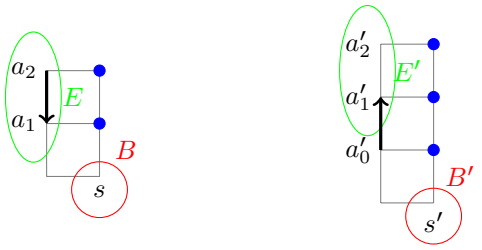


Figure 3: Two frames $\mathcal{F}$ and $\mathcal{F}'$ for the proof of Theorem 5.

of universal frames. [LEFT TO RIGHT] Assume $\mathcal{M}, x \vDash \varphi$. Build $\mathcal{M}' = \langle S', \mathcal{V}'\rangle$ with an extra atom p where: $S' = S$ and $\mathcal{V}' = \mathcal{V} \cup \{\langle p, S\rangle\}$. A simple induction shows that: $\mathcal{M}, x \vDash \varphi$ iff $\mathcal{M}', x \vDash t(\varphi)$. [RIGHT TO LEFT] Assume $\mathcal{M}, x \vDash t(\varphi)$. Build $\mathcal{M}' = \langle S', B, \mathcal{V}'\rangle$ where: $S' = S$, $B = \mathcal{V}(p)$ and $\mathcal{V}' = \mathcal{V} - \{\langle p, S\rangle\}$. A simple induction shows that: $\mathcal{M}, x \vDash t(\varphi)$ iff $\mathcal{M}' \vDash \varphi$. [$\mathsf{DA}^V$ to K^{U}] Similar. □

Proof sketch of Theorem 5. Consider the frames $\mathcal{F}$ and $\mathcal{F}'$ of Figure 3. We have that $\forall \psi \in \mathcal{L}^E$ $\mathcal{F}' \vDash \psi \Rightarrow \mathcal{F} \vDash \psi$, while at the same time, E' in $\mathcal{F}'$ is conflict free. □

Proof sketch of Proposition 4. Simple semantic arguments can be used to prove the claims. We do it for Formulae (8) and (13). [Formula (8)] Assume $\mathcal{M}, (s, a) \vDash \boxminus_{\mathsf{B}}(\Box_{\mathsf{U}}(\varphi \rightarrow \psi) \wedge \Diamond_{\mathsf{U}}\sigma)$. It follows, by the semantics of $\boxminus_{\mathsf{B}}$ and $\Box_{\mathsf{U}}$ that $\mathcal{M}, (s, a) \vDash \boxminus_{\mathsf{B}} \Box_{\mathsf{U}} (\varphi \rightarrow \psi) \wedge \boxminus_{\mathsf{B}} \Diamond_{\mathsf{U}} \sigma$ and then, by propositional reasoning, $\mathcal{M}, (s, a) \vDash \boxminus_{\mathsf{B}}(\Box_{\mathsf{U}}\varphi \wedge \Diamond_{\mathsf{U}}\sigma) \rightarrow \boxminus_{\mathsf{B}}(\Box_{\mathsf{U}}\psi \wedge \Diamond_{\mathsf{U}}\sigma)$, which proves the claim. [Formula (13)] Assume $\mathcal{M}, (s, a) \vDash \boxminus_{\mathsf{B}}(\Box_{\mathsf{U}}\varphi \wedge \Diamond_{\mathsf{E}}(\sigma \wedge \boxminus_{\mathsf{U}}\psi))$. It follows, by the semantics of $\boxminus_{\mathsf{B}}$ and $\Box_{\mathsf{U}}$ and propositional reasoning, that $\mathcal{M}, (s, a) \vDash \Box_{\mathsf{U}} \boxminus_{\mathsf{B}} (\Box_{\mathsf{U}}\varphi \wedge \Diamond_{\mathsf{E}}(\sigma \wedge \boxminus_{\mathsf{U}}\psi)) \wedge \boxminus_{\mathsf{B}} \Diamond_{\mathsf{E}} (\sigma \wedge \boxminus_{\mathsf{U}}\psi))$. From this, by again the semantics of $\boxminus_{\mathsf{B}}$ we conclude $\mathcal{M}, (s, a) \vDash \boxminus_{\mathsf{B}}(\Box_{\mathsf{U}}(\boxminus_{\mathsf{B}}(\Box_{\mathsf{U}}\varphi \wedge \Diamond_{\mathsf{E}}(\sigma \wedge \boxminus_{\mathsf{U}}\psi))) \wedge \Diamond_{\mathsf{E}}(\sigma \wedge \boxminus_{\mathsf{U}}\psi))$, which proves the claim. □

Proof sketch of Proposition 5. It suffices to exhibit some counterexamples. We do that for Formula (14). Consider the model $\mathcal{M} = \langle\langle \{s\}, \{s\}\rangle \times \langle \{a\}, \varnothing, \varnothing\rangle, \mathcal{V}\rangle$ consisting of one pair (s, a) s.t. $(s, a) \in \mathcal{V}(p)$ and $(s, a) \notin \mathcal{V}(\sigma)$. We have that $\mathcal{M}, (s, a) \vDash \neg \boxminus_{\mathsf{B}} (\Box_{\mathsf{U}}\varphi \wedge \Diamond_{\mathsf{U}}\sigma)$ and $\mathcal{M}, (s, a) \nvDash \boxminus_{\mathsf{B}}(\Box_{\mathsf{U}}(\neg \boxminus_{\mathsf{B}} (\Box_{\mathsf{U}}\varphi \wedge \Diamond_{\mathsf{U}}\sigma)) \wedge \Diamond_{\mathsf{U}}\sigma)$. □

Proof sketch of Proposition 6. We prove the claim of Formula (17). The other two are analogous. Assume $\mathcal{M}, (s, a) \vDash \neg SB\varphi \wedge \boxminus_{\mathsf{B}} \Diamond_{\mathsf{U}} \sigma$. It follows by Formula (5) that $\mathcal{M}, (s, a) \vDash \neg \boxminus_{\mathsf{B}} \Box_{\mathsf{U}}\varphi \wedge \boxminus_{\mathsf{B}} \Diamond_{\mathsf{U}} \sigma$, from which by the semantics of $\boxminus_{\mathsf{B}}$ we obtain $\mathcal{M}, (s, a) \vDash \boxminus_{\mathsf{B}}(\Box_{\mathsf{U}}(\neg \boxminus_{\mathsf{B}} \Box_{\mathsf{U}}\varphi \wedge \Diamond_{\mathsf{U}}\sigma) \wedge \boxminus_{\mathsf{B}} \Diamond_{\mathsf{U}} \sigma)$. Hence $\mathcal{M}, (s, a) \vDash SB\neg SB\varphi$. □

B Acknowledgments

The authors wish to thank Wesley Holliday for useful comments on an earlier draft of this paper.

References

S. Artemov. The logic of justification. *The Review of Symbolic Logic*, 1(4):477–513, 2008.

A. Baltag and S. Smets. A qualitative theory of dynamic interactive belief revision. In G. Bonanno, W. Van der Hoek, and M. Wooldridge, editors, *Logic and the Foundation of Game and Decision Theory—LOFT 07*, volume 3 of *Texts in Logic and Games*, pages 11–58. Amsterdam University Press, 2008.

P. Baroni and M. Giacomin. Semantics of abstract argument systems. In I. Rahwan and G. R. Simari, editors, *Argumentation in Artifical Intelligence*. Springer, 2009.

P. Baroni, M. Caminada, and M. Giacomin. An introduction to argumentation semantics. *The Knowledge Engineering Review*, 26(4):365–410, 2011.

P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, Cambridge, 2001.

B. F. Chellas. *Modal Logic. An Introduction*. Cambridge University Press, Cambridge, 1980.

P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about knowledge*. The MIT Press, Cambridge, 1995.

D. Gabbay and Shehtman. Products of modal logics, part 1. *Logic Journal of the IGPL*, 6(1):73–146, 1998.

D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-dimensional modal logics. Theory and applications*. Elsevier, 2003.

D. Grossi. On the logic of argumentation theory. In W. van der Hoek, G. Kaminka, Y. Lespérance, and S. Sen, editors, *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 409–416. IFAAMAS, 2010.

D. Grossi. Fixpoints and iterated updates in abstract argumentation. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*, 2012.

M. Marx. Complexity of products of modal logics. *Journal of Logic and Computation*, 9(2):197–214, 1999.

J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*, volume 41 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.

S. Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173(9-10):901–934, 2009.

K. Sano. Axiomatizing hybrid products of monotone neighborhood frames. *Electronic Notes in Theoretical Computer Science*, 273:51–67, 2011.

F. Schwarzentruber, S. Vesic, and T. Reinstra. Building an epistemic logic for argumentation. In *Proceedings of JELIA'12*, pages 359–371, 2012.

J. van Benthem and E. Pacuit. Dynamic logics of evidence-based beliefs. *Studia Logica*, 99(1):61–92, 2011.

J. van Benthem. *Modal Logic and Classical Logic*. Monographs in Philosophical Logic and Formal Linguistics. Bibliopolis, 1983.

H. van Ditmarsch, B. Kooi, and W. van der Hoek. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library Series*. Springer, 2007.

Belief Change Operations: A Short History of Nearly Everything, Told in Dynamic Logic of Propositional Assignments

Andreas Herzig
Université de Toulouse, CNRS, IRIT
http://www.irit.fr/~Andreas.Herzig

Abstract

We examine several belief change operations in the light of Dynamic Logic of Propositional Assignments DL-PA. We show that we can encode in a systematic way update operations (such as Winslett's 'Possible Models Approach') and revision operations (such as Dalal's) as particular DL-PA programs. Every DL-PA formula being equivalent to a boolean formula, we obtain syntactical counterparts for all these belief change operations.

Introduction

When a belief base B is modified in order to take into account a new piece of information A then we want to build a new belief base $B \circ A$. In that enterprise at least two reasoning problems are interesting.

- The *belief change problem* is a function problem: find a logical formula representing the modified base $B \circ A$.
- The *belief change consequence problem* (called implication problem in (Eiter and Gottlob 1992)) is a decision problem: for a given formula C, decide whether C is a consequence of $B \circ A$, i.e., decide whether the implication $(B \circ A) \rightarrow C$ is valid.

These problems are omnipresent in computer science, e.g. in databases, reasoning about actions, planning, logic programming, and in semantic web ontologies.

Many belief change operations can be found in the literature. Among the most prominent are Dalal's revision operation (Dalal 1988), Winslett's 'Possible Models Approach' (PMA) (Winslett 1988; 1990), Forbus's update operation (Forbus 1989), and Winslett's Standard Semantics (WSS) (Winslett 1995). We refer to (Herzig and Rifi 1999) for a detailed overview and comparison. According to Katsuno and Mendelzon's classical distinction between update and revision operations, the first three are update operations, while Dalal's is a revision operation (Katsuno and Mendelzon 1992). A recent discussion of the distinction between the two concepts and their relationship can be found in (Lang 2007).

Beyond the above concrete belief change operations, there exist more abstract frameworks. Alchourrón, Gärdenfors and Makinson (AGM) studied the properties that 'good' revision operations should have (Alchourrón, Gärdenfors, and Makinson 1985). Similarly, Katsuno and Mendelzon (KM) studied the properties 'good' update operations should have (Katsuno and Mendelzon 1992). The difference between revision and update can be illustrated by the example of a new edition of a dictionary that is advertised as being 'updated and revised': revision corrects information that turned out to be wrong (about a static world), while update takes into account new usages of words, i.e., the dynamics of the world.

Dalal's operation satisfies the AGM postulates for revision, and Winslett's and Forbus's operations satisfy the KM postulates for update. This is however not enough to *characterise* these operations: they validate properties beyond the AGM/KM postulates, such as Parikh's relevance postulate that is based on language splitting (Parikh 1999; Kourousias and Makinson 2007). Only few papers investigated this issue. Similarly, the related problem of syntactically constructing the new belief base $B \circ A$ has only received little attention. Indeed, the above concrete belief change operations were almost exclusively studied from a semantical perspective: instead of viewing $B \circ A$ as a formula, it is viewed as a set of models of classical propositional logic, alias a set of valuations. Therefore $\circ$ is not part of the object language but is in the metalanguage. We call $\circ$ a metalanguage operat*ion*, as opposed to object language operat*or*, i.e., a logical connective.

If the set of propositional variables $\mathbb{P}$ is finite then there is an easy recipe to construct a formula representing $B \circ A$: describe each valuation $v \in B \circ A$ by the conjunction of literals $\mathsf{Fml}(v) = \left(\bigwedge_{p \in v} p \right) \wedge \left(\bigwedge_{p \in \mathbb{P} \setminus v} \neg p \right)$ and take the disjunction of these big conjunctions $\mathsf{Fml}(v)$: the set of models of $\bigvee_{v \in B \circ A} \mathsf{Fml}(v)$ equals $B \circ A$, and therefore the former is the syntactical counterpart of the latter.

Is there a better, syntactic procedure building the new belief base? The answer is positive for WSS, which is the simplest update operation: an update procedure based on variable forgetting is in (Herzig and Rifi 1999) and its generalisation to literal forgetting is in (Herzig, Lang, and Marquis 2013). As to the PMA, there is an axiomatisation in (Herzig 1996) that can be turned into a decision procedure for the belief change consequence problem; it is however based on formulas in disjunctive normal form and is as such very close to the semantics. Finally, there

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

are syntactical belief change operations that are based on the computation of prime implicants and prime implicates and do not have semantics (Bienvenu, Herzig, and Qi 2008; Marchi, Bittencourt, and Perrussel 2010).

In this paper we propose a powerful yet simple logical framework for belief change: Dynamic Logic of Propositional Assignments, abbreviated DL-PA (Herzig et al. 2011; Balbiani, Herzig, and Troquard 2013). DL-PA is a simple instantiation of Propositional Dynamic Logic PDL (Harel 1984; Harel, Kozen, and Tiuryn 2000): instead of PDL's abstract atomic programs, its atomic programs are assignments of propositional variables to either true or false, written $p{\leftarrow}\top$ and $p{\leftarrow}\bot$. Just as in PDL, these atomic programs can be combined by means of program operators: sequential and nondeterministic composition, finite iteration, and test. In the present paper we moreover make use of a program operator that is less frequently considered in PDL, namely the converse operator. While DL-PA programs describe the evolution of the world, DL-PA formulas describe the state of the world. In particular, formulas of the form $\langle\pi\rangle\varphi$ express that φ is true after *some* possible execution of π, and $[\pi]\varphi$ expresses that φ is true after *every* possible execution of π.

The models of DL-PA are considerably simpler than PDL's Kripke models: valuations of classical propositional logic are enough. The assignment $p{\leftarrow}\top$ updates the current valuation by p, while the assignment $p{\leftarrow}\bot$ updates it by $\neg p$. For example, the program $\mathsf{emp}_{\mathsf{Ann,CS}}{\leftarrow}\bot$ updates the current valuation by the fact that Ann is not an employee of the CS department. The other way round, the program $(\mathsf{emp}_{\mathsf{Ann,CS}}{\leftarrow}\bot)^-$ undoes that update. Consider the formula

$$\langle \mathsf{emp}_{\mathsf{Ann,CS}}{\leftarrow}\bot^-\rangle(\mathsf{emp}_{\mathsf{Ann,CS}} \wedge \mathsf{emp}_{\mathsf{Bob,CS}})$$

It says that possibly, before Ann was fired from the CS department, both she and Bob were employees there. This means that it is possible that the belief base $\mathsf{emp}_{\mathsf{Ann,CS}} \wedge \mathsf{emp}_{\mathsf{Bob,CS}}$ has been updated by $\neg\mathsf{emp}_{\mathsf{Ann,CS}}$, resulting in the current belief base. The current base is therefore nothing but

$$(\mathsf{emp}_{\mathsf{Ann,CS}} \wedge \mathsf{emp}_{\mathsf{Bob,CS}}) \circ \neg\mathsf{emp}_{\mathsf{Ann,CS}}$$

This can be generalised: we have

$$B \circ p = \|\langle (p{\leftarrow}\top)^-\rangle B\|$$
$$B \circ \neg p = \|\langle (p{\leftarrow}\bot)^-\rangle B\|$$

for every propositional variable p and each of the update operations among PMA, WSS, and Forbus, where the function $\|.\|$ associates to each formula the set of valuations where it is true. (The corresponding equality for Dalal's revision operation is a bit more involved and will be given later.)

The first contribution of the present paper is to go beyond such updates by literals and to associate to every input formula A a program π_A implementing the update by A. Precisely, these programs π_A depend on the belief change operation under concern, and WSS, PMA and Forbus's operation will be implemented by different programs π_A^{wss}, π_A^{forbus}, and π_A^{pma}. For instance, we will show that the WSS update by the disjunction $p{\vee}q$ is captured by the program

$$\pi^{\mathsf{wss}}_{p\vee q} = (p{\leftarrow}\top \cup p{\leftarrow}\bot); (q{\leftarrow}\top \cup q{\leftarrow}\bot); p{\vee}q?$$

which is equivalent to

$$(p{\leftarrow}\top; q{\leftarrow}\top) \cup (p{\leftarrow}\top; q{\leftarrow}\bot) \cup (p{\leftarrow}\bot; q{\leftarrow}\top).$$

We establish that being in a state where B was updated by A is the same as being in a state where B was true before the update program π_A was executed. Formally, we prove that

$$B \circ A \;=\; \|\langle(\pi_A)^-\rangle B\|$$

for each of the belief change operations $\circ$ and associated family of programs π_A that we consider: WSS, PMA, Forbus, and Dalal.[1]

It is shown in (Herzig et al. 2011; Balbiani, Herzig, and Troquard 2013) that every DL-PA formula can be reduced to an equivalent propositional formula. It is shown in the latter paper that the reduction can be extended to the converse operator. We thereby obtain a syntactical representation of the modified belief base in terms of a boolean formula. This is the second contribution of our paper.

In the next two sections we settle some notations and introduce DL-PA. In the remaining sections we embed several belief change semantics into DL-PA: Winslett's WSS, Winslett's PMA, and Forbus's update operation, and finally Dalal's belief revision operation.

Background: propositional logic

Here are some notations and conventions for propositional logic, as well as the definitions of symmetric difference and Hamming distance between propositional valuations.

A *valuation* associates a truth value to each propositional variable in the countable set $\mathbb{P} = \{p, q, \ldots\}$. We identify valuations with subsets of $\mathbb{P}$ and use v, v_1, v_2, etc. to denote them. The set of all valuations is $\mathbb{V} = 2^{\mathbb{P}}$. Sometimes we write $v(p) = 1$ when $p \in v$ and $v(p) = 0$ when $p \notin v$.

The *symmetric difference* between two valuations v_1 and v_2 is the set of all those p such that $v_1(p) \neq v_2(p)$. Formally, it is defined as

$$v_1 \dot{-} v_2 = (v_1 \setminus v_2) \cup (v_2 \setminus v_1)$$

For example, $\{p, q\} \dot{-} \{q, r, s\} = \{p, r, s\}$.

The *Hamming distance* between v_1 and v_2 is is the number of all those p such that $v_1(p) \neq v_2(p)$. Formally, it is defined to be $\mathsf{card}(v_1 \dot{-} v_2)$, i.e., the cardinality of the symmetric difference between v_1 and v_2. For example, the Hamming distance between $\{p, q\}$ and $\{q, r, s\}$ is $\mathsf{card}(\{p, r, s\}) = 3$. Note that the Hamming distance might be infinite: for instance, $\mathsf{card}(\emptyset \dot{-} \mathbb{P}) = \infty$.

Boolean formulas (also called *propositional formulas*) are built from propositional variables by means of the standard boolean connectives. We will in particular make use of the exclusive disjunction, denoted by $\oplus$. Boolean formulas are denoted by A, B, C, etc. Contrasting with that, *modal formulas*—to be defined in the next section—will be denoted by φ, ψ, etc. For a given boolean formula A, the set of variables occurring in A is denoted by $\mathbb{P}_A$. For example, $\mathbb{P}_{p\vee\neg q} = \{p, q\}$.

A given valuation determines the truth value of each boolean formula. An *A-valuation* is a valuation where the boolean formula A is true.

[1] In the case of Dalal's revision operation, the program actually also depends on the base B. This is as it should be, because AGM revision operations involve checking whether $A{\wedge}B$ is satisfiable.

Dynamic Logic of Propositional Assignments

In this section we define syntax and semantics of Dynamic Logic of Propositional Assignments DL-PA and state decidability and complexity results. Based on earlier work by Tiomkin and Makowski and by van Eijck (Tiomkin and Makowsky 1985; van Eijck 2000), star-free DL-PA was studied in (Herzig et al. 2011) and full DL-PA in (Balbiani, Herzig, and Troquard 2013).

Language The language of DL-PA is defined by the following grammar:

$$\begin{array}{lll} \varphi & ::= & p \mid \top \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi \\ \pi & ::= & p{\leftarrow}\top \mid p{\leftarrow}\bot \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \pi^- \mid \varphi? \end{array}$$

where p ranges over $\mathbb{P}$. So the *atomic programs* of the language of DL-PA are of the form $p{\leftarrow}\top$ and $p{\leftarrow}\bot$. The operators of sequential composition (";"), nondeterministic composition ("$\cup$"), finite iteration ("$(.)^*$", the so-called Kleene star), and test ("$(.)?$") are familiar from PDL. The operator "$(.)^-$" is the converse operator. The *star-free fragment* of DL-PA is the subset of the language made up of formulas without the Kleene star "$(.)^*$".

The set of variables occurring in φ is denoted by $\mathbb{P}_\varphi$, just as for boolean formulas. The *length* of a formula φ, denoted by $|\varphi|$, is the number of symbols used to write down φ, without "$\langle$", "$\rangle$", and parentheses. For example, $|\langle q{\leftarrow}\top\rangle(q \vee r)| = 3{+}3 = 6$. The length of a program π, denoted by $|\pi|$, is defined in the same way. For example, $|p{\leftarrow}\bot; p?| = 6$. We have $\mathsf{card}(\mathbb{P}_\varphi) \leq |\varphi|$ for every φ.

We abbreviate the logical connectives $\wedge$, $\rightarrow$, $\leftrightarrow$, and $\oplus$ in the usual way. Moreover, $[\pi]\varphi$ abbreviates $\neg\langle\pi\rangle\neg\varphi$. Several program abbreviations are familiar from PDL. First, skip abbreviates $\top?$ ("nothing happens"). Second, the conditional "if φ then π_1 else π_2" is expressed by $(\varphi?;\pi_1) \cup (\neg\varphi?;\pi_2)$. Third, the loop "while φ do π" is expressed by $(\varphi?;\pi)^*;\neg\varphi?$. Let us moreover recursively define the n-th iteration of π and the iteration up to n of π, for $n \geq 0$:

$$\pi^n = \begin{cases} \mathsf{skip} & \text{if } n = 0 \\ \pi;\pi^{n-1} & \text{if } n \geq 1 \end{cases}$$

$$\pi^{\leq n} = \begin{cases} \mathsf{skip} & \text{if } n = 0 \\ (\mathsf{skip} \cup \pi);\pi^{\leq n-1} & \text{if } n \geq 1 \end{cases}$$

Finally, we introduce assignments of literals to variables by means of the following two abbreviations:

$$\begin{array}{rl} p{\leftarrow}q = & \text{if } q \text{ then } p{\leftarrow}\top \text{ else } p{\leftarrow}\bot \\ = & (q?; p{\leftarrow}\top) \cup (\neg q?; p{\leftarrow}\bot) \\ p{\leftarrow}\neg q = & \text{if } q \text{ then } p{\leftarrow}\bot \text{ else } p{\leftarrow}\top \\ = & (q?; p{\leftarrow}\bot) \cup (\neg q?; p{\leftarrow}\top) \end{array}$$

The former assigns to p the truth value of q, while the latter assigns to p the truth value of $\neg q$. In particular, the program $p{\leftarrow}\neg p$ flips the truth value of p. Note that the abbreviations $p{\leftarrow}q$ and $p{\leftarrow}\neg q$ have constant length (namely 14).

Semantics DL-PA programs are interpreted by means of a (unique) *relation between valuations*. The atomic programs $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ update valuations in the obvious way, and complex programs are interpreted just as in PDL by mutual recursion. Table 1 gives the interpretation of the DL-PA connectives.

$$\begin{array}{rcl} \|p\| & = & \{v \ : \ p \in v\} \\ \|\top\| & = & \mathbb{V} \ = \ 2^{\mathbb{P}} \\ \|\bot\| & = & \emptyset \\ \|\neg\varphi\| & = & 2^{\mathbb{P}} \setminus \|\varphi\| \\ \|\varphi \vee \psi\| & = & \|\varphi\| \cup \|\psi\| \\ \|\langle\pi\rangle\varphi\| & = & \{v \ : \ \text{there is } v_1 \text{ s.t. } (v, v_1) \in \|\pi\| \text{ and } v_1 \in \|\varphi\|\} \\ \|p{\leftarrow}\top\| & = & \{(v_1, v_2) \ : \ v_2 = v_1 \cup \{p\}\} \\ \|p{\leftarrow}\bot\| & = & \{(v_1, v_2) \ : \ v_2 = v_1 \setminus \{p\}\} \\ \|\pi;\pi'\| & = & \|\pi\| \circ \|\pi'\| \\ \|\pi \cup \pi'\| & = & \|\pi\| \cup \|\pi'\| \\ \|\pi^*\| & = & \bigcup_{k \in \mathbb{N}_0} (\|\pi\|)^k \\ \|\pi^-\| & = & \|\pi\|^{-1} \\ \|\varphi?\| & = & \{(v, v) \ : \ v \in \|\varphi\|\} \end{array}$$

Table 1: Interpretation of the DL-PA connectives.

Two formulas φ_1 and φ_2 are *formula equivalent* if $\|\varphi_1\| = \|\varphi_2\|$. Two programs π_1 and π_2 are *program equivalent* if $\|\pi_1\| = \|\pi_2\|$. For example, $\mathsf{skip};\pi$ is program equivalent to π, and $\mathsf{skip} \cup p{\leftarrow}\neg p$ is program equivalent to $p{\leftarrow}\top \cup p{\leftarrow}\bot$. We write $\pi_1 \equiv \pi_2$ when π_1 and π_2 are progam equivalent.

Proposition 1 (Proposition 10 in (Balbiani, Herzig, and Troquard 2013)). *The following program equivalences are* DL-PA *valid:*

$$\begin{array}{rll} p?; p{\leftarrow}\top & \equiv & p? \\ \neg p?; p{\leftarrow}\bot & \equiv & \neg p? \\ p{\leftarrow}\tau; p{\leftarrow}\top & \equiv & p{\leftarrow}\top \\ p{\leftarrow}\tau; p{\leftarrow}\bot & \equiv & p{\leftarrow}\bot \\ p{\leftarrow}\tau; q{\leftarrow}\top & \equiv & q{\leftarrow}\top; p{\leftarrow}\tau \qquad \textit{if } p \neq q \\ p{\leftarrow}\tau; q{\leftarrow}\bot & \equiv & q{\leftarrow}\bot; p{\leftarrow}\tau \qquad \textit{if } p \neq q \end{array}$$

where τ is a placeholder for either true or false.

An *expression* is a formula or a program. When we say that two expressions are equivalent we mean program equivalence if we are talking about programs, and formula equivalence otherwise. Equivalence is preserved under replacement of a sub-expression by an equivalent expression (Balbiani, Herzig, and Troquard 2013, Proposition 7).

A formula φ is DL-PA *valid* if it is formula equivalent to $\top$, i.e., if $\|\varphi\| = 2^{\mathbb{P}}$. It is DL-PA *satisfiable* if it is not formula equivalent to $\bot$, i.e., if $\|\varphi\| \neq \emptyset$. For example, the formulas $\langle p{\leftarrow}\bot\rangle\top$ and $\langle p{\leftarrow}\top\rangle\varphi \leftrightarrow \neg\langle p{\leftarrow}\top\rangle\neg\varphi$ are DL-PA valid. Other examples of DL-PA validities are $\langle p{\leftarrow}\top\rangle p$ and $\langle p{\leftarrow}\bot\rangle\neg p$. The valid schemas $\varphi \rightarrow [\pi]\langle\pi^-\rangle\varphi$ and $\varphi \rightarrow [\pi^-]\langle\pi\rangle\varphi$ are inherited from converse PDL; they are called the *conversion axioms*. Moreover, $\varphi \rightarrow [\pi]\varphi$ is valid if and only if $\langle\pi^-\rangle\varphi \rightarrow \varphi$ is valid. The two senses of the "if and only if" correspond to the two so-called *conversion rules*.

$$
\begin{aligned}
p{\leftarrow}\top^- &\equiv p?;(\mathsf{skip} \cup p{\leftarrow}\bot)\\
&\equiv p? \cup (p?; p{\leftarrow}\bot)\\
p{\leftarrow}\bot^- &\equiv \neg p?;(\mathsf{skip} \cup p{\leftarrow}\top)\\
&\equiv \neg p? \cup (\neg p?; p{\leftarrow}\top)\\
(\pi_1;\pi_2)^- &\equiv \pi_2^-;\pi_1^-\\
(\pi_1 \cup \pi_2)^- &\equiv \pi_1^- \cup \pi_2^-\\
(\pi^*)^- &\equiv (\pi^-)^*\\
(\varphi?)^- &\equiv \varphi?
\end{aligned}
$$

Table 2: Reduction axioms for the converse operator.

Observe that if p does not occur in φ then both $\varphi \to \langle p{\leftarrow}\top\rangle\varphi$ and $\varphi \to \langle p{\leftarrow}\bot\rangle\varphi$ are valid. This is due to the following semantical property that is instrumental in the proof of several results in the rest of the paper.

Proposition 2. *Suppose $\mathbb{P}_\varphi \cap P = \emptyset$, i.e., none of the variables in P occurs in φ. Then $v \cup P \in \|\varphi\|$ iff $v \setminus P \in \|\varphi\|$.*

Reduction to propositional logic In PDL, all program operators but the Kleene star can be eliminated. In contrast, *all* program operators can be eliminated in DL-PA. We describe this in detail now.

The first step is to eliminate the converse operator. Table 2 lists the program equivalences by means of which we can eliminate the converse operator with only linear increase in program length.[2]

Proposition 3. *For every program π there is an equivalent π' such that no converse operator occurs in π' and such that the length of π' is linear in the length of π.*

For example, the converse of assignments of literals to variables are reduced as follows:

$$
p{\leftarrow}q^- \equiv \begin{cases} \mathsf{skip} & \text{if } q = p\\ p{\leftrightarrow}q?;(p{\leftarrow}\top \cup p{\leftarrow}\bot) & \text{otherwise}\end{cases}
$$

$$
p{\leftarrow}\neg q^- \equiv \begin{cases} p{\leftarrow}\neg p & \text{if } q = p\\ p{\oplus}q?;(p{\leftarrow}\top \cup p{\leftarrow}\bot) & \text{otherwise}\end{cases}
$$

The second step is to eliminate the Kleene star.

[2]This simplifies the original proof of (Balbiani, Herzig, and Troquard 2013, Section VII.B), which instead of directly stating a reduction axiom for the Kleene star rather relies on its preliminary elimination.

We might as well apply de Giacomo's general elimination technique(de Giacomo 1996). It is however more complicated than ours. He proceeds by first applying the four last reduction axioms of Table 2 to the original formula φ_0 in order to push down the converse operator. For the resulting formula φ_1, a theory Γ_{φ_1} is built that is made up of instances of the conversion axioms for all symbols occurring in φ and that are prefixed by a master modality relative to the symbols of φ. Finally, it is proved that the original φ_0 is a theorem if and only if $\Gamma \models \varphi_1'$, where φ_1' is obtained from φ_1 by considering the converse of atomic programs as new program names (so φ_1' is without converse operators). Overall, this is less direct and not as simple as the present reduction axioms.

$$
\begin{aligned}
\mathsf{flip}^1(P) &= p_1{\leftarrow}\neg p_1 \cup \cdots \cup p_n{\leftarrow}\neg p_n\\
\mathsf{flip}^{\geq 0}(P) &= (p_1{\leftarrow}\top \cup p_1{\leftarrow}\bot);\cdots;(p_n{\leftarrow}\top \cup p_n{\leftarrow}\bot)\\
\mathsf{store}(P) &= p_1'{\leftarrow}p_1;\cdots;p_n'{\leftarrow}p_n\\
\mathsf{restore}^1(P) &= (p_1{\oplus}p_1'?;p_1{\leftarrow}p_1') \cup \cdots \cup (p_n{\oplus}p_n'?;p_n{\leftarrow}p_n')\\
\mathsf{restore}^{\geq 0}(P) &= (\mathsf{skip} \cup p_1{\leftarrow}p_1');\cdots;(\mathsf{skip} \cup p_n{\leftarrow}p_n')
\end{aligned}
$$

Table 3: Some useful DL-PA programs, where $P = \{p_1,\ldots,p_n\}$. For $n = 0$ all programs are supposed to equal skip. In $\mathsf{store}(P)$, $\mathsf{restore}^1(P)$, and $\mathsf{restore}^{\geq 0}(P)$, the p_i' are supposed to be fresh.

Proposition 4 (Theorem 1 of (Balbiani, Herzig, and Troquard 2013)). *For every converse-free π there is an equivalent π' such that no Kleene star occurs in π'.*

The resulting formula is in star-free PDL, for which it is known that all the program operators can be eliminated. Once this has been done, modal operators only contain atomic programs. Such modal operators are both serial and deterministic and therefore distribute over negation and disjunction: $\langle p{\leftarrow}\bot\rangle\neg\varphi$ is equivalent to $\neg\langle p{\leftarrow}\bot\rangle\varphi$, and $\langle p{\leftarrow}\bot\rangle(\varphi_1 \vee \varphi_2)$ is equivalent to $\langle p{\leftarrow}\bot\rangle\varphi_1 \vee \langle p{\leftarrow}\bot\rangle\varphi_2$. They can finally be eliminated when they face a propositional variable, according to the following formula equivalences:

$$
\langle p{\leftarrow}\top\rangle q \leftrightarrow \begin{cases}\top & \text{if } q = p\\ q & \text{otherwise}\end{cases}
$$

$$
\langle p{\leftarrow}\bot\rangle q \leftrightarrow \begin{cases}\bot & \text{if } q = p\\ q & \text{otherwise}\end{cases}
$$

For example, $\langle p{\leftarrow}\bot\rangle p \vee \langle p{\leftarrow}\bot\rangle\neg r$ is equivalent to $\bot \vee \neg r$.

Together, these three steps make up a complete set of reduction axioms: for every formula there is an equivalent boolean formula.

Complexity In this paper we only need complexity results for star-free DL-PA. The complexity of both model checking and satisfiability checking for the converse-free and star-free fragment of the language of DL-PA is shown to be PSPACE complete in (Herzig et al. 2011). By Proposition 3 these results transfer to star-free DL-PA with the converse operator.

Some useful DL-PA programs Table 3 collects some DL-PA programs that are going to be convenient in our enterprise. In that table, P is the set of variables $\{p_1,\ldots,p_n\}$, and P' is the set of variables p_k' such that p_k is in P and p_k' is *fresh*: we suppose that p_k' does not occur in P (or rather, in the formula under consideration). In order to alleviate notation we drop set parentheses and write $\mathsf{flip}^1(p)$ instead of $\mathsf{flip}^1(\{p\})$, etc. Note that the length of each program of Table 3 is linear in the cardinality of the set P.

The first two programs of Table 3 change the truth values of some of the variables in the set P. The program $\mathsf{flip}^1(P)$ flips some nondeterministically chosen variable of P. So the iteration $\mathsf{flip}^1(P)^n$ does this n times and $\mathsf{flip}^1(P)^{\leq n}$ does this

at most n times. For example, we have

$$\begin{aligned}\mathsf{flip}^1(p)^1 &= p{\leftarrow}\neg p;\mathsf{skip}\\ &\equiv p{\leftarrow}\neg p\\ \mathsf{flip}^1(p)^{\leq 1} &= (\mathsf{skip}\cup p{\leftarrow}\neg p);\mathsf{skip}\\ &\equiv p{\leftarrow}\top\cup p{\leftarrow}\bot\\ &= \mathsf{flip}^{\geq 0}(p)\end{aligned}$$

The program $\mathsf{flip}^{\geq 0}(P)$ nondeterministically changes the truth value of some of the variables in P. It is program equivalent to $\mathsf{flip}^1(P)^{\leq n}$ while being shorter.[3] It actually implements the operation of *forgetting* the values of the variables in P (Lang, Liberatore, and Marquis 2003). Note that $\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi);\varphi?$ relates any valuation v to all φ-valuations where the variables outside φ have the same truth values.

The last three programs of Table 3 store or restore some of the values of the variables of P. The program $\mathsf{store}(P)$ stores the truth value of each variable p_k of P by means of a fresh variable p'_k. When the truth values of p_k and p'_k differ then we say that p_k is marked; else we say that p_k is unmarked. The program $\mathsf{restore}^1(P)$ restores one of the marked variables p_k of P: after it, some p_k has the truth value of p'_k. Finally, $\mathsf{restore}^{\geq 0}(P)$ restores zero or more marked variables of P.

Proposition 5. *The following hold:*

1. $(v_1, v_2) \in \|\mathsf{flip}^1(p)\|$ *iff* $\mathsf{card}(v_1 \dot{-} v_2) = 1$.
2. $(v_1, v_2) \in \|\mathsf{flip}^1(P)^{\leq n}\|$ *iff* $\mathsf{card}(v_1 \dot{-} v_2) \leq n$.
3. $(v_1, v_2) \in \|\mathsf{flip}^{\geq 0}(P)\|$ *iff* $v_1 \dot{-} v_2 \subseteq P$.
4. $(v_1, v_2) \in \|\mathsf{store}(P)\|$ *iff* $v_2 = (v_1 \setminus P') \cup \{p' \,:\, p \in v_1 \cap P\}$.
5. $(v_1, v_2) \in \|\mathsf{restore}^1(P)\|$ *iff there is* $p_k \in P$ *such that* $v_1 \dot{-} v_2 = \{p_k\}$ *and* $v_1(p_k) \neq v_1(p'_k)$.
6. $(v_1, v_2) \in \|\mathsf{restore}^{\geq 0}(P)\|$ *iff there is* $S \subseteq P$ *such that* $v_1 \dot{-} v_2 = S$ *and* $v_1(p_k) \neq v_1(p'_k)$ *for every* $p_k \in S$.

Note that $\mathsf{flip}^1(P)^n$ may flip less than exactly n variables: for example, $\mathsf{flip}^1(P)^2$ may flip the same variable twice and in that case has the same effect as skip. So $(v_1, v_2) \in \|\mathsf{flip}^1(P)^n\|$ does not imply that the Hammming distance between v_1 and v_2 is n. But what matters is that in Proposition 8 below (and Theorem 3 that is based on it), the occurrence of $\mathsf{flip}^1(\mathbb{P}_A)^m$ is preceded by the test $\mathsf{H}(A, \geq m)?$, which when successfully executed guarantee that the Hamming distance is n.

Proposition 6. *The following programs are equivalent:*

$$\begin{aligned}(\mathsf{flip}^{\geq 0}(P))^- &\equiv \mathsf{flip}^{\geq 0}(P)\\ (\mathsf{flip}^1(P)^n)^- &\equiv \mathsf{flip}^1(P)^n\\ (\mathsf{flip}^1(P)^{\leq n})^- &\equiv \mathsf{flip}^1(P)^{\leq n}\end{aligned}$$

Some useful DL-PA formulas Table 4 collects some interesting DL-PA formulas: $\mathsf{Valid}(\varphi)$ expresses that the formula φ is valid and $\mathsf{Sat}(\varphi)$ expresses that φ is satisfiable. The former is equivalent to $[\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi);\varphi?]\top$ and the latter is equivalent to $\langle\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi);\varphi?\rangle\top$. The formula $\mathsf{D}(\varphi, P)$ is true at a valuation v exactly when there is a closest φ-valuation whose symmetric difference with v is P. The formula $\mathsf{H}(\varphi, \geq m)$ is true at a valuation v exactly when the closest φ-valuations in the sense of the Hamming distance differ in at least m variables from v. For example:

$$\begin{aligned}\mathsf{Valid}(\varphi) &= [\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi)]\varphi\\ \mathsf{Sat}(\varphi) &= \langle\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi)\rangle\varphi\\ \mathsf{D}(\varphi, P) &= \langle\mathsf{flip}^{\geq 0}(P)\rangle\varphi \wedge \neg\langle\bigcup_{p\in P}\mathsf{flip}^{\geq 0}(P\setminus\{p\})\rangle\varphi\\ \mathsf{H}(\varphi, \geq m) &= \begin{cases}\top & \text{if } m = 0\\ \neg\langle\mathsf{flip}^1(\mathbb{P}_\varphi)^{\leq m-1}\rangle\varphi & \text{if } m \geq 1\end{cases}\end{aligned}$$

Table 4: Some useful DL-PA formulas, where φ is a formula, $P \subseteq \mathbb{P}_\varphi$ is a nonempty set of propositional variables, and $m \leq \mathsf{card}(\mathbb{P}_\varphi)$ is a non-negative integer.

$$\begin{aligned}\mathsf{H}(p, \geq 1) &= \neg\langle\mathsf{flip}^1(p)^{\leq 0}\rangle p\\ &= \neg\langle\mathsf{skip}\rangle p\\ &\leftrightarrow \neg p\\ \mathsf{H}(p\vee q, \geq 1) &= \neg\langle\mathsf{flip}^1(p, q)^{\leq 0}\rangle(p\vee q)\\ &\leftrightarrow \neg p \wedge \neg q\\ \mathsf{H}(p\vee q, \geq 2) &= \neg\langle\mathsf{flip}^1(p, q)^{\leq 1}\rangle(p\vee q)\\ &\leftrightarrow \neg\langle\mathsf{skip}\cup p{\leftarrow}\neg p\cup q{\leftarrow}\neg q\rangle(p\vee q)\\ &\leftrightarrow \neg(p\vee q \vee \langle p{\leftarrow}\neg p\rangle(p\vee q) \wedge \langle q{\leftarrow}\neg q\rangle(p\vee q))\\ &\leftrightarrow \neg((p\vee q) \vee (\neg p\vee q) \vee (p \vee \neg q))\\ &\leftrightarrow \bot\end{aligned}$$

The first two items of the next proposition show that both validity checking and satisfiability checking can both be reduced to model checking.

Proposition 7. *Let v a valuation and φ a formula.*

1. $v \in \|\mathsf{Valid}(\varphi)\|$ *iff φ is valid.*
2. $v \in \|\mathsf{Sat}(\varphi)\|$ *iff φ is satisfiable.*
3. $v \in \|\mathsf{D}(\varphi, P)\|$ *iff there is a valuation* $v_1 \in \|\varphi\|$ *such that* $v \dot{-} v_1 = P$ *and there is no valuation* $v_2 \in \|\varphi\|$ *such that* $v \dot{-} v_2 \subset P$.
4. $v \in \|\mathsf{H}(\varphi, \geq m)\|$ *iff there is no valuation* $v_1 \in \|\varphi\|$ *such that* $\mathsf{card}(v \dot{-} v_1) < m$.

The length of each formula of Table 4 is polynomial in the length of φ. (For $\mathsf{Sat}(\varphi)$ it is linear, for $\mathsf{D}(\varphi, P)$ and $\mathsf{H}(\varphi, \geq m)$ it is quadratic.)

From the above programs we can build a program which checks whether the Hamming distance to the closest A-valuation is m.

Proposition 8. *Let $m \leq \mathsf{card}(\mathbb{P}_A)$. Then*

$$(v_1, v_2) \in \|\mathsf{H}(A, \geq m)?;\mathsf{flip}^1(\mathbb{P}_A)^m;A\|$$

[3] The length of $\mathsf{flip}^1(P)^n$ and $\mathsf{flip}^1(P)^{\leq n}$ is quadratic in n. The length of the latter would have been cubic had we adopted the somewhat more natural definition $\bigcup_{k\leq n}\mathsf{flip}^1(P)^k$.

if and only if $v_2 \in \|A\|$, $\mathsf{card}(v_1 \dot{-} v_2) = m$, *and there is no* $v_2' \in \|A\|$ *such that* $\mathsf{card}(v_1 \dot{-} v_2') < m$.

WSS

Winslett's standard semantics (WSS) (Winslett 1995) is a syntax-dependent update operation that is less conservative than the other operations that we are going to examine, in the sense that each of the other operations preserves at least as much information of the original knowledge base as the WSS.

Semantics of the WSS Let v be a valuation and A be a propositional formula. The *WSS update of v by A* is

$$\begin{aligned} v \diamond^{\mathsf{wss}} A &= \{v_1 \in \|A\| \ : \ v \dot{-} v_1 \subseteq \mathbb{P}_A\} \\ &= \{v_1 \in \|A\| \ : \ \text{for every } p \notin \mathbb{P}_A, v(p) = v_1(p)\} \end{aligned}$$

So the set $v \diamond^{\mathsf{wss}} A$ is the set of A-valuations that agree with v on all the variables that do not occur in A. For example,

$$\emptyset \diamond^{\mathsf{wss}} p \vee q \ = \ \{\{p\}, \{q\}, \{p, q\}\}$$

The WSS update operation is syntax-dependent: depending on their language, logically equivalent formulas A_1 and A_2 may update the same valuation v in different ways.

Let A and B be propositional formulas. The *WSS update of B by A* is obtained by collecting the updates of every B-valuation by A:

$$B \diamond^{\mathsf{wss}} A = \bigcup_{v \in \|B\|} v \diamond^{\mathsf{wss}} A$$

The operation $\diamond^{\mathsf{wss}}$ therefore takes two propositional formulas and returns a set of valuations. For example:

$$\begin{aligned} \neg p \wedge \neg q \diamond^{\mathsf{wss}} p &= \|p \wedge \neg q\| \\ \neg p \wedge \neg q \diamond^{\mathsf{wss}} p \vee q &= \|p \vee q\| \\ \neg q \diamond^{\mathsf{wss}} p &= \|p \wedge \neg q\| \\ \neg p \vee \neg q \diamond^{\mathsf{wss}} p &= \|p\| \end{aligned}$$

Expressing the WSS operation in DL-PA Let us turn to the problem of expressing WSS updates in DL-PA.

Theorem 1. *Let A, B be propositional formulas. Let π_A^{wss} be the* DL-PA *program*

$$\mathsf{flip}^{\geq 0}(\mathbb{P}_A); A?$$

Then $B \diamond^{\mathsf{wss}} A = \|\langle (\pi_A^{\mathsf{wss}})^- \rangle B\|$.

Via the program equivalences for the converse operator of Proposition 6, it follows that $B \diamond^{\mathsf{wss}} A = \|\langle A?; \mathsf{flip}^{\geq 0}(\mathbb{P}_A) \rangle B\|$.

For example, consider the input formula $A = p$. Then

$$\begin{aligned} \pi_p^{\mathsf{wss}} &= \mathsf{flip}^{\geq 0}(P); p? \\ &= (p \leftarrow \top \cup p \leftarrow \bot); p? \\ &\equiv (p \leftarrow \top; p?) \cup (p \leftarrow \bot; p?) \\ &\equiv p \leftarrow \top \\ (\pi_p^{\mathsf{wss}})^- &\equiv (p \leftarrow \top)^- \\ &\equiv p? \cup (p?; p \leftarrow \bot) \end{aligned}$$

For the base $B = \neg p \wedge q$ and the input $A = p$ we have:

$$\begin{aligned} \langle (\pi_A^{\mathsf{wss}})^- \rangle B &\leftrightarrow \langle p?; (p \leftarrow \top \cup p \leftarrow \bot) \rangle (\neg p \wedge q) \\ &\leftrightarrow p \wedge (\langle p \leftarrow \top \rangle (\neg p \wedge q) \vee \langle p \leftarrow \bot \rangle (\neg p \wedge q)) \\ &\leftrightarrow p \wedge (\bot \vee (\top \wedge q)) \\ &\leftrightarrow p \wedge q \end{aligned}$$

So the update of $\neg p \wedge q$ by p is equivalent to $p \wedge q$, as expected.

Observe that the length of π_A^{wss} is linear in the length of A.

PMA

Winslett's possible models approach (PMA) operation (Winslett 1988; 1990) is the belief update operation that is most cited and used, e.g. in database theory (Winslett 1995; Chomicki and Marcinkowski 2004), reasoning about actions and planning (Baral 1995), logic programming (Slota and Leite 2010) and in description logics and the semantic web (Baader et al. 2005; Liu et al. 2006; 2011).

The PMA is more conservative than the WSS, in the sense that it preserves more information from the base B. This is achieved by a principle of minimal change.

Semantics of the PMA operation Let v be a valuation and A be a propositional formula. The *PMA update of v by a formula A* is

$$v \diamond^{\mathsf{pma}} A = \{v_1 \in \|A\| \ : \ \text{there is no } v_2 \in \|A\| \text{ s.t. } v \dot{-} v_2 \subset v \dot{-} v_1\}$$

So the set $v \diamond^{\mathsf{pma}} A$ is the set of A-valuations that are closest to v w.r.t. symmetric difference. For example,

$$\begin{aligned} \emptyset \diamond^{\mathsf{pma}} p \vee q &= \{\{p\}, \{q\}\} \\ \emptyset \diamond^{\mathsf{pma}} (p \wedge q) \vee r &= \{\{p, q\}, \{r\}\} \end{aligned}$$

Let A and B be propositional formulas. Just as all the update operations *à la* Katsuno-Mendelzon (Katsuno and Mendelzon 1992), the *PMA update of B by A* collects the updates of each B-valuation by A:

$$B \diamond^{\mathsf{pma}} A = \bigcup_{v \in \|B\|} v \diamond^{\mathsf{pma}} A$$

For example:

$$\begin{aligned} \neg p \wedge \neg q \diamond^{\mathsf{pma}} p &= \|p \wedge \neg q\| \\ \neg q \diamond^{\mathsf{pma}} p &= \|p \wedge \neg q\| \\ \neg p \wedge \neg q \diamond^{\mathsf{pma}} p \vee q &= \|p \oplus q\| \\ \neg p \wedge \neg q \wedge \neg r \diamond^{\mathsf{pma}} (p \wedge q) \vee r &= \|(p \wedge q) \oplus r\| \end{aligned}$$

Expressing the PMA operation in DL-PA We now polynomially translate the update problem $B \diamond^{\mathsf{pma}} A$ into DL-PA: we define a family of update programs π_A^{pma} whose length is linear in the length of A.

It is tempting to define π_A^{pma} to be the program $\mathsf{while}\ \neg A\ \mathsf{do}\ \mathsf{flip}^1(\mathbb{P}_A)$, where while is as defined above. However, it does not have the exact behaviour of $\diamond^{\mathsf{pma}}$. Indeed, the interpretation of

$$\mathsf{while}\ \neg((p \wedge q) \vee r)\ \mathsf{do}\ \mathsf{flip}^1(\{p, q, r\})$$

relates the empty valuation to the four valuations $\{p, q\}$, $\{r\}$,

$\{p, r\}$, and $\{q, r\}$, while we have seen above that the PMA update of the empty valuation only outputs the first two of these valuations. Actually the above program defines an update operation with strength between WSS and PMA; it seems that it has not been considered in the literature up to now.

The exact DL-PA counterpart of the PMA updates is a bit more involved. The most straightforward way is to define π_A^{pma} to be the program

$$A?\ \cup\ \left(\neg A?;\ \bigcup_{\emptyset \subset P \subseteq \mathbb{P}_A} \mathsf{D}(A,P)?\ ;\ \mathsf{flip}^{\geq 0}(P)\ ;\ A?\right)$$

where $\mathsf{D}(A,P)$ is defined in Table 4. While it does the same job as $\diamond^{\mathsf{pma}}$, its length is exponential in the length of A due to the nondeterministic choice in the powerset of $\mathbb{P}_A$.

Our polynomial embedding of the PMA involves storing the values of variables as defined in Table 3.

Theorem 2. *Let A, B be propositional formulas. Let $\mathbb{P}_A = \{p_1, \ldots, p_n\}$ be set of variables occurring in A. Let $\mathbb{P}'_A = \{p'_1, \ldots, p'_n\}$ be such that the p'_k neither occur in A nor in B. Let π_A^{pma} be the following program:*

$$\begin{aligned}&\mathsf{store}(\mathbb{P}_A); \mathsf{flip}^{\geq 0}(\mathbb{P}_A); A?;\\ &[\mathsf{restore}^1(\mathbb{P}_A); \mathsf{restore}^{\geq 0}(\mathbb{P}_A)]\neg A?;\\ &\mathsf{flip}^{\geq 0}(\mathbb{P}'_A)\end{aligned}$$

Then $B \diamond^{\mathsf{pma}} A = \|\langle(\pi_A^{\mathsf{pma}})^-\rangle B\|$ for every valuation v.

The length of the program π_A^{pma} is linear in the length of A. It starts by saving the truth values of the variables of A by means of $\mathsf{store}(\mathbb{P}_A)$. It then goes to an A-valuation by $\mathsf{flip}^{\geq 0}(\mathbb{P}_A); A?$. The next step is to check whether all the ways one can get there are exclusively made up of $\neg A$-valuations: an A-valuation is disregarded if another A-valuation can be attained by fewer changes. Finally the copies p'_k of the variable p_k of A are forgotten by $\mathsf{flip}^{\geq 0}(\mathbb{P}'_A)$. For example,

$$\begin{aligned}\pi_p^{\mathsf{pma}} &= p'{\leftarrow}p; (p{\leftarrow}\top \cup p{\leftarrow}\bot); p?;\\ &\quad [p{\oplus}p'?; p{\leftarrow}p'; (\mathsf{skip} \cup p{\leftarrow}p')]\neg p?; (p'{\leftarrow}\top \cup p'{\leftarrow}\bot)\\ &\equiv p'{\leftarrow}p; p{\leftarrow}\top; [p{\oplus}p'?; p{\leftarrow}p']\neg p?; (p'{\leftarrow}\top \cup p'{\leftarrow}\bot)\\ &\equiv p'{\leftarrow}p; p{\leftarrow}\top; [p{\oplus}p'?]\neg p'?; (p'{\leftarrow}\top \cup p'{\leftarrow}\bot)\\ &\equiv p'{\leftarrow}p; p{\leftarrow}\top; (p'{\leftarrow}\top \cup p'{\leftarrow}\bot)\\ &\equiv p{\leftarrow}\top; (p'{\leftarrow}\top \cup p'{\leftarrow}\bot)\end{aligned}$$

As p' is fresh, the 'net effect' of the update program π_p^{pma} is just $p{\leftarrow}\top$, as expected. (Observe that while the programs π_p^{pma} and $p{\leftarrow}\top$ are not equivalent; what matters to us is that the formulas $\langle(\pi_p^{\mathsf{pma}})^-\rangle B$ and $\langle p{\leftarrow}\top^-\rangle B$ are so.)

Just as for the other update operators, the converse operator can be eliminated from π_A^{pma}. We do not state it because we were not able to simplify it further.

Forbus

Forbus's update operation (Forbus 1989) is even more conservative than Winslett's. It is based on minimisation of the Hamming distance between valuations.

Semantics of Forbus's operation *Forbus's update of v by A* is defined as:

$$v \diamond^{\mathsf{forbus}} A = \{v_1 \in \|A\| \ : \text{there is no } v_2 \in \|A\| \text{ such that } \mathsf{card}(v \dot{-} v_2) < \mathsf{card}(v \dot{-} v_1)\}$$

So the set $v \diamond^{\mathsf{forbus}} A$ is the set of A-valuations that are closest to v w.r.t. the Hamming distance. For example,

$$\begin{aligned}\emptyset \diamond^{\mathsf{forbus}} p \vee q &= \{\{p\}, \{q\}\}\\ \emptyset \diamond^{\mathsf{forbus}} (p \wedge q) \vee r &= \{\{r\}\}\end{aligned}$$

The last example illustrates the difference with the PMA update operation.

Let A and B be propositional formulas. Just as WSS and PMA updates, *Forbus's update of B by A* is the pointwise update of the B-valuations by A:

$$B \diamond^{\mathsf{forbus}} A = \bigcup_{v \in \|B\|} v \diamond^{\mathsf{forbus}} A$$

For example:

$$\begin{aligned}\neg p \wedge \neg q \diamond^{\mathsf{forbus}} p \vee q &= \|p \oplus q\|\\ \neg p \wedge \neg q \wedge \neg r \diamond^{\mathsf{forbus}} (p \wedge q) \vee r &= \|\neg p \wedge \neg q \wedge r\|\end{aligned}$$

Expressing Forbus's operation in DL-PA We polynomially transform update problems of the form $B \diamond^{\mathsf{forbus}} A$ into DL-PA: we define a family of update programs π_A^{forbus} whose length is cubic in the length of A.

Theorem 3. *Let A, B be propositional formulas. Let π_A^{forbus} be the following program:*

$$\left(\bigcup_{0 \leq m \leq \mathsf{card}(\mathbb{P}_A)} \mathsf{H}(A, \geq m)?; \mathsf{flip}^1(\mathbb{P}_A)^m\right); A?$$

Then $B \diamond^{\mathsf{forbus}} A = \|\langle(\pi_A^{\mathsf{forbus}})^-\rangle B\|$.

The length of the program π_A^{forbus} is cubic in the length of A. It nondeterministically selects an integer m, checks whether the Hamming distance to A is at least m, flips m variables, and checks whether A is true. For example:

$$\begin{aligned}\pi_p^{\mathsf{forbus}} &= \Big((\mathsf{H}(p, \geq 0)?; \mathsf{flip}^1(p)^0) \cup (\mathsf{H}(p, \geq 1)?; \mathsf{flip}^1(p)^1)\Big); p?\\ &\equiv \Big((\top?; \mathsf{skip}) \cup (\neg\langle \mathsf{flip}^1(p)^{\leq 0}\rangle p?; p{\leftarrow}\neg p)\Big); p?\\ &\equiv \Big(\mathsf{skip} \cup (\neg p?; p{\leftarrow}\neg p)\Big); p?\\ &\equiv p? \cup (\neg p?; p{\leftarrow}\neg p; p?)\\ &\equiv p{\leftarrow}\top\end{aligned}$$

Therefore $B \diamond^{\mathsf{forbus}} p = \|\langle p{\leftarrow}\top^-\rangle B\|$. Here is another example:

$$\begin{aligned}\pi_{p \vee q}^{\mathsf{forbus}} \equiv\ & (\mathsf{H}(p \vee q, \geq 0)?; \mathsf{flip}^1(p,q)^0; p \vee q?) \cup\\ & (\mathsf{H}(p \vee q, \geq 1)?; \mathsf{flip}^1(p,q)^1; p \vee q?) \cup\\ & (\mathsf{H}(p \vee q, \geq 2)?; \mathsf{flip}^1(p,q)^2; p \vee q?)\\ \equiv\ & (\top?; \mathsf{skip}; p \vee q?) \cup\\ & (\neg(p \vee q)?; (p{\leftarrow}\neg p \cup q{\leftarrow}\neg q); p \vee q?) \cup\\ & (\bot?; \mathsf{flip}^1(p,q)^2; p \vee q?)\\ \equiv\ & p \vee q? \cup (\neg(p \vee q)?; (p{\leftarrow}\neg p \cup q{\leftarrow}\neg q))\end{aligned}$$

The identity can be simplified by means of the program equivalences for the converse operator:

$$B\diamond^{\mathsf{forbus}}A = \|\langle A?; \Big(\bigcup_{0\le m\le \mathsf{card}(\mathbb{P}_A)} \mathsf{flip}^1(\mathbb{P}_A)^m; \mathsf{H}(A,\ge m)?\Big)\rangle B\|.$$

Dalal

According to Katsuno and Mendelzon's distinction (Katsuno and Mendelzon 1992), Dalal's belief change operation (Dalal 1988) is not an update operation but rather a revision operation. We follow the usage in the literature and denote it by $*^{\mathsf{dalal}}$ (and not by $\diamond^{\mathsf{dalal}}$).

We are going to capture $*^{\mathsf{dalal}}$ by a DL-PA program $\pi^{\mathsf{dalal}}_{A,B}$ which depends not only on the input A, but also on the base B. One of the reasons for that is that Dalal's definition distinguishes two cases: when B is satisfiable and when it is unsatisfiable. In the latter case the result of the update by A is stipulated to be $\|A\|$. This guarantees that the revised base is satisfiable as soon as the input formula A is so. This however does not square well with our embedding into DL-PA: in modal logic, when B is unsatisfiable then $\langle(\pi^{\mathsf{dalal}}_{A,B})^-\rangle B$ is unsatisfiable, too. We therefore consider a variant where the revision of an unsatisfiable base is unsatisfiable, too. In the end of the section we embed Dalal's original operation.

Semantics of Dalal's operation Just as Forbus's update operation, Dalal's revision operation is based on minimisation of the Hamming distance between valuations. However, as Dalal's is a revision operator we minimise globally instead of valuation-wise.

Let A and B be propositional formulas. *Dalal's revision of B by A* is defined as:

$$\begin{aligned} B*^{\mathsf{dalal}}A = \{v_A \in \|A\| \ &: \text{there is } v_B \in \|B\| \text{ such that} \\ &\mathsf{card}(v_A \dot{-} v_B) \le \mathsf{card}(v'_A \dot{-} v'_B) \\ &\text{for all } v'_A \in \|A\|, v'_B \in \|B\| \ \} \end{aligned}$$

It follows that when B is unsatisfiable then $B*^{\mathsf{dalal}}A$ is unsatisfiable. As mentioned above, our definition differs from Dalal's original definition in this point.

Here is an example:

$$\neg p\vee\neg q *^{\mathsf{dalal}} p \ = \ \|p\wedge\neg q\|$$

It illustrates that revision operations satisfy the preservation postulate, which says that when $B\wedge A$ is satisfiable then $B * A = \|B\wedge A\|$. This is the main difference with update operations: WSS, PMA, and Forbus's operation all agree that the update of $\neg p\vee\neg q$ by p should equal $\|p\|$.

Expressing Dalal's operation in DL-PA We polynomially transform revision problems of the form $B*^{\mathsf{dalal}}A$ into DL-PA: we define a family of revision programs $\pi^{\mathsf{dalal}}_{A,B}$ whose length is cubic in the length of A and B.

Theorem 4. *Let A, B be propositional formulas. Let $\pi^{\mathsf{dalal}}_{A,B}$ be the following program:*

$$\mathsf{flip}^{\ge 0}(\mathbb{P}_B)\ ;\ B?\ ;\ \left(\bigcup_{0\le m\le \mathsf{card}(\mathbb{P}_A)} [\mathsf{flip}^{\ge 0}(\mathbb{P}_B)\ ;\ B?]\mathsf{H}(A,\ge m)?\ ;\mathsf{flip}^1(\mathbb{P}_A)^m\right); A?$$

Then $B*^{\mathsf{dalal}}A = \|\langle(\pi^{\mathsf{dalal}}_{A,B})^-\rangle\top\|$.

The length of the program $\pi^{\mathsf{dalal}}_{A,B}$ is cubic in the sum of $|A| + |B|$. It visits all the B-valuations v_B via the program $\mathsf{flip}^{\ge 0}(\mathbb{P}_B); B?$, failing if there is no such valuation. It then nondeterministically selects an integer m such that the Hamming distance between the B-valuations and the A-valuations is at least m, flips m such variables, and checks whether A is true.

When the input is a literal then we obtain:

$$\begin{aligned} \pi^{\mathsf{dalal}}_{p,B} &= \mathsf{flip}^{\ge 0}(\mathbb{P}_B); B?\ ; \\ &\quad \Big(([\mathsf{flip}^{\ge 0}(\mathbb{P}_B); B?]\mathsf{H}(p,\ge 0)?\ ; \mathsf{flip}^1(p)^0) \cup \\ &\quad\ ([\mathsf{flip}^{\ge 0}(\mathbb{P}_B); B?]\mathsf{H}(p,\ge 1)?\ ; \mathsf{flip}^1(p)^1)\Big); p? \\ &= \mathsf{flip}^{\ge 0}(\mathbb{P}_B); B?\ ; \\ &\quad \Big([\mathsf{flip}^{\ge 0}(\mathbb{P}_B); B?]\top?; \mathsf{skip} \cup \\ &\quad\ ([\mathsf{flip}^{\ge 0}(\mathbb{P}_B); B?]\neg p?\ ; p{\leftarrow}\neg p)\Big); p? \\ &\equiv \mathsf{flip}^{\ge 0}(\mathbb{P}_B); B?\ ; \ \Big(p?\ \cup ([\mathsf{flip}^{\ge 0}(\mathbb{P}_B); B?]\neg p?\ ; p{\leftarrow}\top)\Big) \end{aligned}$$

and likewise for $B*^{\mathsf{dalal}}\neg p$. So when $B\wedge p$ is consistent then $\pi^{\mathsf{dalal}}_{p,B}$ goes to a $B\wedge p$-valuation, and updates by p otherwise.

The proof of Theorem 4 is in two steps: first we prove that $B*^{\mathsf{dalal}}A = \|\langle(\pi^{\mathsf{dalal}}_{A,B})^-\rangle B\|$, and then that $\|\langle(\pi^{\mathsf{dalal}}_{A,B})^-\rangle B\| = \|\langle(\pi^{\mathsf{dalal}}_{A,B})^-\rangle\top\|$. The second step relies on the validity of the equivalence $\langle B?; \mathsf{flip}^{\ge 0}(\mathbb{P}_B)\rangle\top \leftrightarrow \langle B?; \mathsf{flip}^{\ge 0}(\mathbb{P}_B)\rangle B$.

Finally, we note that we can also capture Dalal's original definition where the revision of an unsatisfiable B by A equals $\|A\|$ (instead of being unsatisfiable as with our definition). The modified identity which does the job is

$$B*^{\mathsf{dalal}}A = \|(\neg\mathsf{Sat}(B)\wedge A)\vee\langle(\pi^{\mathsf{dalal}}_{A,B})^-\rangle\top\|$$

Discussion

Let us come back to the belief change consequence problem. This is a decision problem, and tight complexity bounds are known for each of the operations that we have considered: the belief change consequence problem is

- NP complete for the WSS update operation (Herzig and Rifi 1999),
- $\mathrm{P}^{\mathrm{NP}[O(\log(n))]}$ for Dalal's belief revision operation, i.e., polynomial with $\log(n)$ calls to an NP oracle (Eiter and Gottlob 1992), and
- Π^2_p complete for Winslett's PMA update and Forbus's update (Eiter and Gottlob 1992).

Our embeddings are in the star-free fragment of the language of DL-PA. We therefore obtain a PSPACE upper bound for the complexity of deciding whether $(B\circ A)\to C$ is the case. So this is clearly suboptimal. However, the nesting of modal operators in our decision problems $B\to[\pi_A]C$ is bounded: for the WSS, there is a single $\forall$ quantification and for the PMA, $[\pi_A]C$ is of the form

$$[\pi_1; \mathsf{flip}^{\ge 0}(\mathbb{P}_A); \pi_2; [\pi_3; \mathsf{restore}^{\ge 0}(\mathbb{P}_A)]\neg A?]C$$

where π_1, π_2, π_3 are deterministic programs, which is equivalent to

$$[\pi_1; \mathsf{flip}^{\ge 0}(\mathbb{P}_A); \pi_2](\langle\pi_3; \mathsf{restore}^{\ge 0}(\mathbb{P}_A)\rangle A\vee C)$$

The latter makes the ∀-∃-quantification obvious. For Forbus's operator, $[\pi_A]C$ is basically of the form

$$\Big[\Big(\bigcup_{0\le m\le \mathsf{card}(\mathbb{P}_A)} \neg\langle \mathsf{flip}^1(\mathbb{P}_A)^{\le m-1}\rangle A?; \mathsf{flip}^1(\mathbb{P}_A)^m\Big); A\Big]C$$

which is again a ∀-∃-quantification. The argument for Dalal's operation is similar. By Proposition 7, checking the validity of $B \to [\pi_A]C$ can be done by checking whether $v \in \|\mathsf{flip}^{\ge 0}(\mathbb{P}_A)(B \to [\pi_A]C)\|$. The latter can be checked by an alternating algorithm, and as the alternation-depth is 2 for PMA and Forbus's update and for Dalal's revision, we obtain a Π^2_p upper bound for the respective belief change decision problems.

We leave to future work the implementation of decision procedures for DL-PA and its fragments.

Up to now, complexity characterisations of belief change operations often resorted to translations from or to QBF (see e.g. (Eiter and Gottlob 1992)). Our logic DL-PA is clearly at least as expressive as QBF: the QBF formula $\exists p\varphi$ is equivalent to the DL-PA formula $\langle p{\leftarrow}\top \cup p{\leftarrow}\bot\rangle\varphi$, which is exploited in (Herzig et al. 2011) to show PSPACE hardness of satisfiability and model checking. We believe that the richer language of programs of DL-PA provides a more natural tool to capture domains involving action and change. The present paper is part of a research program demonstrating the applicability of DL-PA to problems involving dynamics in finite domains. In other papers we have embedded several frameworks: Coalition Logic of Propositional Control and its extension by delegation programs (van der Hoek, Walther, and Wooldridge 2010) in (Herzig et al. 2011), equilibrium logic underlying ASP and its extension by update operations in (Fariñas del Cerro, Herzig, and Su 2013), a simple version of separation logic in (Herzig 2013), belief merging operations in (Herzig, Pozos Parra, and Schwarzentruber 2014), and Dung's argumentation frameworks and their update in (Doutre, Herzig, and Perrussel 2014).

Conclusion

We have given polynomial embeddings of several prominent belief change operations into a single framework: Dynamic Logic of Propositional Assignments DL-PA. Within that framework, we have given syntactic counterparts to the most popular semantically defined belief change operations: Winslett's update operations WSS and PMA, Forbus's operation operation, and Dalal's revision operation. The reduction of the DL-PA formula representing the updated belief base can be exponentially longer than the original formula; however, our examples illustrate that sometimes our syntactic representation of the updated belief base can be substantially reduced.

As an aside, we have also defined a new update operation when discussing the program $\mathsf{while}\ \neg A\ \mathsf{do}\ \mathsf{flip}^1(\mathbb{P}_A)$ as a candidate for the update program implementing the PMA. The corresponding update operation is a bit more conservative than the WSS and less than the PMA. It seems that it was not studied in the literature up to now.

Our approach can be extended to cover other concrete belief change operations. For example, the program implementing Satoh's revision operation (Satoh 1988) combines the search of a 'good starting point' as performed by the prefix $\mathsf{flip}^{\ge 0}(\mathbb{P}_B); B?$ of the revision program $\pi^{\mathsf{dalal}}_{A,B}$ with the copy-based change minimisation of the program π^{pma}_A for the PMA.

Acknowledgements

The work in this paper was supported by the French Agence Nationale de la Recherche (ANR) within the DynRes project.

References

Alchourrón, C.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *J. of Symbolic Logic* 50:510–530.

Baader, F.; Lutz, C.; Milicic, M.; Sattler, U.; and Wolter, F. 2005. A description logic based approach to reasoning about web services. In *Proceedings of the WWW 2005 Workshop on Web Service Semantics (WSS2005)*.

Balbiani, P.; Herzig, A.; and Troquard, N. 2013. Dynamic logic of propositional assignments: a well-behaved variant of PDL. In Kupferman, O., ed., *Logic in Computer Science (LICS), New Orleans, June 25-28, 2013*. IEEE.

Baral, C. 1995. Reasoning about actions: Non-deterministic effects, constraints, and qualification. In *IJCAI*, 2017–2026. Morgan Kaufmann.

Bienvenu, M.; Herzig, A.; and Qi, G. 2008. Prime implicate-based belief revision operators. In Ghallab, M.; Spyropoulos, C. D.; Fakotakis, N.; and Avouris, N., eds., *European Conference on Artificial Intelligence (ECAI)*, 741–742. IOS Press.

Chomicki, J., and Marcinkowski, J. 2004. On the computational complexity of minimal-change integrity maintenance in relational databases. In *Inconsistency Tolerance*, 119–150. Springer.

Dalal, M. 1988. Investigations into a theory of knowledge base revision: preliminary report. In *Proc. 7th Conf. on Artificial Intelligence (AAAI'88)*, 475–479.

de Giacomo, G. 1996. Eliminating "converse" from Converse PDL. *Journal of Logic, Language and Information (JoLLI)* 5:193–208.

Doutre, S.; Herzig, A.; and Perrussel, L. 2014. A dynamic logic framework for abstract argumentation. In and., ed., *International Conference, Principles of Knowledge Representation and Reasoning (KR), Vienna, Austria*.

Eiter, T., and Gottlob, G. 1992. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artif. Intell.* 57(2-3):227–270.

Fariñas del Cerro, L.; Herzig, A.; and Su, E. I. 2013. Combining equilibrium logic and dynamic logic. In Cabalar, P., and Son, T. C., eds., *LPNMR*, volume 8148 of *Lecture Notes in Computer Science*, 304–316. Springer.

Forbus, K. D. 1989. Introducing actions into qualitative simulation. In Sridharan, N. S., ed., *Proc. 11th Int. Joint Conf. on Artificial Intelligence (IJCAI'89)*, 1273–1278. Morgan Kaufmann Publishers.

Harel, D.; Kozen, D.; and Tiuryn, J. 2000. *Dynamic Logic*. MIT Press.

Harel, D. 1984. Dynamic logic. In Gabbay, D. M., and Günthner, F., eds., *Handbook of Philosophical Logic*, volume II. D. Reidel, Dordrecht. 497–604.

Herzig, A., and Rifi, O. 1999. Propositional belief base update and minimal change. *Artificial Intelligence Journal* 115(1):107–138.

Herzig, A.; Lorini, E.; Moisan, F.; and Troquard, N. 2011. A dynamic logic of normative systems. In Walsh, T., ed., *International Joint Conference on Artificial Intelligence (IJCAI)*, 228–233. Barcelona: IJCAI/AAAI. Erratum at http://www.irit.fr/~Andreas.Herzig/P/Ijcai11.html.

Herzig, A.; Lang, J.; and Marquis, P. 2013. Propositional update operators based on formula/literal dependence. *ACM Transactions on Computational Logic (TOCL)* 14(3):24.

Herzig, A.; Pozos Parra, P.; and Schwarzentruber, F. 2014. Belief merging in Dynamic Logic of Propositional Assignments. In Beierle, C., and Meghini, C., eds., *International Symposium on Foundations of Information and Knowledge Systems (FoIKS) (FolKS), Bordeaux*. Springer.

Herzig, A. 1996. The PMA revisited. In Aiello, L. C., and Shapiro, S., eds., *Proc. Int. Conf. on Knowledge Representation and Reasoning (KR'96)*, 40–50. Morgan Kaufmann Publishers.

Herzig, A. 2013. A simple separation logic. In Libkin, L.; Kohlenbach, U.; and de Queiroz, R. J. G. B., eds., *WoLLIC*, volume 8071 of *Lecture Notes in Computer Science*, 168–178. Springer.

Katsuno, H., and Mendelzon, A. O. 1992. On the difference between updating a knowledge base and revising it. In Gärdenfors, P., ed., *Belief revision*. Cambridge University Press. 183–203. (preliminary version in Allen, J.A., Fikes, R., and Sandewall, E., eds., Principles of Knowledge Representation and Reasoning: Proc. 2nd Int. Conf., pages 387–394. Morgan Kaufmann Publishers, 1991).

Kourousias, G., and Makinson, D. 2007. Parallel interpolation, splitting, and relevance in belief change. *Journal of Symbolic Logic* 72(3):994–1002.

Lang, J.; Liberatore, P.; and Marquis, P. 2003. Propositional independence: Formula-variable independence and forgetting. *J. Artif. Intell. Res. (JAIR)* 18:391–443.

Lang, J. 2007. Belief update revisited. In *Proc. of the 10th International Joint Conference on Artificiel Intelligence (IJCAI'07)*, 2517–2522.

Liu, H.; Lutz, C.; Milicic, M.; and Wolter, F. 2006. Updating description logic ABoxes. In Doherty, P.; Mylopoulos, J.; and Welty, C., eds., *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, 46–56. AAAI Press.

Liu, H.; Lutz, C.; Milicic, M.; and Wolter, F. 2011. Foundations of instance level updates in expressive description logics. *Artificial Intelligence* 175(18):2170–2197.

Marchi, J.; Bittencourt, G.; and Perrussel, L. 2010. Prime forms and minimal change in propositional belief bases. *Ann. Math. Artif. Intell.* 59(1):1–45.

Parikh, R. 1999. Beliefs, belief revision, and splitting languages. In Moss, L. S.; Ginzburg, J.; and de Rijke, M., eds., *Logic, Language and Computation, Vol. 2*. Stanford, CA, USA: Center for the Study of Language and Information. 266–278.

Satoh, K. 1988. Nonmonotonic reasoning by minimal belief revision. In *FGCS*, 455–462.

Slota, M., and Leite, J. 2010. On semantic update operators for answer-set programs. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, 957–962. IOS Press.

Tiomkin, M. L., and Makowsky, J. A. 1985. Propositional dynamic logic with local assignments. *Theor. Comput. Sci.* 36:71–87.

van der Hoek, W.; Walther, D.; and Wooldridge, M. 2010. On the logic of cooperation and the transfer of control. *J. of AI Research (JAIR)* 37:437–477.

van Eijck, J. 2000. Making things happen. *Studia Logica* 66(1):41–58.

Winslett, M.-A. 1988. Reasoning about action using a possible models approach. In *Proc. 7th Conf. on Artificial Intelligence (AAAI'88)*, 89–93.

Winslett, M.-A. 1990. *Updating Logical Databases*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Winslett, M.-A. 1995. Updating logical databases. In Gabbay, D. M.; Hogger, C. J.; and Robinson, J. A., eds., *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4. Oxford University Press. 133–174.

Belief Change and Base Dependence

Mehrdad Oveisi
School of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
Canada
oveisi@cs.sfu.ca

James P. Delgrande
School of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
Canada
jim@cs.sfu.ca

Francis Jeffry Pelletier
Department of Philosophy
University of Alberta
Edmonton, AB T6G 2E7
Canada
francisp@ualberta.ca

Fred Popowich
School of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
Canada
popowich@cs.sfu.ca

Abstract

A strong intuition for AGM belief change operations, Gärdenfors suggests, is that formulas that are independent of a change should remain intact. Based on this intuition, Farias and Herzig axiomatize a *dependence* relation w.r.t. a belief set, and formalize the connection between dependence and belief change. In this paper, we introduce *base dependence* as a relation between formulas w.r.t. a *belief base*. After an axiomatization of base dependence, we formalize the connection between base dependence and a particular belief base change operation, saturated kernel contraction. Moreover, we prove that base dependence is a *reversible generalization* of Farias and Herzig's dependence. That is, in the special case when the underlying belief base is deductively closed (i.e., it is a belief set), base dependence reduces to dependence. Finally, an intriguing feature of Farias and Herzig's formalism is that it meets other criteria for dependence, namely, Keynes' conjunction criterion for dependence (CCD) and Grdenfors' conjunction criterion for independence (CCI). We show that our base dependence formalism also meets these criteria. More interestingly, we offer a more specific criterion that implies both CCD and CCI, and show our base dependence formalism also meets this new criterion.

Introduction

Belief Change

The *AGM* paradigm of *belief change* studies the dynamics of belief states in light of new information (Alchourrn, GŁrdenfors, and Makinson 1985). For theoretical simplification, AGM idealizes a belief state as a *belief set* or a *theory*: a set of logical formulas that is closed under implication.

An important variant of the original AGM approach uses *belief bases* instead of using belief sets. Belief bases are not necessarily deductively closed, and they are usually *finite*. Thus, they are more suitable to be represented in finite machines. Also, many authors have argued that, compared to belief sets, belief bases are more expressive (Hansson 2003), and they are more tolerant of inconsistency (Hansson and Wassermann 2002). Therefore, belief bases can be more useful in practice than belief sets.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Belief Change and Dependence

A long standing intuition concerning belief change is that formulas independent of a change should remain intact (GŁrdenfors 1990). In *Belief Change and Dependence*, Farias del Cerro and Herzig (1996) (FH) attempt to ground this intuition by axiomatizing a particular *dependence* relation in a close relationship to belief change. FH's work is particularly interesting and unique in the sense that it fits the original AGM model of belief change. Their stated aim is both "to give a formal account of the notion of dependence, and to employ it in belief change." This deep integration into the AGM model sets apart their work from other works on relevance or dependence in the context of belief change.

Belief Change and Base Dependence

A natural next step is to find a similar connection between dependence and *belief base contraction*. We call such a dependence (or relevance) relation *base dependence* (or *base relevance*). In this work, we provide an axiomatization of base dependence, and establish its relation to belief base contraction. Interestingly, base dependence turns out to be a *reversible generalization* of FH's dependence. That is, in the special case that a belief base is deductively closed (i.e., it is a belief set), the base dependence relation reduces to FH's original dependence relation.

Composite Dependence

One interesting aspect of FH's work is that some of the axioms that they use to capture the concept of dependence come from intuitions put forward previously. For example, Keynes (1921) holds that there is an intuitive relationship between relevance (dependence) and logical conjunction that should stay valid for any reasonable definition of relevance. Calling it the *Conjunction Criterion for Dependence*, CCD, FH

formulate it as follows:

> If δ depends on α and δ depends on β then δ depends on $\alpha \wedge \beta$. (CCD)

Moreover, Gärdenfors (1978) puts forward another principle that he believes should hold for relevance/dependence relations, the *Conjunction Criterion for Independence*, CCI.

> If δ is independent of α and δ is independent of β then δ is independent of $\alpha \wedge \beta$.

It maintains its intuitive appeal in its contrapositive form:

> If δ depends on $\alpha \wedge \beta$ then δ depends on α or δ depends on β. (CCI)

Notably, our formalism preserves both CCD and CCI. Furthermore, we offer a new and more specific criterion for dependence, which we call the *Conjunction Criterion of Dependence Factoring*, CCDF. We show that CCDF implies both CCD and CCI, and that our base dependence formalism meets the three criteria: CCDF and so CCD and CCI.

Contributions

The contributions in this work are as follows. We offer an axiomatization of base dependence for belief base formulas, and we provide characterization theorems relating base dependence to belief base contraction. We then show that base dependence is a reversible generalization of FH's dependence relation. We also show that while generalizing the dependence relation, base dependence preserves some of the most interesting properties of dependence, particularly, Keynes' CCD, and Grdenfors' CCI. Finally we put forward a more specific conjunction criterion of dependence, CCDF, that implies both CCD and CCI, and show that this new criterion is also met by base dependence.

Background

Formal Preliminaries

We assume $\mathcal{L}$ to be a propositional language defined on a finite set of propositional variables or atoms $\mathcal{V}$ with the usual Boolean operators negation $\neg$, conjunction $\wedge$, disjunction $\vee$, and implication $\rightarrow$. We will use lower case Greek letters α, β, δ, etc. as meta variables over sentences in $\mathcal{L}$. For convenience, we introduce the sentential constants $\top$ and $\bot$ representing truth and falsity respectively. A logical consequence α of a set of formulas B is represented by $B \vdash \alpha$. Also Cn is a consequence operator, a total function taking sets of formulas to sets of formulas, defined as $\mathrm{Cn}(B) = \{\alpha \mid B \vdash \alpha\}$.

For the proofs of theorems in this paper, please refer to (Oveisi 2013).

Belief Contraction

To model rational belief change, AGM uses *rationality postulates* to describe what constitute operators for belief change and it also specifies how to *construct* such operators. The belief contraction postulates are as follows:

$(\div 1)$ $K \div \alpha$ is a belief set (closure)

$(\div 2)$ $K \div \alpha \subseteq K$ (inclusion)

$(\div 3)$ If $\alpha \notin K$ then $K \div \alpha = K$ (vacuity)

$(\div 4)$ If $\nvdash \alpha$ then $\alpha \notin K \div \alpha$ (success)

$(\div 5)$ If $\alpha \in K$ then $K \subseteq (K \div \alpha) + \alpha$ (recovery)

$(\div 6)$ If $\vdash \alpha \leftrightarrow \beta$ then $K \div \alpha = K \div \beta$ (extensionality)

$(\div 7)$ $K \div \alpha \cap K \div \beta \subseteq K \div \alpha \wedge \beta$ (conjunctive overlap)

$(\div 8)$ If $\alpha \notin K \div \alpha \wedge \beta$ then $K \div \alpha \wedge \beta \subseteq K \div \alpha$ (conjunctive inclusion)

For motivation and interpretation of these postulates see (GŁrdenfors 1988). Any operator $\div$ on K satisfying postulates $(\div 1)$–$(\div 6)$ is called a basic AGM contraction operator. The *supplementary postulates* $(\div 7)$ and $(\div 8)$ specify properties of composite belief contraction operators, which involve contraction by conjunction of sentences. Indeed, AGM also provides a third composite contraction postulate:

> Either $K \div \alpha \wedge \beta = K \div \alpha$, or
> $K \div \alpha \wedge \beta = K \div \beta$, or (conjunctive factoring)
> $K \div \alpha \wedge \beta = K \div \alpha \cap K \div \beta$.

A basic AGM contraction operator that satisfies conjunctive factoring, also satisfies both $(\div 7)$ and $(\div 8)$, and vice versa (Alchourrn, GŁrdenfors, and Makinson 1985).

Some of our beliefs are more *epistemically entrenched* than others, making them harder to give up. Based on this intuition, Grdenfors (1988) introduced epistemic entrenchment, and defined the properties of an order relation $\leq$ between sentences. Grdenfors and Makinson (1988) show that an AGM contraction operator $\div$ can be constructed using a $\leq$ relation, and that, conversely, an epistemic entrenchment relation $\leq$ can be constructed using an AGM contraction operator $\div$.

Turning now to belief base contraction, as with Hansson (1999), we assume that an operation needs to at least satisfy success and inclusion to be considered a belief contraction.

Definition 1. An operator $\div$ for a set B is an operator of contraction if and only if it satisfies success and inclusion.

Some other contraction axioms are listed below:

$B \cap \mathrm{Cn}(B \div \alpha) \subseteq B \div \alpha$ (relative closure)

If $\alpha \in \mathrm{Cn}(B')$ iff $\beta \in \mathrm{Cn}(B')$ for all $B' \subseteq B$ then $B \div \alpha = B \div \beta$ (uniformity)

If $\beta \in B$ and $\beta \notin B \div \alpha$ then (relevance)
$\alpha \notin \mathrm{Cn}(B')$ and $\alpha \in \mathrm{Cn}(B' \cup \{\beta\})$ for some $B \div \alpha \subseteq B' \subseteq B$

If $\beta \in B$ and $\beta \notin B \div \alpha$ then (core-retainment)
$\alpha \notin \mathrm{Cn}(B')$ and $\alpha \in \mathrm{Cn}(B' \cup \{\beta\})$ for some $B' \subseteq B$

Different combinations of the contraction axioms specify different contraction operations. In particular, we note the following:

Partial meet contraction: $(\div 1)$–$(\div 6)$ (Alchourrn, GŁrdenfors, and Makinson 1985); or success, inclusion, relevance and uniformity (Hansson 2003).

Kernel contraction: success, inclusion, core-retainment and uniformity (Hansson 1995).

Saturated (or smooth) kernel contraction: success, inclusion, core-retainment, uniformity and relative closure (Hansson 1995).

Belief Change and Dependence

Farias del Cerro and Herzig (1996) formalized the notion of dependence and its connection with belief change, in a similar approach to Gärdenfors and Makinson (1988) for epistemic entrenchment. To formalize dependence, they investigate a binary relation $\rightsquigarrow$ on formulas. $\alpha \rightsquigarrow \beta$ reads as "β depends on α" (or equivalently "α is relevant to β"). Independence, then, is denoted by $\not\rightsquigarrow$, which is the complement of $\rightsquigarrow$, so $\alpha \not\rightsquigarrow \beta$ reads as "β is independent of α" (or "α is irrelevant to β"). They provide the following axiomatization of dependence.

If $\alpha \leftrightarrow \beta$ and $\alpha \rightsquigarrow \delta$ then $\beta \rightsquigarrow \delta$ (LE^l)

If $\alpha \leftrightarrow \beta$ and $\delta \rightsquigarrow \alpha$ then $\delta \rightsquigarrow \beta$ (LE^r)

If $\alpha \wedge \beta \rightsquigarrow \delta$ then $\alpha \rightsquigarrow \delta$ or $\beta \rightsquigarrow \delta$ (CCI^l)

If $\delta \rightsquigarrow \alpha \wedge \beta$ then $\delta \rightsquigarrow \alpha$ or $\delta \rightsquigarrow \beta$ (CCI^r)

$\alpha \in K$ iff either $\vdash \alpha$ or $\alpha \rightsquigarrow \beta$ for some β (Def-K)

If $\alpha \rightsquigarrow \beta$ then $\alpha \rightsquigarrow \alpha$ (Cond-ID)

If $\vdash \alpha \vee \beta$ then $\alpha \not\rightsquigarrow \beta$ (Disj)

If $\alpha \rightsquigarrow \delta$ and $\alpha \wedge \beta \rightsquigarrow \alpha$ then $\alpha \wedge \beta \rightsquigarrow \delta$ (CCD_0^l)

If $\delta \rightsquigarrow \alpha$ and $\beta \rightsquigarrow \beta$ then $\delta \rightsquigarrow \alpha \wedge \beta$ (CCD_0^r)

The following are also derivable principles:

If $\alpha \rightsquigarrow \delta$ and $\beta \rightsquigarrow \delta$ then $\alpha \wedge \beta \rightsquigarrow \delta$. ($\mathsf{CCD}^l$)

If $\delta \rightsquigarrow \alpha$ and $\delta \rightsquigarrow \beta$ then $\delta \rightsquigarrow \alpha \wedge \beta$. ($\mathsf{CCD}^r$)

For motivation and interpretation of these postulates, please refer to their work. Here, it suffices to note that Keynes' CCD and Grdenfors' CCI are represented by CCD^l and CCI^l, respectively.

FH's dependence relation is defined as follows:

Definition 2. A relation $\rightsquigarrow$ is a *dependence* relation if and only if it satisfies the axioms LE^l, LE^r, CCI^l, CCI^r, Def-K, Cond-ID, Disj, CCD_0^l and CCD_0^r.

For studying the relationship between dependence and belief change, FH use the following *preservation criterion* as their guiding principle:

> "If a belief state is revised by a sentence A, then all sentences in K that are independent of the validity of A should be retained in the revised state of belief" (GŁrdenfors 1990).

For example, if $\beta \in K$ to begin with, but $\beta \notin K \div \alpha$, then we can say that β depends on α, or $\alpha \rightsquigarrow \beta$.

Similar to epistemic entrenchment, to provide the connection between dependence and contraction, FH introduce two conditions, namely, $\mathsf{Cond}\rightsquigarrow$ and $\mathsf{Cond}\div$.

$\alpha \rightsquigarrow \beta$ iff $\beta \in K$ and $\beta \notin K \div \alpha$. ($\mathsf{Cond}\rightsquigarrow$)

This equivalence condition allows one to define $\rightsquigarrow$ based on a given AGM contraction operation $\div$ for belief set K.

Theorem 3 (FH). *Given two relations $\rightsquigarrow$ and $\div$ such that $\mathsf{Cond}\rightsquigarrow$ holds, if $\div$ is an AGM contraction, then $\rightsquigarrow$ is a dependence relation.*

The next condition allows defining an AGM contraction operation $\div$, given a dependence relation $\rightsquigarrow$.

$\beta \in K \div \alpha$ iff either $\vdash \beta$ or $\beta \rightsquigarrow \beta$ and $\alpha \not\rightsquigarrow \beta$. ($\mathsf{Cond}\div$)

An AGM contraction operation $\div$ is defined with respect to some belief set K. Thus, $\div$ obtained via $\mathsf{Cond}\div$ also requires an associated K to be specified. For this purpose, FH provide the following definition for the belief set $K_\rightsquigarrow$:

$$K_\rightsquigarrow = \{\alpha \mid \vdash \alpha \text{ or } \alpha \rightsquigarrow \beta \text{ for some } \beta\}.$$

For the sake of brevity, FH simply use K to refer to $K_\rightsquigarrow$ afterwards. The next theorem defines a contraction operation, given a dependence relation using $\mathsf{Cond}\div$.

Theorem 4 (FH). *Given two relations $\rightsquigarrow$ and $\div$ such that $\mathsf{Cond}\div$ holds, if $\rightsquigarrow$ is a dependence relation then $\div$ is an AGM contraction.*

As the last step, FH need to complete the link between AGM contraction and dependence using a characterization theorem to state that for any two arbitrary relations $\rightsquigarrow$ and $\div$ that satisfy $\mathsf{Cond}\rightsquigarrow$, $\div$ is an AGM contraction iff $\rightsquigarrow$ is a dependence relation. To achieve this, however, it turns out that they first have to make the following assumption:

Remark 5. In order to establish an axiomatic characterization based on $\mathsf{Cond}\rightsquigarrow$, it is assumed that the relation $\div$ satisfies inclusion, $K \div \alpha \subseteq K$.

Theorem 6 (FH). *Let two relations $\rightsquigarrow$ and $\div$ be such that $\div$ satisfies* inclusion *and that $\mathsf{Cond}\rightsquigarrow$ holds. Then $\div$ is an AGM contraction if and only if $\rightsquigarrow$ is a dependence relation.*

This completes FH's work. Also another characterization theorem that they provide for a weaker dependence relation proves to be useful in our study.

Theorem 7 (FH). *Let two relations $\rightsquigarrow$ and $\div$ be such that $\div$ satisfies* inclusion *and that $\mathsf{Cond}\rightsquigarrow$ holds. Then $\div$ is a basic AGM contraction satisfying $(\div 1)$–$(\div 6)$ if and only if $\rightsquigarrow$ is a dependence relation satisfying LE^l, LE^r, CCI^r, Def-K, Cond-ID, Disj and CCD_0^r.*

Belief Change and Base Dependence

Overview and Problem Definition

Gärdenfors' preservation criterion, as with FH, lays the foundation of the present work. Our work is another attempt to connect notions of dependence and belief change, but using belief bases instead of belief sets.

As discussed, belief bases are a generalization of belief sets. Hence, it seems natural to anticipate that base dependence also be a generalization of FH's dependence relation. Furthermore, base dependence should ideally be a *reversible generalization* of dependence. That is, where a base dependence relation corresponds to a belief set, the base dependence relation should reduce to FH's dependence.

Finally, one significant feature of FH's formalism is that it adheres to Keynes' CCD and Gärdenfors' CCI. We would like to preserve this characteristic of FH's dependence while generalizing it to base dependence. It turns out that our base dependence formalism meets a third maxim, CCDF, that implies both CCD and CCI.

Characteristics of an Anticipated Solution

A formalization of Gärdenfors' preservation criterion that involves belief bases instead of belief sets should still retain the general scheme of FH's work. More specifically, we need an *axiomatization of base dependence*, as well as a suitable corresponding *belief base contraction*. We also need a *conditional, similar to FH's* Cond$\rightsquigarrow$, that allows construction of a base dependence given a base contraction, and another *conditional, similar to FH's* Cond$\div$, that allows construction of a base contraction using a base dependence.

As discussed, base dependence should be a reversible generalization of FH's dependence. Naturally, an appropriate base contraction to correspond to base dependence must also be a *reversible generalization of AGM contraction*.

Finally, our base dependence and its corresponding base contraction must be such that CCD and CCI (and ideally CCDF) are satisfied.

These characteristics are pictured in Figure 1, located toward the end of our paper. The remainder of this work is to establish the background concepts and theorems necessary to justify the diagram in Figure 1.

Base Dependence

The meaning of "dependence" in base dependence is the same as what Fariñas and Herzig (and Grdenfors) studied, which refers to the dependence or relevance of logical statements towards one another. Using their notation, we read $\alpha \rightsquigarrow \beta$ as "β depends on α" or "doubting in α leads to doubting in β."

In FH's study, dependence can only happen between (contingent) sentences from K:

$$\text{If } \alpha \rightsquigarrow \beta \text{ then } \alpha \in K \text{ and } \beta \in K.$$

If B is a base for K, $K = \mathrm{Cn}(B)$, then we have:

$$\text{If } \alpha \rightsquigarrow \beta \text{ then } \alpha \in \mathrm{Cn}(B) \text{ and } \beta \in \mathrm{Cn}(B). \quad (1)$$

One way to generalize the dependence relation $\rightsquigarrow$ is to make α or β be from B instead of $\mathrm{Cn}(B)$. Therefore, our sought-for *base dependence* should somehow involve formulas explicitly mentioned in a belief base, thus the name.

Using $\overset{\sim}{\rightsquigarrow}$ to denote base dependence, we read $\alpha \overset{\sim}{\rightsquigarrow} \beta$ as "β *base-depends* on α," which is the same as $\alpha \rightsquigarrow \beta$ except that $\alpha \overset{\sim}{\rightsquigarrow} \beta$ also implies that α or β or both are formulas in the base. Now, we need to decide which one of these three alternatives should be the case.

$$\text{If } \alpha \overset{\sim}{\rightsquigarrow} \beta \text{ then } \alpha \in B \text{ and } \beta \in \mathrm{Cn}(B) \quad (2a)$$
$$\text{If } \alpha \overset{\sim}{\rightsquigarrow} \beta \text{ then } \alpha \in \mathrm{Cn}(B) \text{ and } \beta \in B \quad (2b)$$
$$\text{If } \alpha \overset{\sim}{\rightsquigarrow} \beta \text{ then } \alpha \in B \text{ and } \beta \in B. \quad (2c)$$

We believe the third alternative (2c), requiring both α and β to be in B, appears to be too strong to try first. The second alternative (2b) offers a more interesting semantics, compared to the first alternative (2a). The second alternative means that "doubting in α leads to doubting in β from the base." It allows us to study how the statements in the *base* depend on, or are susceptible to, changes of other statements. Stated in terms of belief change, it means contracting B by any formula α from the *infinite* set $\mathrm{Cn}(B)$ can result in removal of β from the *finite* set $B \setminus B \div \alpha$. On the other hand, the first alternative (2a) means contracting B by any formula α from the usually *finite* set B can result in removal of β from the *infinite* set $\mathrm{Cn}(B) \setminus \mathrm{Cn}(B \div \alpha)$.

Therefore, although the alternatives (2a) and (2c) remain open and may be found useful in other studies, we proceed with (2b), and from now on we assume $\alpha \overset{\sim}{\rightsquigarrow} \beta$ requires that $\beta \in B$:

$$\text{If } \alpha \overset{\sim}{\rightsquigarrow} \beta \text{ then } \beta \in B. \quad (3)$$

It turns out that (3) does not need to be explicitly specified as an axiom. Rather, it will be implied by other axioms and conditions for base dependence which will be put forward in the upcoming sections (e.g., Def-B and Cond$\overset{\sim}{\rightsquigarrow}$).

Example 8. Assume Mary believes that p, q and $q \to r$; i.e., $B = \{p,\ q,\ q \to r\}$. By implication, she also believes that r, as it is entailed by q and $q \to r$. Now, say, for some reason, she starts to doubt that r is true. Consequently, this leads her to also doubt either q, or $q \to r$, or even both. Thus we know that at least one of $r \overset{\sim}{\rightsquigarrow} q$ or $r \overset{\sim}{\rightsquigarrow} (q \to r)$ hold, and that $r \not\rightsquigarrow p$ and $r \not\rightsquigarrow r$. □

Mutual Construction

From Contraction to Base Dependence As discussed earlier, FH use Cond$\rightsquigarrow$ to construct a dependence relation via a given AGM contraction. Cond$\rightsquigarrow$ can be straightforwardly transformed to an equivalent base-generated representation. That is, if $K = \mathrm{Cn}(B)$ and $\div$ is a base-generated contraction, which exists for any given AGM contraction (Hansson 1993), then Cond$\rightsquigarrow$ can be stated as follows:

$$\alpha \rightsquigarrow \beta \text{ iff } \beta \in \mathrm{Cn}(B) \text{ and } \beta \notin \mathrm{Cn}(B \div \alpha). \quad (\text{Cond}\rightsquigarrow)$$

Note that we have used the same name "Cond$\rightsquigarrow$" here since this is only an alternative representation. This new, base-generated representation makes it easier to compare and contrast Cond$\rightsquigarrow$ with the corresponding conditions for belief bases that will be introduced shortly.

If $\alpha \rightsquigarrow \beta$ then β is retracted as a result of α's contraction, so $\beta \in [\mathrm{Cn}(B) \setminus \mathrm{Cn}(B \div \alpha)]$. For base dependence, we assume that (3) holds: if $\alpha \overset{\sim}{\rightsquigarrow} \beta$ then $\beta \in B$. Thus, it is intuitively appealing to say that if $\alpha \overset{\sim}{\rightsquigarrow} \beta$, then β is retracted from B as a result of α's contraction, so $\beta \in [B \setminus B \div \alpha]$. With this intuition in mind, we propose the following to correspond to FH's Cond$\rightsquigarrow$:

$$\alpha \overset{\sim}{\rightsquigarrow} \beta \text{ iff } \beta \in B \text{ and } \beta \notin B \div \alpha. \quad (\text{Cond}\overset{\sim}{\rightsquigarrow})$$

Notice that in common with FH, if α is a tautology, no formula contributes to its truth, and $\alpha \overset{\sim}{\rightsquigarrow} \beta$ cannot hold for any β. Also, if β is not in B, then $\alpha \overset{\sim}{\rightsquigarrow} \beta$ cannot hold for any α by definition. Therefore, both $\beta \rightsquigarrow \beta$ and $\beta \overset{\sim}{\rightsquigarrow} \beta$ imply that β is contingent, and the latter additionally implies that β is in B. This will prove useful in the next section when constructing contraction operations using (base) dependence, via Cond$\div$ and Cond$\overset{\sim}{\div}$.

From Base Dependence to Contraction We first provide a simplifying notation $\vdash_B$ to help represent *tautologies present in the base*.

Definition 9. Given a base B and an entailment relation $\vdash$, the *base entailment* relation $\vdash_B$ is defined as follows: $A \vdash_B \beta$ if and only if $\beta \in B$ and $A \vdash \beta$.

A useful special case is when $A = \emptyset$. For example, $\vdash_B \beta$ means β is a tautology in the base: $\beta \in B$ and $\vdash \beta$. One important usage is to help handling tautologies in base dependence axioms. Such axioms are primarily concerned with contingencies, but they have to also deal with tautologies, usually as exceptional cases.

Next, we need to reconstruct belief bases, given base dependence relations. We saw that FH provide the following:

$$K_{\rightsquigarrow} = \{\alpha \mid \vdash \alpha \text{ or } \alpha \rightsquigarrow \beta \text{ for some } \beta\}.$$

Note that we could swap the role of α and β above and obtain the same results:

$$K_{\rightsquigarrow} = \{\beta \mid \vdash \beta \text{ or } \alpha \rightsquigarrow \beta \text{ for some } \alpha\}.$$

Similarly, a base dependence relation $\bar{\rightsquigarrow}$ is associated with a belief base B. Thus, it should be possible to recreate the associated belief base B via a given $\bar{\rightsquigarrow}$ relation:

$$B_{\bar{\rightsquigarrow}} = \{\beta \mid \alpha \bar{\rightsquigarrow} \beta \text{ for some } \alpha\}.$$

Note, however, that one caveat is that $B_{\bar{\rightsquigarrow}}$ will not contain any *tautologies* that may be in B. Still, we can say that B and $B_{\bar{\rightsquigarrow}}$ are equivalent for most practical purposes. Also, their closure is obviously equivalent:

$$\mathrm{Cn}(B_{\bar{\rightsquigarrow}}) = \mathrm{Cn}(B).$$

If in addition to the base dependence relation $\bar{\rightsquigarrow}$, we are also given $\vdash_B$ that identifies tautologies in the base, then we can have the following, which guarantees that $B_{\bar{\rightsquigarrow}} = B$:

$$B_{\bar{\rightsquigarrow}} = \{\beta \mid \vdash_B \beta \text{ or } \alpha \bar{\rightsquigarrow} \beta \text{ for some } \alpha\}$$

In the rest of this work, we assume $B_{\bar{\rightsquigarrow}} = B$. In the worst case scenario, there are tautologies in B and only $\bar{\rightsquigarrow}$ is given, so the tautologies in B are not present in $B_{\bar{\rightsquigarrow}}$.

Now for $\mathsf{Cond}\bar{\div}$, again we start with $\mathsf{Cond}\div$. As in the case of $\mathsf{Cond}\rightsquigarrow$, we present a straightforward transformation to the equivalent base-generated operation. Again, we reuse the equation's name, "$\mathsf{Cond}\div$":

$$\beta \in \mathrm{Cn}(B \div \alpha) \text{ iff either } \vdash \beta \text{ or } \beta \rightsquigarrow \beta \text{ and } \alpha \not\rightsquigarrow \beta. \qquad (\mathsf{Cond}\div)$$

$\mathsf{Cond}\div$ says $\beta \in \mathrm{Cn}(B \div \alpha)$ means either that β is a tautology, or that β is a contingent truth, $\beta \rightsquigarrow \beta$, but contraction by α does not lead to retraction of β, meaning $\alpha \not\rightsquigarrow \beta$.

To adapt this for belief bases, we need something along the line of the following: $\beta \in B \div \alpha$ means either that β is a tautology in B, $\vdash_B \beta$, or that β is a contingent truth in B, $\beta \bar{\rightsquigarrow} \beta$, but contraction by α does not lead to retraction of β from B, $\alpha \not\rightsquigarrow \beta$:

$$\beta \in B \div \alpha \text{ iff either } \vdash_B \beta \text{ or } \beta \bar{\rightsquigarrow} \beta \text{ and } \alpha \not\rightsquigarrow \beta. \qquad (\mathsf{Cond}\bar{\div})$$

Basic Postulates of Base Dependence

A goal of this work is to provide an axiomatization of base dependence as a generalization of FH's dependence. It turns out that some of base dependence axioms closely resemble dependence axioms (e.g. $\mathsf{Cond\text{-}ID}^B$), and some remain valid and derivable but are no longer needed as axioms (e.g. Disj^B). Yet, there are also some other axioms offered for base dependence (e.g. redundancy) that are not similar to any of the dependence axioms. The following are the basic postulates of base dependence.

$\beta \in B$ iff either $\vdash_B \beta$ or $\alpha \bar{\rightsquigarrow} \beta$ for some α. (Def-B)

If $\alpha \bar{\rightsquigarrow} \beta$ then $\beta \bar{\rightsquigarrow} \beta$. ($\mathsf{Cond\text{-}ID}^B$)

If $\alpha \in \mathrm{Cn}(B')$ iff $\beta \in \mathrm{Cn}(B')$ for all $B' \subseteq B$ then $\alpha \bar{\rightsquigarrow} \delta$ iff $\beta \bar{\rightsquigarrow} \delta$. (conjugation)

If $\alpha \bar{\rightsquigarrow} \beta$ then (contribution)
$\alpha \notin \mathrm{Cn}(B')$ and $\alpha \in \mathrm{Cn}(B' \cup \{\beta\})$ for some $B' \subseteq B$.

If $\alpha \in \mathrm{Cn}(B')$ and $B' \subseteq B$ then (modularity)
either $\vdash \alpha$ or $\alpha \bar{\rightsquigarrow} \beta$ for some $\beta \in B'$.

If $\beta \in \mathrm{Cn}(B')$ and $B' \subseteq B$ then (redundancy)
either $\alpha \not\rightsquigarrow \beta$ or $\alpha \bar{\rightsquigarrow} \delta$ for some $\delta \in B'$.

A brief description of these axioms is as follows:

(Def-B) To say that β is in the base, $\beta \in B$, is equivalent to saying either that β is a tautology in the base, $\vdash_B \beta$, or that it is a contingency in the base. The contingent truth of β then has to be inferentially relevant to some (contingent) formula α (where α could be β), so $\alpha \bar{\rightsquigarrow} \beta$.

($\mathsf{Cond\text{-}ID}^B$) The inferential relevance between α and β means that neither is a tautology. Also, β is in the base because $\alpha \bar{\rightsquigarrow} \beta$. Thus, β is a contingency in the base, or $\beta \bar{\rightsquigarrow} \beta$.

(conjugation) When α and β are logically equivalent, $\vdash \alpha \leftrightarrow \beta$, there is inferential relevance between α and δ from B if and only if there is inferential relevance between β and δ. This makes sense not only for α and β, but also for the formulas that are true *just because* α or β are true.

(contribution) This axiom says that if β from B is inferentially relevant to α then β must somehow contribute to the justification of α.

(modularity) Consider a subset $B' \subseteq B$ that implies α, $\alpha \in \mathrm{Cn}(B')$. This could be because α is a tautology. But when $\nvdash \alpha$, there is some β from the same subset B' that base-depends on α.

(redundancy) To consider the principal case of redundancy, assume that: $\beta \in \mathrm{Cn}(B')$ and $B' \subseteq B$ and $\alpha \bar{\rightsquigarrow} \beta$. When $\beta \in B'$, $\alpha \bar{\rightsquigarrow} \delta$ for some $\delta \in B'$ trivially holds because then δ could be β which means $\alpha \bar{\rightsquigarrow} \beta$, which is assumed. When $\beta \notin B'$, there is some redundancy in the base B because on the one hand $\beta \in B$ (as $\alpha \bar{\rightsquigarrow} \beta$), and on the other hand there is $B' \subseteq B$ such that $\beta \notin B'$ but $\beta \in \mathrm{Cn}(B')$. Thus, in order for $\alpha \bar{\rightsquigarrow} \beta$ to hold, $\alpha \bar{\rightsquigarrow} \delta$ should also hold at least for one formula $\delta \in B'$.

The following principles, which are derivable from the above axioms, also have counterpart in FH's framework:

$$\text{If } \vdash \alpha \vee \beta \text{ then } \alpha \not\sim \beta. \qquad (\mathsf{Disj}^B)$$

$$\text{If } \vdash \alpha \leftrightarrow \beta \text{ and } \alpha \mathrel{\tilde{\rightarrow}} \delta \text{ then } \beta \mathrel{\tilde{\rightarrow}} \delta. \qquad (\mathsf{LE}^B)$$

Example 10. Assume that $B = \{p \leftrightarrow q, p \vee q, p\}$, and that $\mathrel{\tilde{\rightarrow}}$ is a relation such that $(p \wedge q) \mathrel{\tilde{\rightarrow}} (p \vee q)$ and $(p \wedge q) \not\mathrel{\tilde{\rightarrow}} p$ hold. We show that $\mathrel{\tilde{\rightarrow}}$ violates redundancy. Let $B' = \{p\}$. Clearly $B' \subseteq B$ and $p \vee q \in \mathrm{Cn}(B')$, and, by assumption, $(p \wedge q) \mathrel{\tilde{\rightarrow}} (p \vee q)$. Thus, by redundancy, $(p \wedge q) \mathrel{\tilde{\rightarrow}} \delta$ for some $\delta \in B'$. Since $B' = \{p\}$, δ has to be p, so $(p \wedge q) \mathrel{\tilde{\rightarrow}} p$. This contradicts the initial assumption that $(p \wedge q) \not\mathrel{\tilde{\rightarrow}} p$. □

Definition 11. A relation $\mathrel{\tilde{\rightarrow}}$ is a *base dependence* if and only if it satisfies the axioms Def-B, Cond-ID^B, conjugation, contribution, modularity and redundancy.

Notice that so far we have not specified any criteria on how to handle conjunctions, which we will do shortly.

Saturated Kernel Contraction and Base Dependence

As discussed, we need a base contraction to correspond to base dependence. We also saw that a suitable candidate base contraction must be a reversible generalization of AGM contraction. Indeed, saturated kernel contraction is a subclass of kernel contraction that is a reversible generalization of basic AGM contraction. That is, saturated kernel contraction is a base contraction, and when it corresponds to a belief set, it is equivalent to AGM partial meet contraction (Hansson 1995).

In this section, we show that indeed saturated kernel contraction and base dependence have a mutual correspondence.

From Base Dependence to Contraction To construct a contraction operator $\div$, assume all the following are present: a base dependence relation $\mathrel{\tilde{\rightarrow}}$ (Definition 11), a list of tautologies present in the base $T \subseteq B$ where $T = \{\beta \mid \vdash_{_B} \beta\}$, and the Cond$\mathrel{\tilde{\div}}$.

We do *not* need to assume that B is provided because it can be obtained using $\mathrel{\tilde{\rightarrow}}$ and $\vdash_{_B}$.

Theorem 12 states that the contraction operator $\div$ obtained from $\mathrel{\tilde{\rightarrow}}$ is indeed a saturated kernel contraction.

Theorem 12 (Base Dependence to Contraction). *Given relations $\mathrel{\tilde{\rightarrow}}$ and $\div$ for base B such that* Cond$\mathrel{\tilde{\div}}$ *holds, if $\mathrel{\tilde{\rightarrow}}$ is a base dependence, then $\div$ is a saturated kernel contraction.*

To prove this theorem, we assume the Cond$\mathrel{\tilde{\div}}$ and postulates of base dependence hold, and we show one by one that the postulates of saturated kernel contraction also hold.

From Contraction to Base Dependence This subsection shows how to obtain a base dependence $\mathrel{\tilde{\rightarrow}}$ relation given a saturated kernel contraction operator $\div$. We assume the following are present: a saturated kernel contraction operator $\div$, and the Cond$\mathrel{\tilde{\rightarrow}}$. Theorem 13 states that, given the above assumptions, all axioms of base dependence $\mathrel{\tilde{\rightarrow}}$ relation are satisfied.

Theorem 13 (Contraction to Base Dependence). *Given relations $\mathrel{\tilde{\rightarrow}}$ and $\div$ for base B such that* Cond$\mathrel{\tilde{\rightarrow}}$ *holds, if $\div$ is a saturated kernel contraction, then $\mathrel{\tilde{\rightarrow}}$ is a base dependence.*

For this theorem, assuming the Cond$\mathrel{\tilde{\rightarrow}}$ and axioms of saturated kernel contraction, we show one by one that the properties of base dependence hold.

Axiomatic Characterization We now need an axiomatic characterization theorem. We also adopt the FH assumption in Remark 5, which means that inclusion needs to be assumed in the characterization theorem. The rationale for this assumption is as follows. When constructing the $\mathrel{\tilde{\rightarrow}}$ relation using a contraction operation via Cond$\mathrel{\tilde{\rightarrow}}$, the set of all β such that $\alpha \mathrel{\tilde{\rightarrow}} \beta$ is equal to those $\beta \in B$ and $\beta \notin B \div \alpha$, or using set difference notation $\beta \in B \setminus (B \div \alpha)$. We know as a matter of fact that $B \div \alpha \subseteq B$ holds because, by Definition 1, any contraction operator satisfies inclusion. However, even if, for the sake of argument, $\div$ did not satisfy inclusion and there were some statements in $B \div \alpha$ that were not in B, such statements would have been lost in the set difference $\beta \in B \setminus (B \div \alpha)$. That, in turn, means that to use $\mathrel{\tilde{\rightarrow}}$ to construct a contraction $\div$ via Cond$\mathrel{\tilde{\rightarrow}}$, we do not have enough information to prove or disprove inclusion. Instead, we have to assume that $\div$ already satisfies inclusion. Since all contraction operations satisfy inclusion, this assumption is not a serious loss of generality.

Theorem 14 (Characterization). *Let the relations $\mathrel{\tilde{\rightarrow}}$ and $\div$ for base B be such that $\div$ satisfies* inclusion *and that* Cond$\mathrel{\tilde{\rightarrow}}$ *holds. Then, $\div$* is a saturated kernel contraction if and only if $\mathrel{\tilde{\rightarrow}}$ is a base dependence.

To prove this characterization theorem, we show that in presence of inclusion, Cond$\mathrel{\tilde{\rightarrow}}$ entails Cond$\mathrel{\tilde{\div}}$. Thus, assuming inclusion and Cond$\mathrel{\tilde{\rightarrow}}$, based on Theorems 12 and 13, saturated kernel contraction and base dependence are logically equivalent.

Conjunction Criterion of Dependence Factoring

We have seen Keynes' CCD and Grdenfors' CCI, and the respective dependence axioms CCD^I and CCI^I. Likewise, we have:

$$\text{If } \alpha \mathrel{\tilde{\rightarrow}} \delta \text{ and } \beta \mathrel{\tilde{\rightarrow}} \delta \text{ then } \alpha \wedge \beta \mathrel{\tilde{\rightarrow}} \delta. \qquad (\mathsf{CCD}^B)$$

$$\text{If } \alpha \wedge \beta \mathrel{\tilde{\rightarrow}} \delta \text{ then } \alpha \mathrel{\tilde{\rightarrow}} \delta \text{ or } \beta \mathrel{\tilde{\rightarrow}} \delta. \qquad (\mathsf{CCI}^B)$$

CCD and CCI state intuitions regarding dependence on conjunctions in the form of *conditional* statements. One wonders whether it is possible to capture such intuitions regarding dependence on conjunctions using *equivalences*. Such a statement would have to capture different cases. That is, for any reasonable dependence relation, at least one of the following statements hold:

Case 1: The set of formulas that depend on $\alpha \wedge \beta$ is *the same as* the set of those that depend on α

Case 2: The set of formulas that depend on $\alpha \wedge \beta$ is *the same as* the set of those that depend on β

Case 3: The set of formulas that depend on $\alpha \wedge \beta$ is *the same as* the set of those that depend on α *or* depend on β

Using set notation, these cases can be stated as follows:

$$\begin{aligned} \text{Either } \{\delta \mid \alpha \wedge \beta \mathrel{\tilde{\rightarrow}} \delta\} &= \{\delta \mid \alpha \mathrel{\tilde{\rightarrow}} \delta\}, \text{ or} \\ \{\delta \mid \alpha \wedge \beta \mathrel{\tilde{\rightarrow}} \delta\} &= \{\delta \mid \beta \mathrel{\tilde{\rightarrow}} \delta\}, \text{ or} \\ \{\delta \mid \alpha \wedge \beta \mathrel{\tilde{\rightarrow}} \delta\} &= \{\delta \mid \alpha \mathrel{\tilde{\rightarrow}} \delta\} \cup \{\delta \mid \beta \mathrel{\tilde{\rightarrow}} \delta\} \end{aligned} \qquad (4)$$

or equivalently,

$$\begin{array}{rl} \text{Either } [\alpha \wedge \beta \mathrel{\tilde{\hookrightarrow}} \delta_1 & \text{iff } \alpha \mathrel{\tilde{\hookrightarrow}} \delta_1], \text{ or} \\ [\alpha \wedge \beta \mathrel{\tilde{\hookrightarrow}} \delta_2 & \text{iff } \beta \mathrel{\tilde{\hookrightarrow}} \delta_2], \text{ or} \\ [\alpha \wedge \beta \mathrel{\tilde{\hookrightarrow}} \delta_3 & \text{iff } \alpha \mathrel{\tilde{\hookrightarrow}} \delta_3 \text{ or } \beta \mathrel{\tilde{\hookrightarrow}} \delta_3]. \end{array} \quad (\mathsf{CCDF}^B)$$

Each line of CCDF^B needs to use a unique variable name δ_1, δ_2 and δ_3 because, in each line of (4), $\{\delta \mid \alpha \wedge \beta \mathrel{\tilde{\hookrightarrow}} \delta\}$ refers to a different set.

CCDF^B is a formalization of the intuition expressed in the three cases above, which we restate more concisely as follows, calling it the *Conjunction Criterion of Dependence Factoring*, CCDF:

> The set of all formulas that depend on $\alpha \wedge \beta$ is the same as the set of all formulas that depend on α, or on β, or on either of them. (CCDF)

Indeed, CCDF may be considered as a third maxim for dependence of conjunctions in addition to Keynes' CCD and Grdenfors' CCI.

As a side note, although it seems that the third clause of CCDF^B should be redundant, in light of the first two, in fact it isn't.

Example 15. Assume α, β, θ_1, θ_2 and θ_3 are formulas and $\tilde{\hookrightarrow}$ is a relation such that

$$\begin{array}{lll} \alpha \wedge \beta \mathrel{\tilde{\hookrightarrow}} \theta_1 & \alpha \mathrel{\tilde{\hookrightarrow}} \theta_1 & \beta \not\mathrel{\tilde{\hookrightarrow}} \theta_1 \\ \alpha \wedge \beta \mathrel{\tilde{\hookrightarrow}} \theta_2 & \alpha \not\mathrel{\tilde{\hookrightarrow}} \theta_2 & \beta \mathrel{\tilde{\hookrightarrow}} \theta_2 \\ \alpha \wedge \beta \mathrel{\tilde{\hookrightarrow}} \theta_3 & \alpha \mathrel{\tilde{\hookrightarrow}} \theta_3 & \beta \mathrel{\tilde{\hookrightarrow}} \theta_3. \end{array}$$

Clearly, $\tilde{\hookrightarrow}$ violates the first two clauses of CCDF^B, but not the third one. This may be easier to see using (4). Note that $\{\delta \mid \alpha \wedge \beta \mathrel{\tilde{\hookrightarrow}} \delta\} = \{\theta_1, \theta_2, \theta_3\}$, $\{\delta \mid \alpha \mathrel{\tilde{\hookrightarrow}} \delta\} = \{\theta_1, \theta_3\}$ and $\{\delta \mid \beta \mathrel{\tilde{\hookrightarrow}} \delta\} = \{\theta_2, \theta_3\}$, which satisfy the third clause of (4) but not the first two. □

As a second note, CCDF^B is only one way of formalizing CCDF, using base dependence relation; of course, it can also be formalized using FH's dependence relation as shown below, which we call CCDF^l:

$$\begin{array}{rl} \text{Either } [\alpha \wedge \beta \mathrel{\sim} \delta_1 & \text{iff } \alpha \mathrel{\sim} \delta_1], \text{ or} \\ [\alpha \wedge \beta \mathrel{\sim} \delta_2 & \text{iff } \beta \mathrel{\sim} \delta_2], \text{ or} \\ [\alpha \wedge \beta \mathrel{\sim} \delta_3 & \text{iff } \alpha \mathrel{\sim} \delta_3 \text{ or } \beta \mathrel{\sim} \delta_3]. \end{array} \quad (\mathsf{CCDF}^l)$$

Finally, an important observation here is that CCDF^B is a more specific criterion than CCD^B and CCI^B, and it implies both of them.

Theorem 16. *If a relation $\tilde{\hookrightarrow}$ satisfies CCDF^B, then it also satisfies both CCD^B and CCI^B.*

Notice that although Theorem 16 is stated in terms of base dependence $\tilde{\hookrightarrow}$, it does not have to be. Indeed, the theorem (and its proof) may straightforwardly be restated in terms of CCDF that implies both CCD and CCI. As such, the dependence version of CCDF, i.e. CCDF^l, also implies both FH's CCD^l and CCI^l.

The following are all the conjunction criteria and related axioms we have discussed:

Criterion	Dependence	Base Dependence
CCD (Keynes)	CCD^l (FH)	CCD^B
CCI (Grdenfors)	CCI^l (FH)	CCI^B
CCDF	CCDF^l	CCDF^B

We may now state axiomatic characterization and its associated theorems, with the addition of corresponding conjunction criteria as follows.

Theorem 17 (Base Dependence to Contraction). *Given relations $\tilde{\hookrightarrow}$ and $\div$ for base B such that* Cond$\tilde{\div}$ *holds, if $\tilde{\hookrightarrow}$ is a base dependence that satisfies* CCDF^B*, then $\div$ is a saturated kernel contraction that satisfies* conjunctive factoring.

Theorem 18 (Contraction to Base Dependence). *Given relations $\tilde{\hookrightarrow}$ and $\div$ for base B such that* Cond$\tilde{\hookrightarrow}$ *holds, if $\div$ is a saturated kernel contraction that satisfies* conjunctive factoring*, then $\tilde{\hookrightarrow}$ is a base dependence that satisfies* CCDF^B.

Theorem 19 (Main Characterization). *Let the relations $\tilde{\hookrightarrow}$ and $\div$ for base B be such that $\div$ satisfies* inclusion*, $B \div \alpha \subseteq B$, and that* Cond$\tilde{\hookrightarrow}$ *holds: $\alpha \mathrel{\tilde{\hookrightarrow}} \beta$ iff $\beta \in B$ and $\beta \notin B \div \alpha$. Then, $\div$ is a saturated kernel contraction that satisfies* conjunctive factoring *if and only if $\tilde{\hookrightarrow}$ is a base dependence that satisfies* CCDF^B.

Base Dependence as a Reversible Generalization of Dependence

Last, we show that base dependence is a reversible generalization of FH's dependence.

Theorem 20 (Dependence Generalization). *Let relations $\tilde{\hookrightarrow}$, $\sim$ and $\div$ for base B be such that* Cond$\tilde{\hookrightarrow}$ *and* Cond$\sim$ *hold and* inclusion *is satisfied. In the case where B is logically closed,*

(1) *the following are logically equivalent:*

a) *$\tilde{\hookrightarrow}$ is a base dependence, which satisfies* Def-B, Cond-ID^B, conjugation, contribution, modularity *and* redundancy

b) *$\sim$ is a dependence that satisfies* Def-K, Cond-ID, Disj, LE^l, LE^r, CCI^r *and* CCD$^r_0$

c) *$\div$ is a saturated kernel contraction, which satisfies* success, inclusion, core-retainment, uniformity *and* relative closure

d) *$\div$ is a basic AGM contraction, which satisfies* $(\div 1)$–$(\div 6)$

(2) *if any one of* 1.a–1.d *above hold, then $\tilde{\hookrightarrow}$ reduces to $\sim$:* $\alpha \mathrel{\tilde{\hookrightarrow}} \beta$ iff $\alpha \mathrel{\sim} \beta$.

Now everything is in place to extend the formalism for a base dependence relation $\tilde{\hookrightarrow}$ that also satisfies CCDF^B. Satisfying CCDF^B allows $\tilde{\hookrightarrow}$ to meet both CCD and CCI.

Theorem 21 (Dependence Generalization with Conjunction). *Let relations $\tilde{\hookrightarrow}$, $\sim$ and $\div$ for base B be such that* Cond$\tilde{\hookrightarrow}$ *and* Cond$\sim$ *hold and* inclusion *is satisfied. In the case where B is logically closed,*

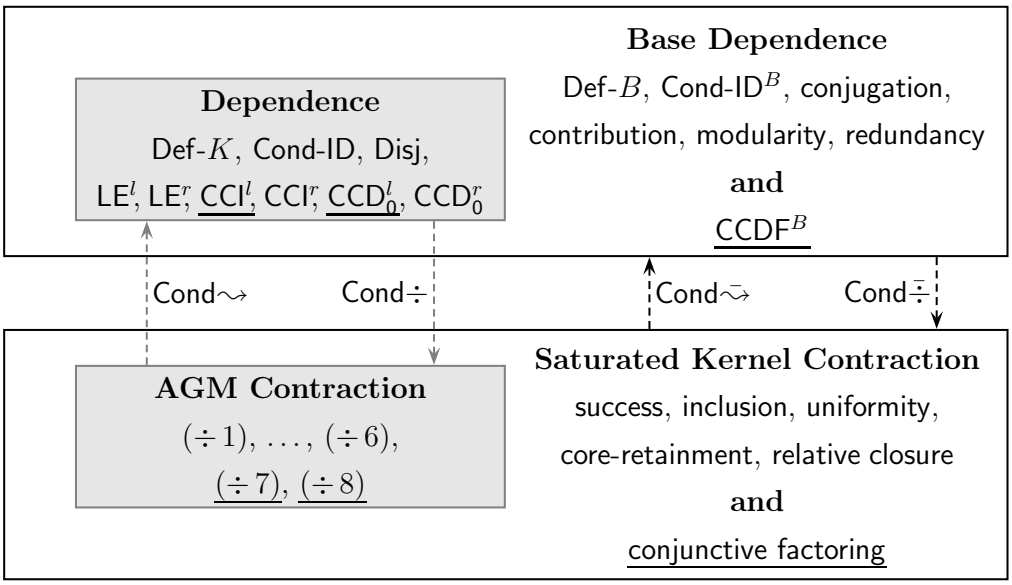


Figure 1: Belief Change and Base Dependence (omitting the underlined axioms results in a weaker characterization)

(1) *the following are logically equivalent:*
   a) $\bar{\rightsquigarrow}$ *is a base dependence that satisfies* Def-B, Cond-ID^B, conjugation, contribution, modularity, redundancy *and* CCDF^B
   b) $\rightsquigarrow$ *is a dependence, which satisfies* Def-K, Cond-ID, Disj, LE^l, LE^r, CCI^l, CCI^r, CCD$^l_0$ *and* CCD$^r_0$
   c) $\div$ *is a saturated kernel contraction that satisfies* success, inclusion, core-retainment, uniformity, relative closure *and* conjunctive factoring
   d) $\div$ *is an AGM contraction, which satisfies* $(\div 1)$–$(\div 6)$, $(\div 7)$ *and* $(\div 8)$

(2) *if any one of* 1.a–1.d *above hold, then* $\bar{\rightsquigarrow}$ *reduces to* $\rightsquigarrow$*:* $\alpha \bar{\rightsquigarrow} \beta$ iff $\alpha \rightsquigarrow \beta$.

This concludes our work. The diagram in Figure 1 captures the key results from Theorems 12-21.

Discussion and Related Work

There are several works that define the concepts of relevance and dependence of formulas. Hansson and Wassermann (2002) propose that these can be classified into two groups. Some authors capture relevance/dependence of formulas through syntactical means such as variable sharing and language splitting, including (Parikh 1999; Chopra and Parikh 2000; Makinson and Kourousias 2006; Kourousias and Makinson 2007; Makinson 2007; Ji, Qi, and Haase 2008; Suntisrivaraporn et al. 2008; Ismail and Kasrin 2010; Wu, Zhang, and Zhang 2011; Perrussel, Marchi, and Zhang 2011; Falappa et al. 2011). Other authors have focused on inferential dependency of formulas, or, in other words, how some formulas deductively contribute to inference of other formulas. Examples of this approach include (Farias del Cerro and Herzig 1996; Hansson and Wassermann 2002; Cuenca Grau, Halaschek-Wiener, and Kazakov 2007), as well as the work reported in the present paper. Typically, syntactical approaches are simpler and computationally more efficient compared to inferential approaches. However, the latter usually provide a more accurate and tighter definition of relevance and dependence than syntactical approaches.

What sets FH's work and our work apart from all other works, is the integration with the theory of belief change. This has an important implication: it provides the most theoretically accurate definition of dependence in the context of belief change. For example, because FH construct their dependence relation using AGM contraction, any other definition of dependence that is put forward to be used in relation to AGM contraction is either as good as their dependence relation or less accurate in capturing dependence of formulas in this context. Of course, being a generalization of FH's work, our framework inherits this property for belief sets, and preserves it for belief bases as well.

Conclusion

With these results, we have provided a formalism of Gärdenfors' preservation criterion such that it generalizes the dependence formalism studied by FH so that it works for belief bases (and belief sets). In the case when a belief base is closed the generalized dependence, base dependence, is equivalent to the original FH dependence relation. While generalizing FH's work, it preserves some of the important characteristics of their study such as Keynes' conjunction criterion for dependence (CCD) and Grdenfors' conjunction criterion for independence (CCI). Additionally, we provide a more specific intuition called *conjunction criterion of dependence factoring*, CCDF, that encompasses both Keynes' CCD and Grdenfors' CCI intuitions.

References

Alchourrn, C. E.; Grdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic* 50:510530.

Chopra, S., and Parikh, R. 2000. Relevance sensitive belief structures. *Annals of Mathematics and Artificial Intelligence* 28(1):259–285.

Cuenca Grau, B.; Halaschek-Wiener, C.; and Kazakov, Y. 2007. History matters: Incremental ontology reasoning using modules. 183–196.

Falappa, M.; Garca, A.; Kern-Isberner, G.; and Simari, G. 2011. Stratified belief bases revision with argumentative inference. *Journal of Philosophical Logic* 133.

Farias del Cerro, L., and Herzig, A. 1996. Belief change and dependence. In *Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge (TARK'96)*, 147–161. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Grdenfors, P., and Makinson, D. 1988. Revisions of knowledge systems using epistemic entrenchment. In *Proceedings of the 2nd conference on Theoretical aspects of reasoning about knowledge*, 83–95. Pacific Grove, California: Morgan Kaufmann Publishers Inc.

Grdenfors, P. 1978. On the logic of relevance. *Synthese* 37(3):351–367.

Grdenfors, P. 1988. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. Cambridge, Mass: MIT Press.

Grdenfors, P. 1990. Belief revision and relevance. In *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*, 349–365. Philosophy of Science Association.

Hansson, S. O., and Wassermann, R. 2002. Local change. *Studia Logica* 70(1):4976.

Hansson, S. O. 1993. Theory contraction and base contraction unified. *The Journal of Symbolic Logic* 58(2):602.

Hansson, S. O. 1995. Kernel contraction. *Journal of Symbolic Logic* 845–859.

Hansson, S. O. 1999. *A Textbook of Belief Dynamics: Theory Change and Database Updating*. Kluwer Academic Publishers.

Hansson, S. O. 2003. A dyadic representation of belief. In Grdenfors, P., ed., *Belief Revision*. Cambridge University.

Ismail, H., and Kasrin, N. 2010. High-level perception as focused belief revision. In *Proceeding of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, 11451146.

Ji, Q.; Qi, G.; and Haase, P. 2008. A relevance-based algorithm for finding justifications of DL entailments. Technical report, Technical report, University of Karlsruhe.

Keynes, J. M. 1921. *A Treatise on Probability*. Courier Dover Publications.

Kourousias, G., and Makinson, D. 2007. Parallel interpolation, splitting, and relevance in belief change. *Journal of symbolic logic* 72(3):994–1002.

Makinson, D., and Kourousias, G. 2006. Respecting relevance in belief change. *Anlisis filosfico* 26(1):5361.

Makinson, D. C. 2007. Propositional relevance through letter-sharing: review and contribution. In Bonanno, G.; Delgrande, J.; Lang, J.; and Rott, H., eds., *Formal models of belief change in rational agents*. Dagstuhl, Germany: Internationales Begegnungs und Forschungszentrum fur Informatik (IBFI).

Oveisi, M. 2013. *Belief Change and Base Dependence*. Ph.D. Dissertation, Simon Fraser University, Canada.

Parikh, R. 1999. Beliefs, belief revision, and splitting languages. *Logic, language and computation* 2:266278.

Perrussel, L.; Marchi, J.; and Zhang, D. 2011. Characterizing relevant belief revision operators. *AI 2010: Advances in Artificial Intelligence* 4251.

Suntisrivaraporn, B.; Qi, G.; Ji, Q.; and Haase, P. 2008. A modularization-based approach to finding all justifications for OWL DL entailments. In *Proceedings of the 3rd Asian Semantic Web Conference on The Semantic Web*, 1–15. Springer.

Wu, M.; Zhang, D.; and Zhang, M. 2011. Language splitting and relevance-based belief change in horn logic. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Belief Change and Semiorders

Pavlos Peppas
Dept of Business Administration
University of Patras
and
Faculty of Engineering and IT
University of Technology, Sydney

Mary-Anne Williams
Faculty of Engineering and IT
University of Technology, Sydney

Abstract

A central result in the AGM framework for belief revision is the construction of revision functions in terms of *total preorders* on possible worlds. These preorders encode comparative plausibility: $r \prec r'$ states that the world r is at least as *plausible* as r'. *Indifference* in the plausibility of two worlds, r, r', denoted $r \sim r'$, is defined as $r \not\prec r'$ and $r' \not\prec r$. Herein we take a closer look at plausibility indifference. We contend that the transitivity of indifference assumed in the AGM framework is not always a desirable property for comparative plausibility. Our argument originates from similar concerns in *preference modelling*, where a structure weaker than a total preorder, called a *semiorder*, is widely consider to be a more adequate model of preference. In this paper we essentially re-construct revision functions using semiorders instead of total preorders. We formulate postulates to characterise this new, wider, class of revision functions, and prove that the postulates are sound and complete with respect to the semiorder-based construction. The corresponding class of contraction functions (via the Levi and Harper Identities) is also characterised axiomatically.

Introduction

Plausibility rankings lie at the heart of modelling *belief change*. They take different forms depending on the approach employed and the type of belief change encoded – see (Peppas and Williams 1995) – ranging from *preorders* on possible worlds (Katsuno and Mendelzon 1991), to *epistemic entrenchments* (Gardenfors and Makinson 1988), to *ordering on remainders* (Alchourron, Gardenfors, and Makinson 1985). The essence however in all such forms is the same: plausibility ranking are used to determine the most plausible objects (worlds, sentences, or remainders respectively), among the alternatives in view of the epistemic input and the given type of belief change. That is, whenever during belief change, one is forced to choose between α or β, the most plausible of the two will be selected (as determined by the corresponding plausibility ranking).

An underlying assumption employed in the prevalent *AGM framework*[1] for belief change, is that *indifference of comparative plausibility is transitive*. To better illustrate our point, let us focus on plausibility rankings $\leqslant$ on possible worlds, called *faithful preorders* in (Katsuno and Mendelzon 1991).

Suppose that two distinct possible worlds w_1, w_2 are equally plausible (or implausible) relative to the agents' current belief state K. We shall denote this by $w_1 \sim w_2$; formally, $w_1 \sim w_2$ iff $w_1 \not\prec w_2$ and $w_2 \not\prec w_1$, where $\prec$ denotes the strict part of $\leqslant$. Suppose now that w_2 is equally plausible to a third world w_3; i.e. $w_2 \sim w_3$. In the AGM framework we automatically conclude that $w_1 \sim w_3$.

Economists on the other hand are more cautious. It has long been acknowledged in the area of *preference modelling* that transitivity is not always a natural property for indifference of preference. The following quote from (Luce 1956) illustrates the problem:

> *"Find a subject who prefers a cup of coffee with one cube of sugar to one with five cubes (this should not be difficult). Now prepare 401 cups of coffee with* $(1 + i/100) \cdot x$ *grams of sugar,* $i = 0, 1, \cdots, 400$*, where x is the weight of one cube of sugar. It is evident that he will be indifferent between cup i and cup* $i + 1$*, for any i, but by choice he is not indifferent between* $i = 0$ *and* $i = 400$*."*

The above example, along with further arguments made in (Luce 1956), support the view that rational agents tend to discriminate between two alternatives α, β only when their difference[2] exceeds a certain *threshold*. As noted by Armstrong in (Armstrong 1950),

> *"The nontransitiveness of indifference must be recognized and explained on [sic] any theory of choice, and the only explanation that seems to work is based on the imperfect powers of discrimination of the human mind whereby inequalities become recognizable only when of sufficient magnitude."*

One can imagine similar scenarios in the context of belief change. To use (an adequately adjusted variation of) a

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] The AGM framework is the formal framework introduced by *Alchourron, Gardenfors, and Makinson* in (Alchourron, Gardenfors, and Makinson 1985), and further developed by many others (see (Peppas 2008)) for the study of belief change. We shall review its main constituents later in the paper.

[2] Or more accurately, the difference of their respective utilities.

well known example, called *the bald man paradox* – see (Ludwig and Ray 2002) – suppose that our agent, Myrto, believes that her grandfather Speros, whom she never met, had a full head of hair. It is therefore reasonable to assume that the possible world w_{5000} in which her grandfather has 5000 hairs is more plausible to Myrto than the world w_{50} in which Speros has only 50 hairs. On the other hand, it is also reasonable to assume that Myrto is indifferent between worlds, like w_{4657} and w_{4656}, which differ only in that Speros has a single hair less in the latter. If indifference was transitive (as it is the case in the AGM framework), with 4050 applications of transitivity we would derive that Myrto is indifferent between w_{5000} and w_{50}, which of course it is not true.

Indifference between alternatives with fine variations may also be conscious, and not due to 'imperfect powers of discrimination". Ackerman illustrates this point with the following example, (Ackerman 1994):

> *"It is entirely plausible to suppose that an instructor would be indifferent to having the number of students in his seminar be 6 vs. 7, 7 vs. 8, etc., without being indifferent to having it be 6 vs. 15; he might consider 15 students too many for a seminar. But he can certainly discriminate between 6 and 7 students or 7 and 8, etc."*

Another reason for the intransitivity of preference indifference (and likewise, for the intransitivity of plausibility indifference) is the *multi-dimensionality of alternatives*. To take a two-dimensional example from (Fishburn 1970a):[3]

> *"You are going to buy a car. You have no definite preference between (Ford, at $2600) and (Chevrolet, at $2700), and also have no definite preference between (Ford, at $2600) and (Chevrolet, at $2705). However, (Chevrolet, at $2705) < (Chevrolet, at $2700)."*

Considerations like these led Luce to develop in (Luce 1956) a new structure, called a *semiorder*, which has since become widely accepted in the community of mathematical psychology as an adequate model for human preference (see for example, (Fishburn 1970b), (Jamison and Lau 1973), (Pirlot and Vincke 1997), (Rabinovitch 1977), (Rubinstein 2012)).

Herein, and for similar reasons, semiorders are imported in Belief Change; i.e. we re-construct the basic models and results of the AGM framework using semiorders (instead of total preorder) to model comparative plausibility. In particular, we formulate postulates that characterize the (wider) class of revision functions induced from semiorders, and we prove their correctness (i.e. we show that the postulates are *sound* and *complete* with respect to the semiorders-based construction). Moreover, we characterize axiomatically the class of *contraction functions* corresponding, via the Levi and Harper Identities, to semiorder-based revision functions.

The paper is structured as follows. In the next three sections we introduce some notation, we briefly review the AGM framework, and we recall the main definitions and results on semiorders. Following that there are two sections containing the main results of the paper. Finally, in the last two sections we discuss related works and make some concluding remarks.

[3]Notice that in this example "better" items appear higher in the $<$-ordering. This is essentially the reverse of the case with faithful preorders where the more plausible a world is, the lower it appears in the preorder.

Formal Preliminaries

Throughout this paper we work with a finite set of propositional variables P. We define L to be the propositional language generated from P (using the standard boolean connectives $\wedge, \vee, \rightarrow, \leftrightarrow, \neg$ and the special symbols $\top, \bot$) and governed by classical propositional logic.

For a set of sentences Γ of L, we denote by $Cn(\Gamma)$ the set of all logical consequences of Γ, i.e., $Cn(\Gamma) = \{x \in L: \Gamma \models x\}$. We shall often write $Cn(x_1, \ldots, x_n)$, for sentences $x_1, \ldots, x_n$, as an abbreviation of $Cn(\{x_1, \ldots, x_n\})$. For any two sentences x, y we shall write $x \equiv y$ iff $Cn(x) = Cn(y)$.

A theory K of L is any set of sentences of L closed under $\models$, i.e., $K = Cn(K)$. We shall denote the set of all theories of L by $\mathcal{T}$. We define a *possible world* r (or simply a *world*), to be a consistent set of literals such that for any propositional variables $x \in P$, either $x \in r$ or $\neg x \in r$. We will often identify a world r with the conjunction of its literals, leaving it to the context to resolve any ambiguity (for example in "$\neg r$", r is a sentence, whereas in "$r \cap \{x\}$", r is a set of literals). We denote the set of all possible worlds by $\mathcal{M}$.

For a set of sentences Γ of L, $[\Gamma]$ denotes the set of all possible worlds that entail Γ; i.e. $[\Gamma] = \{r \in \mathcal{M} : r \models \Gamma\}$. Often we use the notation $[x]$ for a sentence $x \in L$, as an abbreviation of $[\{x\}]$. For a theory K and a set of sentences Γ of L, we denote by $K + \Gamma$ the closure under $\models$ of $K \cup \Gamma$, i.e., $K + \Gamma = Cn(K \cup \Gamma)$. For a sentence $x \in L$ we often write $K + x$ as an abbreviation of $K + \{x\}$.

Finally, some definitions on binary relations. Let V be a nonempty set and R a binary relation in V. For any subset S of V, by $min(S, R)$ we denote the set $min(S, R) = \{w \in S$: for all $w' \in S$, $w'Rw$ entails $wRw'\}$. The elements in $min(S, R)$ are called *minimal in* S with respect to R (or simply *minimal in* S, when R is understood from the context). Observe that if R is irreflexive and anti-symmetric, the above definition of *min* is equivalent to: $min(S, R) = \{w \in S$: there is no $w' \in S$, such that $w'Rw\}$.

We shall say that a binary R relation in V is a preorder iff R is *reflexive* and *transitive*. Moreover, R is said to be *total* iff for all $w, w' \in V$, wRw' or $w'Rw$.

The AGM Framework

Much research in belief change is based on the work of Alchourron, Gardenfors and Makinson (Alchourron, Gardenfors, and Makinson 1985), who have developed a research framework for this process, known as the *AGM framework*. In this section we shall briefly review the *rationality postulates* for the two most important types of belief change, namely *belief revision* and *belief contraction*, as well a constructive model for belief revision based on total preorders on possible worlds.

The AGM Postulates

In the AGM framework, belief revision is modeled as a function $*$ mapping a theory K and a sentence x to the theory $K*x$. Alchourron, Gardenfors, and Makinson have proposed the following set of postulates for belief revision:

(K*1) $K * x$ is a theory.

(K*2) $x \in K * x$.

(K*3) $K * x \subseteq K + x$.

(K*4) If $\neg x \notin K$ then $K + x \subseteq K * x$.

(K*5) If $x \not\models \bot$ then $K * x \not\models \bot$.

(K*6) If $x \equiv y$ then $K * x = K * y$.

(K*7) $K * (x \wedge y) \subseteq (K * x) + y$.

(K*8) If $\neg y \notin K * x$ then $(K * x) + y \subseteq K * (x \wedge y)$.

A revision function $*$ models the process by which a rational agents modifies her belief set K to incorporate some new information x. A *contraction function* $\div$ on the other hand models the process by which a rational agent uproots a belief x from K.

Formally, a contraction function is defined as a function $\div$ mapping a theory K and a sentence x to the theory $K \div x$, that satisfies the following postulates:

(K$\div$1) $K \div x$ is a theory.

(K$\div$2) $K \div x \subseteq K$.

(K$\div$3) If $x \notin K$ then $K \div x = K$.

(K$\div$4) If $\not\models x$ then $x \notin K \div x$.

(K$\div$5) If $x \in K$, then $K \subseteq (K \div x) + x$.

(K$\div$6) If $x \equiv y$ then $K \div x = K \div y$.

(K$\div$7) $(K \div x) \cap (K \div y) \subseteq K \div (x \wedge y)$.

(K$\div$8) If $x \notin K \div (x \wedge y)$ then $K \div (x \wedge y) \subseteq K \div x$.

Revision and contraction functions are connected through the following well known identities:

$$K \div x = (K * \neg x) \cap K \quad \text{(Harper Identity)}$$

$$K * x = (K \div \neg x) + x \quad \text{(Levi Identity)}$$

It has been shown that for every revision function $*$ satisfying (K*1) - (K*8), the function $\div$induced from $*$ via the Harper Identity, satisfies (K$\div$1) - (K$\div$8). And conversely, for every contraction function $\div$ satisfying (K$\div$1) - (K$\div$8), the function $*$ induced from $\div$ via the Levi Identity, satisfies (K*1) - (K*8) – see (Gardenfors 1988), (Peppas 2008), (Alchourron and Makinson 1982) for details.

Faithful Preorders

Apart from axiomatic approaches to belief change, a number of explicit constructions have also been proposed. One popular such construction is the one proposed by Grove, (Grove 1988), based on a structure called a *system of spheres*. Later Katsuno and Mendelzon, (Katsuno and Mendelzon 1991), re-casted Grove's model for the special case of finitary propositional logic, introducing the notion of *faithful preorders* on possible worlds.

Figure 1: Semiorder with Intransitive Indifference

For a theory K, a preorder on possible worlds $\preccurlyeq$ is said to be *faithful* to K iff it is total and such that the minimal worlds (wrt $\preccurlyeq$) are those satisfying K; i.e. $min(\mathcal{M}, \preccurlyeq) = [K]$.[4] Given a preorder $\preccurlyeq$ that is faithful to K, the revision of K by any sentence x can be defined as follows:

(RP) $[K * x] = min([x], \preccurlyeq)$.

Intuitively $\preccurlyeq$ represents a plausibility ranking on possible worlds: the more plausible a world r is, the lower it appears in the ranking. Hence (RP) essentially defines $K * x$ are the theory induced by the most plausible worlds satisfying the new information x.

Katsuno and Mendelzon proved that the family of functions induced from faithful preorders via (RP), are precisely those satisfying the AGM postulates for revision.

Semiorders

As mentioned in the introduction, semiorders were proposed in (Luce 1956) as a more natural alternative to total preorders for modelling preference.

Given a finite set of choices V, a semiorder $\prec$ in V is defined as a binary relation in V that satisfies the following axioms, for any $r_1, r_2, r_3, w \in V$:

(SO1) $r_1 \not\prec r_1$.

(SO2) If $r_1 \prec r_2 \prec r_3$ then $r_1 \prec w$ or $w \prec r_3$.

(SO3) If $r_1 \prec r_2$ and $r_3 \prec r_4$ then $r_1 \prec r_4$ or $r_3 \prec r_2$.

For any two choices $r_1, r_2 \in V$, we shall say that we are *indifferent* between r_1 and r_2, denoted $r_1 \sim r_2$, iff $r_1 \not\prec r_2$ and $r_2 \not\prec r_1$. It is not hard to verify that with semiorders, indifference is not in general transitive. Consider for example the semiorder $\prec$ depicted in Figure 1, where the arrows between alternatives indicate preference. It is easy to verify that $\prec$ satisfies (SO1) - (SO3). Moreover observe that $r_2 \sim r_3$, $r_3 \sim r_4$, and yet $r_2 \prec r_4$.

A central result on semiorders that sheds light to their inner workings, relates to their numerical representation. It was been shown, (Scott and Suppes 1958), (Rabinovitch 1977), that every semiorder $\prec$ in V can be mapped to a *utility function* $u : V \mapsto \mathbb{R}$ such that for all $r_1, r_2 \in V$, $r_1 \prec r_2$ iff $u(r_2) - u(r_1) \geqslant 1$, and $r_1 \sim r_2$ iff $|u(r_2) - u(r_1)| < 1$. Intuitively this result says that the agent differentiates between two alternatives r_1 and r_2 iff the difference in their

[4] To be precise, in (Katsuno and Mendelzon 1991) a belief state is represented as a sentence rather than as a theory, and accordingly, faithful preorders are associated to sentences rather than theories. However, given that we are working with a language built over finitely many propositional variables, the two approaches are equivalent.

corresponding utilities exceeds a certain threshold (set to 1 in this case). For example, the semiorder $\prec$ of Figure 1 can be represented by the following utility function u:

$$u(r_1) = 0$$
$$u(r_2) = 1$$
$$u(r_3) = 1.5$$
$$u(r_4) = 2$$

Semiorders in Belief Revision

As mentioned in the introduction, the main aim of this article is to re-build (the main elements of) the AGM framework using semiorders rather than total preorders as models of comparative plausibility. In this section we start with revision functions.

Let $\prec$ be a semiorder in the set $\mathcal{M}$ of possible worlds. The function $* : \mathcal{T} \times L \mapsto \mathcal{T}$ induced from $\prec$ is defined by condition (AS) below:[5]

(AS) $\quad [K * x] = min([x], \prec)$.

We are interested in the class of functions induced from semiorders via (AS). Consider in particular the following set of postulates:

(A1) $\quad K * x$ is a theory.

(A2) $\quad x \in K * x$.

(A3) $\quad K * x \subseteq K + x$.

(A4) $\quad$ If $\neg x \notin K$ then $K + x \subseteq K * x$.

(A5) $\quad$ If $x \not\models \bot$ then $K * x \not\models \bot$.

(A6) $\quad$ If $x \equiv y$ then $K * x = K * y$.

(A7) $\quad K * (x \wedge y) \subseteq (K * x) + y$.

(A8) $\quad K * (x \vee y) \subseteq (K * x) + (K * y)$.

(A9) $\quad$ If $K * y \nsubseteq (K * x) + y$ then $K * x \subseteq (K * y) + x$.

(A10) $\quad$ If $\neg y \in K * x$ and $\neg z \notin K * x$ then $K * (x \wedge y) \subseteq (K * z) + (x \wedge y)$.

Postulates (A1) - (A7) are identical to the AGM postulates (K*1) - (K*7) for revision. Postulate (K*8) however cannot simply be copied across.

To see this, assume that the semiorder $\prec$ in Figure 1 represents the comparative plausibility of the worlds r_1, r_2, r_3, r_4. Moreover assume that the agent's current belief set is $K = Cn(r_1)$. In (Gardenfors 1988) it was shown in that (K*8) is equivalent to the following condition:

(3.15) $\quad$ If $\neg y \notin K * (x \vee y)$ then $K * (x \vee y) \subseteq K * y$

Set $x = r_2 \vee r_3 \vee r_4$ and $y = r_3 \vee r_4$. From Figure 1 and (AS) it follows that $[K * x] = \{r_2, r_3\}$, $[K * y] = \{r_3, r_4\}$, and $[K * (x \vee y)] = \{r_2, r_3\}$. Hence $\neg y \notin K * (x \vee y)$ and therefore by (3.15), $[K * y] \subseteq [K * (x \vee y)]$. This of course leads us to a contradiction since $r_4 \in [K * y]$ and $r_4 \notin [K * (x \vee y)]$.

Postulate (K*8) is therefore too strong for semiorder-based revision functions. It turns out, as shown by Theorems 1, 2 below, that (A8) - (A10) are precisely the weakening of (K*8) required to characterise the functions induced from semiorders via (AS).

[5]Condition (AS) is derived from (RP) with semiorders $\prec$ replacing preorders $\preccurlyeq$.

We note that postulate (A8) is not new. It has been used, in a slightly different format, by Benferhat et. al. in their axiomatization of revision functions induced from *partial preorders* (see postulate (P7) in (Benferhat, Lagrue, and Papini 2005)).

Before we present our representation results we need to adjust the definition of faithfulness to semiorders.

Let $\prec$ be a semiorder in $\mathcal{M}$ and K a theory of L. We shall say that $\prec$ is *faithful to* K iff the following conditions are satisfied:

(i) $\quad$ If $r \in [K]$ then there is no $r' \in \mathcal{M}$ such that $r' \prec r$.

(ii) $\quad$ If $r \in [K]$ and $r' \in (\mathcal{M} - [K])$ then $r \prec r'$.

Theorem 1 *Let K be a theory and $*$ a revision function satisfying (A1) - (A10). Then there exists a semiorder $\prec$ faithful to K, that satisfies (AS).*

Proof. Define $\prec$ to be the following binary relation in $\mathcal{M}$: $r \prec r'$ iff $r' \notin [K * (r \vee r')]$. Firstly we show that $\prec$ satisfies (SO1). Consider any possible world $r \in \mathcal{M}$. By (A6), $[K * (r \vee r)] = [K * r]$. Moreover, by (A2), (A5), it follows that $[K * r] = \{r\}$, and therefore $[K * (r \vee r)] = \{r\}$. Hence $r \nprec r$ as desired.

Next we show that (AS) is satisfied by showing $[K * x] \subseteq min([x], \prec)$ and $min([x], \prec) \subseteq [K * x]$.

Starting with the former, let r be any world in $[K * x]$. By (A2), $r \in [x]$. Consider now any world $r' \in [x]$. By (A7), $[K * x] \cap [r \vee r'] \subseteq [K * (x \wedge (r \vee r'))]$. Moreover notice that from $r, r' \in [x]$, it follows that $x \wedge (r \vee r') \equiv r \vee r'$. Hence by (A6), $[K * x] \cap [r \vee r'] \subseteq [K * (r \vee r')]$ and therefore $r \in [K * (r \vee r')]$. Hence $r' \nprec r$. Since r' was chosen as an arbitrary element of $[x]$ it then follows that $r \in min([x], \prec)$. Hence $[K * x] \subseteq min([x], \prec)$.

We prove the converse by induction on the size of $[x]$. If $|[x]| = 1$, then $[x] = \{r\}$. In this case, from (A2) and (A5), and (SO1) (established above) we derive that $[K * x] = \{r\} = min([x], \prec)$.

Assume that $\min([x], \prec) \subseteq [K * x]$, for any $x \in L$ such that $|[x]| \leqslant k$ (Induction Hypothesis).

For the inductive step, assume $|[x]| = k + 1$, and let r be any world in $min([x], \prec)$. We will show that $r \in [K * x]$. To this end, consider any world $r' \in [x]$ that is different from r, and define y to be the sentence $y = x \wedge (\neg r')$. Clearly $[y] = [x] - \{r'\}$ and therefore $r \in min([y], \prec)$. Moreover, $|[y]| = k$, and therefore by the induction hypothesis, $r \in [K * y]$. Also notice that since $r \in min([x], \prec)$ and $r' \in [x]$, it follows that $r' \nprec r$ and therefore, by the construction of $\prec$, $r \in [K * (r \vee r')]$. Hence by (A8), $r \in [K * (y \vee r \vee r')]$, and consequently by (A6), $r \in [K * x]$ as desired. Therefore (AS) holds.

Faithfulness of $\prec$ to K is straightforward. In particular, assume $r \in [K]$ and let r' be an arbitrary possible world. By (A3) - (A4), $[K * (r \vee r')] = [K] \cap [r \vee r']$ and therefore, $r \in [K * (r \vee r')]$. Hence $r' \nprec r$. Notice moreover that if $r' \notin [K]$, then $[K * (r \vee r')] = [K] \cap [r \vee r'] = \{r\}$, and consequently $r \prec r'$. Hence $\prec$ is faithful to K.

To conclude the proof of Theorem 1 we need to show that $\prec$ also satisfies (SO2) and (SO3). To this end, we will first show that $\prec$ is transitive.

Consider any three worlds $r_1, r_2, r_3 \in \mathcal{M}$ such that $r_1 \prec r_2 \prec r_3$. By (AS) it follows that $[K * (r_1 \vee r_2 \vee r_3)] = \{r_1\}$.

Clearly then, $[K * (r_1 \vee r_2 \vee r_3)] \cap [r_2 \vee r_3] = \emptyset$ and $[K * (r_1 \vee r_2 \vee r_3)] \cap [r_1 \vee r_3] \neq \emptyset$, which by (A6) and (A10) entails that $[K * (r_1 \vee r_3)] \cap [r_2 \vee r_3] \subseteq [K * (r_2 \vee r_3)]$. Hence, since $r_2 \prec r_3$ we derive that $r_3 \notin [K * (r_1 \vee r_3)]$, and therefore $r_1 \prec r_3$. Consequently $\prec$ is transitive.

Next for (SO2), let $r_1, r_2, r_3 \in \mathcal{M}$ be any three worlds such that $r_1 \prec r_2 \prec r_3$. Assume towards contradiction that there is a $w \in \mathcal{M}$ such that $r_1 \not\prec w$ and $w \not\prec r_3$. Notice that if $r_2 \prec w$ or $r_3 \prec w$, then transitivity gives us $r_1 \prec w$ which of course contradicts our assumption. Hence $r_2 \not\prec w$ and $r_3 \not\prec w$ (on top of $r_1 \not\prec w$), which combined with (SO1) and (AS) gives us, $w \in [K * (r_1 \vee r_2 \vee r_3 \vee w)]$ and $r_2, r_3 \notin [K * (r_1 \vee r_2 \vee r_3 \vee w)]$. Hence, $[K * (r_1 \vee r_2 \vee r_3 \vee w)] \cap [r_2 \vee r_3] = \emptyset$ and $[K * (r_1 \vee r_2 \vee r_3 \vee w)] \cap [r_3 \vee w] \neq \emptyset$, which by (A6) and (A10) entails $[K * (r_3 \vee w)] \cap [r_2 \vee r_3] \subseteq [K * (r_2 \vee r_3)]$. Consequently, since $r_3 \notin [K * (r_2 \vee r_3)]$ it follows that $r_3 \notin [K * (r_3 \vee w)]$, and therefore $w \prec r_3$, which of course contradicts our initial assumption.

Finally for (SO3), assume towards contradiction that for some $r_1, r_2, r_3, r_4 \in \mathcal{M}$, $r_1 \prec r_2$, $r_3 \prec r_4$, $r_1 \not\prec r_4$, and $r_3 \not\prec r_2$. From transitivity we then derive that $r_2 \not\prec r_4$, and $r_4 \not\prec r_2$. Consequently, by (SO1) and (AS), $r_4 \in [K * (r_1 \vee r_2 \vee r_4)]$ and $r_4 \notin [K * (r_2 \vee r_3 \vee r_4)]$. (A9) then entails that $[K * (r_2 \vee r_3 \vee r_4)] \cap [r_1 \vee r_2 \vee r_4] \subseteq [K * (r_1 \vee r_2 \vee r_4)]$. On the other hand, from (SO1) and (AS) we derive that $r_2 \in [K * (r_2 \vee r_3 \vee r_4)]$, and therefore $r_2 \in [K * (r_1 \vee r_2 \vee r_4)]$. This of course contradicts $r_1 \prec r_2$. ■

In proving the converse of Theorem 1, the following result will be useful:

Lemma 1 *Let $\prec$ be a semiorder in $\mathcal{M}$ and $S \subseteq \mathcal{M}$ a nonempty set of worlds. Then,*

- *(a)* *$\prec$ is transitive.*
- *(b)* *$min(S, \prec) \neq \emptyset$.*
- *(c)* *if $min(S, \prec) \neq \emptyset$, $r \in S$, and $r \notin min(S, \prec)$, then there exists a $r' \in min(S, \prec)$ such that $r' \prec r$.*

Proof. The transitivity of a semiorder follows directly from (Luce 1956). Nevertheless we include the proof herein for completeness. Consider any three worlds $r_1, r_2, r_3 \in \mathcal{M}$ such that $r_1 \prec r_2$ and $r_2 \prec r_3$. From (SO3) it follows that $r_1 \prec r_3$ or $r_2 \prec r_2$. The latter is excluded by (SO1), and consequently $r_1 \prec r_3$ as desired.

Next we prove the third statement of Lemma 1 by induction on the size of S. If $|S| = 1$, then $S = \{r\}$ for some world r. Consequently by (SO1), $min(S, \prec) = \{r\}$ and hence (c) trivially holds.

Assume that (c) is true for all nonempty sets with cardinality no greater than k (Induction Hypothesis).

For the inductive step, consider a set $S \subseteq \mathcal{M}$ such that $|S| = k + 1$, and assume that $min(S, \prec) \neq \emptyset$, and for some $r \in S$, $r \notin min(S, \prec)$. Then there is a $r' \in S$ such that $r' \prec r$. From transitivity and (SO1) we then derive that $r' \neq r$ and $r \not\prec r'$. Consider now the set $S - \{r\}$. Clearly, $r' \in (S - \{r\})$ and $|S - \{r\}| = k$. If $r' \in min(S - \{r\}, \prec)$, then from $r \not\prec r'$ it follows that $r' \in min(S, \prec)$, and therefore $r' \prec r$ gives us (c). Assume therefore that $r' \notin min(S - \{r\}, \prec)$. Then by the induction hypothesis, there exists a $r'' \in min(S - \{r\}, \prec)$ such that $r'' \prec r'$. Then transitivity entails $r'' \prec r$. Moreover (SO1) together with transitivity entail $r \not\prec r''$. Hence $r'' \in min(S, \prec)$. Combined with $r'' \prec r$ we derive (c).

Finally we prove (b), once again using induction on the size of S. If $|S| = 1$, then $S = \{r\}$ for some world r. Then by (SO1), $min(S, \prec) = \{r\}$ and hence $min(S, \prec) \neq \emptyset$.

Assume that (b) is true for all nonempty sets $S \subseteq \mathcal{M}$ of size up to k (Induction Hypothesis).

Suppose now that $S \subseteq \mathcal{M}$ is a set with cardinality $|S| = k + 1$. Pick an arbitrary world $r \in S$. If $r \in min(S, \prec)$ then (b) is satisfied. Assume therefore that $r \notin min(S, \prec)$. Then there is a $r' \in S$ such that $r' \prec r$. From (SO1) and transitivity it follows that $r' \neq r$ and $r \not\prec r'$. Hence $r' \in (S - \{r\})$. If $r' \in min(S - \{r\}, \prec)$, from $r \not\prec r'$ we derive that $r' \in min(S - \{r\}, \prec)$ and therefore $min(S, \prec) \neq \emptyset$ as desired. Assume therefore that $r' \notin min(S - \{r\}, \prec)$. By the induction hypothesis we derive that $min(S - \{r\}, \prec) \neq \emptyset$. Hence from (c) it follows that there is a $r'' \in min(S - \{r\}, \prec)$ such that $r'' \prec r'$. Transitivity then entails that $r'' \prec r$, and consequently from (SO1) and transitivity it follows that $r \not\prec r''$. This again entails that $r'' \in min(S, \prec)$, and therefore, $min(S, \prec) \neq \emptyset$. ■

With the aid of Lemma 1 we can now prove the soundness of our postulates wrt the semiorder-based construction:

Theorem 2 *Let K be a theory and $\prec$ a semiorder faithful to K. The revision function $*$ induced from $\prec$ via (AS), satisfies (A1) - (A10).*

Proof. The validity of (A1), (A2), and (A6) is straightforward. For (A3), notice that if $\neg x \in K$ then $K + x = L$, and therefore $K * x \subseteq K + x$. Assume now that $\neg x \notin K$, or equivalently, $[K] \cap [x] \neq \emptyset$. Then by the faithfulness of $\prec$ to K it follows that $min([x], \prec) = [K] \cap [x]$. Hence by (AS), $K * x = K + x$. This proves (A3). Moreover, the second part of the argument also proves (A4).

For (A5), assume that $x \not\models \bot$ or equivalently that $[x] \neq \emptyset$. Then from Lemma 1.(b) it follows that $min([x], \prec) \neq \emptyset$. Hence $K * x$ is consistent as desired.

For (A7), let r be any world in $[(K * x) + y]$ and assume towards contradiction that $r \notin [K * (x \wedge y)]$. From $r \in [(K * x) + y]$ it follows that $[x \wedge y] \neq \emptyset$, and hence from $r \notin [K * (x \wedge y)]$ we derive that for some $r' \in [x] \cap [y]$, $r' \prec r$. Consequently, $r \notin min([x], \prec)$, which of course contradicts $r \in [(K * x) + y]$.

For (A8), assume that $r \in [K * x] \cap [K * y]$. Clearly then $[x] \neq \emptyset$ and $[y] \neq \emptyset$. Suppose contrary to (A8), that $r \notin [K * (x \vee y)]$. Then, since $r \in [x \vee y]$ we derive that there is a $r' \in [x \vee y]$ such that $r' \prec r$. Since $[x \vee y] = [x] \cup [y]$ it follows that $r \in [x]$ or $r \in [y]$. In the first case, from $r' \prec r$ we derive that $r \in min([x], \prec)$. In the second case we derive that $r \in min([y], \prec)$. Hence in both cases we have that $r \notin [K * x] \cap [K * y]$, contradicting our initial assumption.

For (A9), assume that for some $x, y \in L$, $[K * x] \cap [y] \not\subseteq [K * y]$. Then there is a $r \in [K * x] \cap [y]$ such that $r \notin [K * y]$. From $r \in [K * x] \cap [y]$ it follows that $[x] \neq \emptyset$ and $[y] \neq \emptyset$. Hence from $r \notin [K * y]$ we derive that there is a $r' \in [y]$ such that $r' \prec r$. Now assume toward contradiction that $[K * y] \cap [x] \not\subseteq [K * x]$, and let $w \in \mathcal{M}$ be such that $w \in [K * y] \cap [x]$ and $w \notin [K * x]$. From the latter we derive that there is a $w' \in [x]$ such that $w' \prec w$. Then (SO3) entails

that $r' \prec w$ or $w' \prec r$. In the first case, since $r' \in [y]$, it follows that $w \notin min([y], \prec)$, which of course contradicts $w \in [K * y]$. In the second case, from $w' \in [x]$ we derive that $r \notin min([x], \prec)$, which contradicts $r \in [K * x]$. Hence, in both case we derive a contradiction, and therefore (A9) is true.

Finally for (A10), let $x, y, z \in L$ be such that $[K*x] \cap [y] = \emptyset$, and $[K * x] \cap [z] \neq \emptyset$. Assume towards contradiction that there exists a $r \in [K*z] \cap [x] \cap [y]$ such that $r \notin [K*(x \wedge y)]$. Then, by Lemma 1.(c), there exists a $r' \in min([x \wedge y], \prec)$ such that $r' \prec r$. From $[K * x] \cap [y] = \emptyset$ we then derive that $r' \notin min([x], \prec)$. Hence from Lemma 1.(c) it follows that there is a $r'' \in min([x], \prec)$ such that $r'' \prec r'$. Moreover from $[K * x] \cap [z] \neq \emptyset$ we derive that there is a $w \in [z]$ such that $w \in [K * x]$. From $r'' \prec r' \prec r$ and (SO2) we derive that $r'' \prec w$ or $w \prec r$. In the first case, since $r'' \in [x]$, it follows that $w \notin min([x], \prec)$, contradicting $w \in [K * x]$. In the latter case, since $w \in [z]$, we derive that $r \notin min([z], \prec)$, contradicting $r \in [K * z]$. Hence in both cases we derive a contradiction. Thus (A10) hold. ■

Semiorders in Belief Contraction

In this section we identify the class of contraction functions that correspond, via the Harper and Levi Identities, to semiorder-induced revision functions.

In particular, consider the following postulates:

(B1) $K \div x$ is a theory.

(B2) $K \div x \subseteq K$.

(B3) If $x \notin K$ then $K \div x = K$.

(B4) If $\not\models x$ then $x \notin K \div x$.

(B5) If $x \in K$ then $K \subseteq (K \div x) + x$.

(B6) If $x \equiv y$ then $K \div x = K \div y$.

(B7) $(K \div x) \cap (K \div y) \subseteq K \div (x \wedge y)$.

(B8) $K \div (x \wedge y) \subseteq (K \div x) + (K \div y)$.

(B9) If $K \div y \notin (K \div x) + \neg y$ then $K \div x \subseteq (K \div y) + \neg x$.

(B10) If $x \vee y \in K \div x$, and $x \vee z \notin K \div x$ then $K \div (x \vee y) \subseteq K \div (x \vee z) + \neg(x \vee y)$.

Postulates (B1) - (B7) are identical to (K÷1) - (K÷7). On the other hand (B8) - (B10) are weaker variants of (K÷8) (assuming the presence of (B1) - (B7)).

The following results show that the class of contraction functions satisfying (B1) - (B10) corresponds precisely, via the Levi and Harper Identities, to the class of revision functions satisfying (A1) - (A10).

Theorem 3 *Let K be a theory, $*$ a revision function satisfying (A1) - (A10), and $\div$ the function induced from $*$ via the Harper Identity. Then $\div$ satisfies (B1) - (B10).*

Proof. From Theorems 3.4, 3.5 in (Gardenfors 1988), it follows that $\div$ satisfies (B1) - (B7).

For (B8), consider any $x, y \in L$. By the Harper Identity, $[K \div x] \cap [K \div y] = ([K * \neg x] \cup [K]) \cap ([K * \neg y] \cup [K]) = ([K*\neg x] \cap [K*\neg y]) \cup ([K*\neg x] \cap [K]) \cup ([K*\neg y] \cap [K]) \cup [K]$. Moreover by (B2), $[K] \subseteq [K * \neg x]$, $[K] \subseteq [K * \neg y]$, and by (A6) and (A8), $[K*\neg x] \cap [K*\neg y] \subseteq [K*\neg(x \wedge y)]$. Consequently, $[K \div x] \cap [K \div y] \subseteq [K * \neg(x \wedge y)] \cup [K]$. Hence, by the Harper Identity, $K \div (x \wedge y) \subseteq (K \div x) + (K \div y)$ as desired.

For (B9), consider any $x, y \in L$ such that $[K \div x] \cap [\neg y] \notin [K \div y]$. Then by the Harper Identity, $([K * \neg x] \cup [K]) \cap [\neg y] \notin [K*\neg y] \cup [K]$ and consequently, $([K*\neg x] \cap [\neg y]) \cup ([K] \cap [\neg y]) \notin [K * \neg y] \cup [K]$. Since by (A3) $[K] \cap [\neg y] \subseteq [K * \neg y]$, we then derive that $[K * \neg x] \cap [\neg y] \notin [K * \neg y]$. Then by (A9) it follows that $[K * \neg y] \cap [\neg x] \subseteq [K * \neg x]$. Hence $([K*\neg y] \cap [\neg x]) \cup [K] \subseteq [K*\neg x] \cup [K]$ and therefore $([K * \neg y] \cup [K]) \cap ([\neg x] \cup [K]) \subseteq [K * \neg x] \cup [K]$. By the Harper Identity we then derive $([K \div y] \cap ([\neg x] \cup [K]) \subseteq [K \div x]$, and therefore, $([K \div y] \cap [\neg x]) \cup ([K \div y] \cap [K]) \subseteq [K \div x]$. Hence $[K \div y] \cap [\neg x] \subseteq [K \div x]$ or equivalently, $K \div x \subseteq (K \div y) + \neg x$.

Finally for (B10), assume that for some $x, y, z \in L$, $[K \div x] \cap [\neg x] \subseteq [y]$, and $[K \div x] \cap [\neg x] \notin [z]$. Firstly assume that $x \notin K$. Then by (B3), $K \div x = K$. Hence from $[K \div x] \cap [\neg x] \notin [z]$ we derive that $(x \vee z) \notin K$, and consequently by (B3), $K \div (x \vee z) = K$. Then by (B2), $K \div (x \vee y) \subseteq K \div (x \vee z)$, and therefore, $K \div (x \vee y) \subseteq K \div (x \vee z) + \neg(x \vee y)$ as desired.

Assume therefore that $x \in K$. From $[K \div x] \cap [\neg x] \subseteq [y]$, and $[K \div x] \cap [\neg x] \notin [z]$ and the Harper Identity it follows that $([K * \neg x] \cup [K]) \cap [\neg x] \subseteq [y]$, and $([K * \neg x] \cup [K]) \cap [\neg x] \notin [z]$. Consequently, $([K * \neg x] \cap [\neg x]) \cup ([K] \cap [\neg x]) \subseteq [y]$, and $([K * \neg x] \cap [\neg x]) \cup ([K] \cap [\neg x]) \notin [z]$. Since $x \in K$ we then derive, $[K*\neg x] \cap [\neg x] \cap [\neg y] = \emptyset$, and $[K*\neg x] \cap [\neg x] \cap [\neg z] \neq \emptyset$, or equivalently, $[K*\neg x] \cap [\neg x \wedge \neg y] = \emptyset$, and $[K*\neg x] \cap [\neg x \wedge \neg z] \neq \emptyset$. Axiom (A10) then entails that $[K*(\neg x \wedge \neg z)] \cap [\neg x \wedge \neg y] \subseteq [K*(\neg x \wedge \neg y)]$. Hence $([K*(\neg x \wedge \neg z)] \cap [\neg x \wedge \neg y]) \cup [K] \subseteq [K*(\neg x \wedge \neg y)] \cup [K]$, and consequently, by (A6), $([K * \neg(x \vee z)] \cup [K]) \cap ([\neg x \wedge \neg y] \cup [K]) \subseteq [K*\neg(x \vee y)] \cup [K]$. Therefore by the Harper Identity, $[K \div (x \vee z)] \cap ([\neg(x \vee y)] \cup [K]) \subseteq [K \div (x \vee y)]$. Hence $([K \div (x \vee z)] \cap [\neg(x \vee y)]) \cup ([K \div (x \vee z)] \cap [K]) \subseteq [K \div (x \vee y)]$, and consequently, $K \div (x \vee y) \subseteq K \div (x \vee z) + \neg(x \vee y)$ as desired. ■

Theorem 4 *Let K be a theory, $*$ a contraction function satisfying (B1) - (B10), and $*$ the function induced from $*$ via the Levi Identity. Then $*$ satisfies (A1) - (A10).*

Proof. From Theorems 3.2, 3.3 in (Gardenfors 1988), it follows that $*$ satisfies (A1) - (A7).

For (A8), let x, y be any sentences in L. By the Levi Identity, $[K * x] \cap [K * y] = [K \div \neg x] \cap [x] \cap [K \div \neg y] \cap [y]$ and therefore by (B8) and (B6), $[K * x] \cap [K * y] \subseteq [K \div \neg(x \vee y)] \cap [x \wedge y]$. Moreover notice that $[x \wedge y] \subseteq [x \vee y]$ and consequently we derive $[K * x] \cap [K * y] \subseteq [K \div \neg(x \vee y)] \cap [x \vee y]$. Hence from the Levi Identity it follows that $K * (x \vee y) \subseteq (K * x) + (K * y)$ as desired.

For (A9), assume that for some $x, y \in L$, $[K * x] \cap [y] \notin [K * y]$. Then by the Levi Identity we have that $[K \div \neg x] \cap [x] \cap [y] \notin [K \div \neg y] \cap [y]$ and consequently $[K \div \neg x] \cap [y] \notin [K \div \neg y]$. Then by (B9), $[K \div \neg y] \cap [x] \subseteq [K \div \neg x]$, and therefore $[K \div \neg y] \cap [x] \subseteq [K \div \neg x] \cap [x]$. This again entails $[K \div \neg y] \cap [x] \cap [y] \subseteq [K \div \neg x] \cap [x]$ which by the Levi Identity entails $[K * y] \cap [x] \subseteq [K * x]$, or equivalently, $K * x \subseteq (K * y) + x$ as desired.

Finally for (A10), assume that for some $x, y, z \in L$, $[K * x] \cap [y] = \emptyset$ and $[K * x] \cap [z] \neq \emptyset$. Then by the Levi Identity we get, $[K \div \neg x] \cap [x] \cap [y] = \emptyset$ and $[K \div \neg x] \cap [x] \cap [z] \neq \emptyset$, or equivalently, $\neg y \in (K \div \neg x) + x$ and $\neg z \notin (K \div \neg x) + x$. Hence by (B10) and (B6), $[K \div \neg(x \wedge z)] \cap [x \wedge y] \subseteq [K \div \neg(x \wedge y)]$, and consequently, $[K \div \neg(x \wedge z)] \cap [x \wedge y] \subseteq [K \div \neg(x \wedge y)] \cap [x \wedge y]$. This again entails, $[K \div \neg(x \wedge z)] \cap [x \wedge z] \cap [x \wedge y] \subseteq [K \div \neg(x \wedge y)] \cap [x \wedge y]$, and therefore by the Levi Identity we derive, $K * (x \wedge y) \subseteq K * (x \wedge z) + (x \wedge y)$. Moreover by (A6) - (A7), $K * (x \wedge z) \subseteq (K * z) + x$ and therefore $K * (x \wedge y) \subseteq (K * z) + (x \wedge y)$ as desired. ■

Related Work

To our knowledge, this is the first time that semiorders are used in Belief Change. Yet a number of researchers have considered other weaker alternatives of total preorders as models of comparative plausibility. We have already mentioned (Benferhat, Lagrue, and Papini 2005), in which the authors study revision functions based on *partial preorders*.

Another instance is (Rott to appear). In this paper Rott, focuses on contraction functions, and uses the metaphor of a multilayer edifice to classify them. In particular, on the ground floor of the edifice one encounters the contraction functions satisfying the basic postulates $(K \div 1)$ - $(K \div 6)$. On the top floor dwell the fully fledged AGM contraction functions satisfying all eight postulates $(K \div 1)$ - $(K \div 8)$. In the space between the ground and the top floors, Rott identifies three more floors, each of which is of interest in its own right. The 3rd floor of the edifice is the one most relevant to the present work: on this floor live the relational partial meet contraction functions induced from *interval orders* (on remainders).[6] We recall that an interval order is a binary relation $<$ satisfying (SO1) and (SO3); hence every semiorder is an interval order, but not vice versa.

Rott proves that the functions residing on the 3rd floor can be characterised axiomatically by $(K \div 1)$ - $(K \div 7)$, together with the postulates $(K \div 8\text{c})$ and $(K \div 8\text{d})$ below; notice the close resemblance between $(K \div 8\text{d})$ and our postulate (B8):

$(K \div 8\text{c})$ If $y \in K \div (x \wedge y)$, then $K \div (x \wedge y) \subseteq K \div x$.

$(K \div 8\text{d})$ $K \div (x \wedge y) \subseteq (K \div x) \cup (K \div y)$.

In view of the prominence of semiorders in modeling intransitivity of preference indifference (and of plausibility indifference), we believe that an extra floor could be added to Rott's edifice between the 3rd and the top floor, to accommodate the contraction functions of the previous section.

Interval orders have also been used in (Booth and Meyer 2011). However, whereas herein we use semiorders to model comparative plausibility, in (Booth and Meyer 2011) interval orders are used at a meta-level to guide the *iterated revision* process (with comparative plausibility still been modelled with total preorders like in the classical AGM framework).

Conclusion

Researchers in the *preference modelling* community have long argued in favour of *semiorders* over *total preorders* as a more natural model for preference. The main advantage of semiorders is the intransitivity of indifference.

Herein we have argued that in the context of Belief Change indifference in plausibility could also be non-transitive. We therefore re-built the AGM framework with semiorders replacing total preorders. In particular we characterize axiomatically the class of revision functions induced by semiorders, along with the corresponding class or contraction functions (via the Levi and Harper Identities). Future work will include similar results for the epistemic entrenchment and the partial meet models.

Of course Belief Change is not the only area in Knowledge Representation that uses plausibility rankings. *Nonmonotonic Reasoning* for example also relies heavily on such rankings. Moreover, like in Belief Change, plausibility indifference in Nonmonotonic Reasoning is also typically assumed to be transitive. Hence we believe that the study of semiorders in the context of Nonmonotonic Reasoning is an important avenue for future work that could lead to significant contributions.

[6] See (Gardenfors 1988), (Peppas 2008) for details on the partial meet model for contraction.

Acknowledgements

We are grateful to Abhaya Nayak and Maurice Pagnucco for many valuable discussion on this work. Moreover we would like to thank the anonymous reviewers for their detailed comments and insightful input, as well as Hans Rott for bringing (Rott to appear) to our attention.

References

Ackerman, F. 1994. Roots and consequences of vagueness. *Philosophical Perspectives* 8(2).

Alchourron, C., and Makinson, D. 1982. On the logic of theory change: Contraction functions and their associated revision functions. *Theoria* 46(1):14–37.

Alchourron, C.; Gardenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic* 50:510–530.

Armstrong, W. 1950. A note on the theory of consumer's behavior. *Oxford Economic Papers* 2:119–12.

Benferhat, S.; Lagrue, S.; and Papini, O. 2005. Revision of partially ordered information: Axiomatization, semantics and iteration. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Booth, R., and Meyer, T. 2011. How to revise a total preorder. *Journal of Philosophical Logic* 40:193–238.

Fishburn, P. 1970a. Intransitive indifference in preference theory: A survey. *Operations Research* 18:207–228.

Fishburn, P. 1970b. Intransitive indifference with unequal indifference intervals. *Journal of Mathematical Psychology* 7:144–149.

Gardenfors, P., and Makinson, D. 1988. Revisions of knowledge systems using epistemic entrenchment. In *Proceedings of Theoretical Aspects of Reasoning about Knowledge*, 83–95. Morgan-Kaufmann.

Gardenfors, P. 1988. *Knowledge in Flux*. MIT press.

Grove, A. 1988. Two modellings for theory change. *Journal of Philosophical Logic* 17:157–170.

Jamison, D., and Lau, L. 1973. Semiorders and the theory of choice. *Econometrica* 41:901–912.

Katsuno, H., and Mendelzon, A. 1991. Propositional knowledge base revision and minimal change. *Artificial Intelligence* 52(3):263–294.

Luce, R. D. 1956. Semiorders and a theory of utility discrimination. *Econometrica* 24(2):178–191.

Ludwig, K., and Ray, G. 2002. Vagueness and the sorites paradox. *Philosophical Perspectives* 16.

Peppas, P., and Williams, M.-A. 1995. Constructive modelings for theory change. *Notre Dame Journal of Formal Logic* 36(1):120–133.

Peppas, P. 2008. Belief revision. In *Handbook of Knowledge Representation*. Elsevier Science. 317–359.

Pirlot, M., and Vincke, P. 1997. *Semiorders: Properties, representations, applications*. Kluwer Academic Publishers.

Rabinovitch, I. 1977. The scott-suppes theorem on semiorders. *Journal of Mathematical Psychology* 1.

Rott, H. (to appear). Three floors for the theory of theory change. *The Logica Yearbook 2013*.

Rubinstein, A. 2012. *Lecture notes in microeconomic theory: the economic agent*. Princton University Press.

Scott, D., and Suppes, P. 1958. Foundational aspects of theories of measurement. *Journal of Symbolic Logic* 23(2).

David Poole's Specificity Revised*

Claus-Peter Wirth and **Frieder Stolzenburg**
FB AI, Harz University of Applied Sciences, 38855 Wernigerode, Germany
wirth@logic.at, fstolzenburg@hs-harz.de

Abstract

In the middle of the 1980s, David Poole introduced a semantical, model-theoretic notion of specificity to the artificial-intelligence community. Since then it has found further applications in non-monotonic reasoning, in particular in defeasible reasoning. Poole's notion, however, turns out to be intricate and problematic, which — as we show — can be overcome to some extent by a closer approximation of the intuitive human concept of specificity. Besides the intuitive advantages of our novel specificity ordering over Poole's specificity relation in the classical examples of the literature, we also report some hard mathematical facts: Contrary to what was claimed before, we show that Poole's relation is not transitive. Our new notion of specificity is transitive and also monotonic w.r.t. conjunction.

Keywords: specificity, defeasible reasoning, argumentation.

1 Introduction

A possible explanation of how humans manage to interact with reality — in spite of the fact that their information on the world is partial and inconsistent — mainly consists of the following two points: Humans use a certain amount of rules for default reasoning and are aware that some arguments relying on these rules may be defeasible. In case of the frequent conflicting or even contradictory results of their reasoning, they prefer more specific arguments to less specific ones. An intuitive concept of specificity plays an essential rôle in this explanation, which seems to be highly successful in practice. On the long way approaching this proven intuitive human concept of specificity, the first milestone marks the development of a semantical, model-theoretic notion of specificity having passed first tests of its usefulness and empirical validity. Indeed, at least as the first step, a semantical, model-theoretic notion will probably offer a broader and better basis for applications in systems for common sense reasoning than notions of specificity that depend on peculiarities of special calculi or even on extra-logical procedures. This holds in particular if the results of these systems are to be accepted by humans.

*Supported by the DFG grant Sto 421/5-1.
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

David Poole (1985) has sketched such a notion as a binary relation on arguments and evaluated its intuitive validity with some examples. Poole's notion of specificity was given a more appropriate formalization in (Simari and Loui 1992). The properties of this formalization were examined in detail in (Stolzenburg et al. 2003).

In this paper, before we give a specification of the formal requirements on any reasonably conceivable relation of specificity in Sect. 4, we present a detailed analysis of the intentional motivation of our *intuition that Poole's specificity is a first step on the right way* (Sect. 3). Moreover, in Sect. 5, we clearly disambiguate Poole's specificity from slightly improved versions such as the one in (Simari and Loui 1992), and introduce a novel specificity relation ($\lesssim_{\mathrm{CP1}}$), which presents a major correction of Poole's specificity because it removes a crucial shortcoming of Poole's original relation ($\lesssim_{\mathrm{P1}}$) and its slight improvements ($\lesssim_{\mathrm{P2}}$, $\lesssim_{\mathrm{P3}}$), namely their lack of transitivity. Finally, in Sect. 6, we present several examples that will convince every carefully contemplating reader of the superiority of our novel specificity relation $\lesssim_{\mathrm{CP1}}$ w.r.t. human intuition, including monotonicity w.r.t. conjunction. We briefly discuss related works in Sect. 7, and conclude with Sect. 8.

2 Basic Notions and Notation

Let us narrow the general logical setting of specificity down to the concrete framework of *defeasible logic with the restrictions of logic programming*, as found e.g. in (Stolzenburg et al. 2003; Chesñevar et al. 2003).

A *literal* is an atom, possibly prefixed with the symbol "$\neg$" for negation. A *rule* is a literal, but possibly suffixed with a reverse implication symbol "$\Leftarrow$" that is followed by a conjunction of literals, consisting of one literal at least. Let Π be a set of rules. The *theory of* Π is the set $\mathfrak{T}_\Pi$ inductively defined to contain all instances of literals from Π and all literals L for which there is a conjunction C of literals from $\mathfrak{T}_\Pi$ such that $L \Leftarrow C$ is an instance of a rule in Π. For $\mathfrak{L} \subseteq \mathfrak{T}_\Pi$, we also say that Π *derives* $\mathfrak{L}$, and write $\Pi \vdash \mathfrak{L}$. Π is *contradictory* if there is an atom A such that $\Pi \vdash \{A, \neg A\}$; otherwise Π is *non-contradictory*.

Throughout this paper, we will assume a set of literals and two sets of rules to be given: A set Π^{F} of literals meant to describe the *Facts* of the concrete situation under consideration, a set Π^{G} of *General rules* meant to hold in all possible worlds, and a set Δ of *defeasible* (or default) rules meant to hold in most situations. The set $\Pi := \Pi^{\mathrm{F}} \cup \Pi^{\mathrm{G}}$ is the set of *strict* rules that — contrary to the defeasible rules — are considered to be safe and are not doubted in any concrete situation. There is no difference in derivations between the strict rules from Π and the defeasible rules from Δ. If a contradiction occurs, however, we will narrow the defeasible rules from Δ down to a subset $\mathcal{A}$ of its *ground* instances, i.e. instances without free variables, such that no further instantiation can occur. Such a subset, together with the literal whose derivation is in focus, is called an *argument*. With implicit reference to the fixed sets of rules Π and Δ, we formally define:

Definition 2.1 (Argument)
$(\mathcal{A}, L)$ is an *argument* if $\mathcal{A}$ is a set of ground instances of rules from Δ and $\mathcal{A} \cup \Pi \vdash \{L\}$.

For ease of distinction, we will use the special symbol "$\leftarrow$" as a syntactical sugar in concrete examples of defeasible rules from Δ, instead of the symbol "$\Leftarrow$", which — in our concrete examples — will be used only in strict rules.

Example 2.2 (Poole 1985, Example 1)
$\Pi^{\mathrm{F}}_{2.2} := \{\ \mathsf{bird(tweety)},\ \mathsf{emu(edna)}\ \},$
$\Pi^{\mathrm{G}}_{2.2} := \left\{ \begin{array}{l} \mathsf{bird}(x) \Leftarrow \mathsf{emu}(x), \\ \neg\mathsf{flies}(x) \Leftarrow \mathsf{emu}(x) \end{array} \right\},$
$\Delta_{2.2} := \{\ \mathsf{flies}(x) \leftarrow \mathsf{bird}(x)\ \},$
$\mathcal{A}_2 := \{\ \mathsf{flies(edna)} \leftarrow \mathsf{bird(edna)}\ \}.$
We have $\mathfrak{T}_{\Pi_{2.2}}$=$\{\mathsf{bird(tweety)}, \mathsf{emu(edna)}, \mathsf{bird(edna)}, \neg\mathsf{flies(edna)}\}$, $\mathfrak{T}_{\Pi_{2.2}\cup\Delta_{2.2}}$=$\{\mathsf{flies(edna)}, \mathsf{flies(tweety)}\} \cup \mathfrak{T}_{\Pi_{2.2}}$. It is intuitively clear here that we prefer the argument $(\emptyset, \neg\mathsf{flies(edna)})$ to the argument $(\mathcal{A}_2, \mathsf{flies(edna)})$, simply because the former is more specific. We will further discuss this in Example 5.17.

We will use several binary relations comparing arguments according to their specificity. For any relation written as $\lesssim_N$ ("being more or equivalently specific w.r.t. N"), we set
$\gtrsim_N := \{\ (X,Y) \mid Y \lesssim_N X\ \}$ ("less or equivalently specific"),
$\approx_N := \lesssim_N \cap \gtrsim_N$ ("equivalently specific"),
$<_N := \lesssim_N \setminus \gtrsim_N$ ("properly more specific"),
$\leq_N := <_N \cup \{\ (X,X) \mid X \text{ is an argument}\ \}$ ("more specific or equal"),
$\triangle_N := \left\{ (X,Y) \;\middle|\; \begin{array}{l} X, Y \text{ are arguments with} \\ X \not\lesssim_N Y \text{ and } X \not\gtrsim_N Y \end{array} \right\}$ ("incomparable w.r.t. specificity").
A *quasi-ordering* is a reflexive transitive relation. An *(irreflexive) ordering* is an irreflexive transitive relation. A *reflexive ordering* (also called: "partial ordering") is an antisymmetric quasi-ordering. An *equivalence* is a symmetric quasi-ordering.

Corollary 2.3
If $\lesssim_N$ is a quasi-ordering, then $\approx_N$ is an equivalence, $<_N$ is an ordering, and $\leq_N$ is a reflexive ordering.

3 An Intuitive Notion of Specificity

It is part of general knowledge that a criterion is [properly] more specific than another one if the *class of candidates that satisfy it* is a [proper] subclass of that of the other one. Analogously — taking logical formulas as the criteria — a formula A is [properly] more specific than a formula B, if the model class of A is a [proper] subclass of the model class of B, i.e. if $A \models B$ [and $B \not\models A$].

To enable a closer investigation of the critical parts of a defeasible derivation, we have to isolate its defeasible parts. Abstracting from the concrete derivation of a literal L, let us take the set $\mathcal{A}$ of the ground instances of the defeasible rules that are actually applied in the derivation, and form the pair $(\mathcal{A}, L)$, which we already called an *argument* in Definition 2.1.

If we want to classify a derivation with defeasible rules according to its specificity, then we have to isolate the defeasible part of the derivation and look at its input. In our setting, the input consists of the set of those literals on which the defeasible part of the derivation is based, called the *activation set* for the defeasible part of the derivation. In our framework of defeasible logic programming, the only relevant property of an activation set can be the conjunction of its literals which is immediately represented by the set itself.

Because all literals of an activation set have been derived from the given specification, it does not make sense to compare activation sets w.r.t. the models of the entire specification. Indeed, only a comparison w.r.t. the models of a sub-specification can show any differences between them.

It is clear that we want to have the *entire* set Π^{G} for our comparison of activation sets, simply because we want to base our specificity classification on our specification, namely on its general and strict part. We have to exclude Π^{F} from this comparison, however. This exclusion makes sense because the defeasible rules are typically default rules not written in particular for the given concrete situation that is formalized by Π^{F}, and because the inclusion of Π^{F} would trivially admit every activation set. Moreover, as we want to compare the defeasible parts of derivations, we should exclude the defeasible rules from Δ from this comparison. We conclude that Π^{G} is that part of our specification according to which activation sets are to be compared.

Very roughly speaking, if we have fewer activation sets, then we have fewer models, which again means to have a higher specificity. Accordingly, a first sketch of a notion of specificity can now be given as follows:

> An argument $(\mathcal{A}_1, L_1)$ is [properly] *more specific than* an argument $(\mathcal{A}_2, L_2)$ if, for each activation set H_1 for $(\mathcal{A}_1, L_1)$, there is an activation set $H_2 \subseteq \mathfrak{T}_{H_1 \cup \Pi^{\mathrm{G}}}$ for $(\mathcal{A}_2, L_2)$ [but not vice versa].

Note that this notion of specificity is preliminary, and that the notion of an activation set for argument has not been properly defined yet.

On the one hand, the argument $(\mathcal{A}, L)$ is a nice abstraction from the derivation of L, because it perfectly suits our model-theoretic intentions described in Sect. 1. By this abstraction, on the other hand, we lose the possibility to isolate the defeasible parts of the derivation more precisely.

Let us compare this set $\mathcal{A}$ with an *and-tree of the derivation*. Every node in such a tree is labeled with the conclusion of an instance of a rule, such that its children are labeled exactly with the elements of the conjunction in the condition of this instance.

An isolation of the defeasible parts of an and-tree of the derivation may proceed as follows:

- Starting from the root of the tree, we iteratively erase all applications of strict rules. This results in a set of trees, each of which has the application of a defeasible rule at the root.
- Starting now from the leaves of these trees, we again erase all applications of strict rules. This results in a set of trees where all nodes *all* of whose children are leaves result from an application of a defeasible rule.

In a first approximation, we may now take all leaves of all resulting trees as the activation set for the original derivation.

Note that in the set of trees resulting from the described procedure, there may well have remained instances of rules from Π^{G} connecting a defeasible root application with the defeasible applications right at the leaves. Thus — to cover the whole defeasible part of the derivation in our abstraction — we have to consider the set $\mathcal{A} \cup \Pi^{\mathrm{G}}$ instead of just the set $\mathcal{A}$.

More precisely, we have to include all proper rules (i.e. those with non-empty conditions) from Π^{G}, and may also include the literals in Π^{G} because they cannot do any harm. As a consequence, in the modeling via our abstraction $\mathcal{A}$, we cannot prevent the precisely isolated defeasible sub-trees resulting from the described procedure from using the rules from Π^{G} to grow toward the root and toward the leaves again. It is clear, however, that only the growth toward the leaves can affect our activation sets and our notion of specificity.

Let us have a closer look at the effects of such a growth toward the leaves in the most simple case. In addition to a given activation set $\{\mathsf{Q}(\mathsf{a})\}$, in the presence of a general rule

$$\mathsf{Q}(x) \Leftarrow \mathsf{P}_0(x) \wedge \cdots \wedge \mathsf{P}_{n-1}(x)$$

from Π^{G}, we will also have to consider the activation set $\{\, \mathsf{P}_i(\mathsf{a}) \mid i \in \{0, \ldots, n-1\} \,\}$. This has two effects.

The first effect is that we immediately realize that every model of Π^{G} that is represented by the activation set $\{\, \mathsf{P}_i(\mathsf{a}) \mid i \in \{0, \ldots, n-1\} \,\}$ is also represented by the activation set $\{\mathsf{Q}(\mathsf{a})\}$; simply because $\{\, \mathsf{P}_i(\mathsf{a}) \mid i \in \{0, \ldots, n-1\} \,\}$ is added to the activation sets by the growth toward the leaves, such that an argument based on $\{\mathsf{Q}(\mathsf{a})\}$ becomes obviously less (or equivalently) specific than any argument that gets along with $\{\, \mathsf{P}_i(\mathsf{a}) \mid i \in \{0, \ldots, n-1\} \,\}$.

3.1 Preference of the "More Concise"

The second effect is that an argument that gets along with $\{\mathsf{Q}(\mathsf{a})\}$ becomes even *properly* less specific than one that actually requires $\{\, \mathsf{P}_i(\mathsf{a}) \mid i \in \{0, \ldots, n-1\} \,\}$ and does not get along with $\{\mathsf{Q}(\mathsf{a})\}$, simply because the former argument has the additional activation set $\{\mathsf{Q}(\mathsf{a})\}$. This effect is called *preference of the "more concise",* cf. e.g. (Stolzenburg et al. 2003, p. 94), (García and Simari 2004, p. 108). The standard example for the preference of the "more concise" is the following minor variation of Example 2.2.

Example 3.1 (Poole 1985, Example 2)
$\Pi^{\mathrm{F}}_{3.1} := \Pi^{\mathrm{F}}_{2.2} \qquad \Pi^{\mathrm{G}}_{3.1} := \{\ \mathsf{bird}(x) \Leftarrow \mathsf{emu}(x)\ \},$
$\Delta_{3.1} := \left\{ \begin{array}{l} \neg\mathsf{flies}(x) \leftarrow \mathsf{emu}(x), \\ \mathsf{flies}(x) \leftarrow \mathsf{bird}(x) \end{array} \right\},$
$\mathcal{A}_1 := \{\ \neg\mathsf{flies}(\mathsf{edna}) \leftarrow \mathsf{emu}(\mathsf{edna})\ \},$
$\mathcal{A}_2 := \{\ \mathsf{flies}(\mathsf{edna}) \leftarrow \mathsf{bird}(\mathsf{edna})\ \}.$
The argument $(\mathcal{A}_2, \mathsf{flies}(\mathsf{edna}))$ gets along with the activation set $\{\mathsf{bird}(\mathsf{edna})\}$, and thus it is properly less specific than the argument $(\mathcal{A}_1, \neg\mathsf{flies}(\mathsf{edna}))$, which actually requires the stronger $\{\mathsf{emu}(\mathsf{edna})\}$.

The problem now is that the statement

$$\mathsf{Q}(\mathsf{a}) \not\models \mathsf{P}_0(\mathsf{a}) \wedge \cdots \wedge \mathsf{P}_{n-1}(\mathsf{a}),$$

which is required to justify the the appropriateness of this effect, is not explicitly given by the specification via $(\Pi^{\mathrm{F}}, \Pi^{\mathrm{G}}, \Delta)$.

Nevertheless — if we do not just want to see it as a matter-of-fact property of notions of specificity in the style of Poole — the preference of the "more concise" can be justified by the habits of human specifiers as follows: If human specifiers write an implication in form of a rule $\mathsf{Q}(x) \Leftarrow \mathsf{P}_0(x) \wedge \cdots \wedge \mathsf{P}_{n-1}(x)$ into a specification Π of strict (i.e. non-defeasible) knowledge, then they typically intend that the implication is proper in the sense that its converse does not hold in general; otherwise they would have used an equivalence or equality symbol instead of the implication symbol, or replaced each occurrence of each $\mathsf{Q}(t)$ with $\mathsf{P}_0(t) \wedge \cdots \wedge \mathsf{P}_{n-1}(t)$, respectively. In particular, in our setting of logic programming — where disjunctive properties of the definition of a predicate are spread over several rules — the implications definitely tend to be proper. Therefore, if seasoned specifiers write down such a rule, then they do not want to exclude models where Q holds for some object a, but not all of the P_i do. This means that if we find such a rule in the strict and general part Π^{G} of a specification, then it is reasonable to assume that the implication is proper w.r.t. the intuition captured in the defeasible rules in Δ.

Thus, it makes sense to consider a defeasible argument based on $\{\, \mathsf{P}_i(\mathsf{a}) \mid i \in \{0, \ldots, n-1\} \,\}$ to be properly more specific than an argument that can get along with $\mathsf{Q}(\mathsf{a})$.

Finally, let us remark that our justification for the preference of the "more concise" does not apply if

$$\mathsf{Q}(x) \Leftarrow \mathsf{P}_0(x) \wedge \cdots \wedge \mathsf{P}_{n-1}(x)$$

is a *defeasible* rule instead of a strict one, because we then have the following three problems:

- the inclusion given by the rule is not generally intended (otherwise the rule should be a strict one),
- we cannot easily describe the actual instances to which the default rule is meant to apply (otherwise this more concrete description of the defeasible rule should be stated as a strict rule), and
- the direct treatment of a defeasible equivalence neither has to be appropriate as a default rule in the given situation, nor do we have any means to express a defeasible equivalence in the current setting.

Example 3.2 (Poole 1985, Example 3, renamed)
$\Pi^{\mathrm{F}}_{3.2} := \{\ \mathsf{emu(edna)}\ \}, \qquad \Pi^{\mathrm{G}}_{3.2} := \emptyset,$
$\Delta_{3.2} := \left\{ \begin{array}{l} \neg\mathsf{flies}(x) \leftarrow \mathsf{emu}(x), \\ \mathsf{flies}(x) \leftarrow \mathsf{bird}(x), \\ \mathsf{bird}(x) \leftarrow \mathsf{emu}(x) \end{array} \right\},$
$\mathcal{A}_1 := \{\ \neg\mathsf{flies(edna)} \leftarrow \mathsf{emu(edna)}\ \},$
$\mathcal{A}_2 := \left\{ \begin{array}{l} \mathsf{flies(edna)} \leftarrow \mathsf{bird(edna)}, \\ \mathsf{bird(edna)} \leftarrow \mathsf{emu(edna)} \end{array} \right\}.$
According to the above discussion, there is no clear reason why we should consider the argument $(\mathcal{A}_2, \mathsf{flies(edna)})$ to be *properly* less specific than the argument $(\mathcal{A}_1, \neg\mathsf{flies(edna)})$.

3.2 Preference of the "More Precise"

By an analogous argumentation, we can say that an argument that essentially requires an activation set $\{\ \mathsf{P}_i(\mathsf{a}) \mid i \in \{0,\ldots,n\}\ \}$ is *properly* more specific than an argument that gets along with a proper subset $\{\ \mathsf{P}_i(\mathsf{a}) \mid i \in I\ \}$ for some index set $I \subsetneq \{0,\ldots,n\}$. The effect of the assumption of this intention is called *preference of the "more precise"*.

There is, however, an exception to be observed where this analogy does not apply, namely the case that we actually can derive the set from its subset with the help of Π^{G}. In this case, the before-mentioned growth toward the leaves with rules from Π^{G} again implements the approximation of the subclass relation among model classes via the one among activation sets. This is demonstrated in the following example, which also nicely shows that a notion of specificity based only on single defeasible rules (without considering the context of the strict rules as a whole) cannot be intuitively adequate.

Example 3.3 (Stolzenburg et al. 2003, p. 95)
$\Pi^{\mathrm{F}}_{3.3} := \{\ \mathsf{q(a)}\ \}, \quad \Pi^{\mathrm{G}}_{3.3} := \{\ \mathsf{s}(x) \Leftarrow \mathsf{q}(x)\ \},$
$\Delta_{3.3} := \left\{ \begin{array}{l} \mathsf{p}(x) \leftarrow \mathsf{q}(x), \\ \neg\mathsf{p}(x) \leftarrow \mathsf{q}(x) \wedge \mathsf{s}(x) \end{array} \right\},$
$\mathcal{A}_1 := \{\ \neg\mathsf{p(a)} \leftarrow \mathsf{q(a)} \wedge \mathsf{s(a)}\ \},$
$\mathcal{A}_2 := \{\ \mathsf{p(a)} \leftarrow \mathsf{q(a)}\ \}$
The arguments $(\mathcal{A}_1, \neg\mathsf{p(a)})$ and $(\mathcal{A}_2, \mathsf{p(a)})$ are equivalently specific because the minimal activation set of each of them is $\{\mathsf{q(a)}\}$.

Apart from this exception, however, there is again a problem, namely that it is not the case that
$$\bigwedge_{i\in I} \mathsf{P}_i(\mathsf{a}) \not\models \bigwedge_{i\in\{0,\ldots,n\}} \mathsf{P}_i(\mathsf{a})$$
would be explicitly given by the specification via $(\Pi^{\mathrm{F}}, \Pi^{\mathrm{G}}, \Delta)$.

Nevertheless — if we do not just want to see preference of the "more precise" as a matter-of-fact property of notions of specificity in the style of Poole — we can again justify that it is unlikely that a seasoned specifier would not have intended this non-consequence statement, namely by an argumentation analogous to the one we gave for the preference of the "more concise". Indeed, a seasoned specifier who wants to exclude the above non-consequence would just specify a rule like $\mathsf{P}_j(x) \Leftarrow \bigwedge_{i\in I} \mathsf{P}_i(x)$, for each $j \in \{0,\ldots,n\}\backslash I$.

Example 3.4 *(continuing Example 3.3)*
For an example of preference of the "more precise" let us modify Example 3.3 by setting $\Pi^{\mathrm{F}}_{3.3} := \{\mathsf{q(a)}, \mathsf{s(a)}\}$ and $\Pi^{\mathrm{G}}_{3.3} := \emptyset$. Then $(\mathcal{A}_1, \neg\mathsf{p(a)})$ becomes properly more specific than $(\mathcal{A}_2, \mathsf{p(a)})$, because the latter argument has the additional activation set $\{\mathsf{q(a)}\}$.

4 Requirements Specification

With implicit reference to specification via $(\Pi^{\mathrm{F}}, \Pi^{\mathrm{G}}, \Delta)$, let us designate Poole's relation of being more (or equivalently) specific by "$\lesssim_{\mathrm{P1}}$". Here, "P1" stands for "Poole's original version".

The standard usage of the symbol "$\lesssim$" is to denote a *quasi-ordering* (cf. Sect. 2). Instead of the symbol "$\lesssim$", however, Poole (1985) uses the symbol "$\leq$". The standard usage of the symbol "$\leq$" is to denote a *reflexive ordering* (cf. Sect. 2). We cannot conclude from this, however, that Poole intended the additional property of anti-symmetry; indeed, we find a concrete example specification in (Poole 1985) where the lack of anti-symmetry of $\lesssim_{\mathrm{P1}}$ is made explicit (see last three sentences of Sect. 3.2 in (Poole 1985, p.145)).

The possible lack of anti-symmetry of quasi-orderings — i.e. that different arguments may have an equivalent specificity — cannot be a problem because any quasi-ordering $\lesssim_N$ immediately provides us with its equivalence $\approx_N$, its ordering $<_N$, and its reflexive ordering $\leq_N$ (cf. Corollary 2.3).

By contrast to the non-intended anti-symmetry, *transitivity* is obviously a *conditio sine qua non* for any useful notion of specificity. Indeed, if we already have an argument $(\mathcal{A}_2, \mathsf{wine})$ that is more specific than another argument $(\mathcal{A}_3, \mathsf{vodka})$, and if we come up with yet another argument $(\mathcal{A}_1, \mathsf{beer})$ that is even more specific than $(\mathcal{A}_2, \mathsf{wine})$, then, by all means, $(\mathcal{A}_1, \mathsf{beer})$ should be more specific than the argument $(\mathcal{A}_3, \mathsf{vodka})$ as well. It is obvious that a notion of specificity without transitivity could hardly be helpful in practice.

A further *conditio sine qua non* for any useful notion of specificity is that the conjunctive combination of respectively more specific arguments results in a more specific argument. Indeed, if a square is more specific than a rectangle and a circle is more specific than an ellipse, then a square inscribed into a circle should be more specific than a rectangle inscribed into an ellipse. This property is called monotonicity of conjunction, which we discuss in Sect. 6.2.

5 Formalizations of Specificity

A generative, bottom-up (i.e. from the leaves to the root) derivation with defeasible rules can now be split into three phases of derivation of literals from literals. This splitting follows the discussion in Sect. 3 on how to isolate the defeasible parts of a derivation (phase 2) from strict parts that may occur toward the root (phase 3) and toward the leaves (phase 1):

(phase 1) First we derive the literals that provide the basis for specificity considerations.

In our approach we derive the set $\mathfrak{T}_\Pi$ here. Poole takes the set $\mathfrak{T}_{\Pi\cup\Delta}$ instead.

(phase 2) On the basis of

- a subset H of the literals derived in phase 1,
- the first item $\mathcal{A}$ of a given argument $(\mathcal{A}, L)$, and
- the general rules Π^{G},

we derive a further set of literals $\mathfrak{L}$: $H\cup\mathcal{A}\cup\Pi^{\mathrm{G}}\vdash\mathfrak{L}$.

(phase 3) Finally, on the basis of $\mathfrak{L}$, the literal of the argument is derived: $\mathfrak{L}\cup\Pi\ \vdash\ \{L\}$.

In Poole's approach, phase 3 is empty and we simply have $\mathfrak{L}=\{L\}$. In our approach, however, it is admitted to use the facts from Π^{F} in phase 3, in addition to the general rules from Π^{G}, which were already admitted in phase 2.

With implicit reference to our sets $\Pi = \Pi^{\mathrm{F}} \cup \Pi^{\mathrm{G}}$ and Δ, the phases 2 and 3 can be more easily expressed with the help of the following notions.

Definition 5.1 (Activation Set)
Let $\mathcal{A}$ be a set of ground instances of rules from Δ, and let L be a literal. H is a *simplified activation set for* $(\mathcal{A}, L)$ if $L \in \mathfrak{T}_{H\cup\mathcal{A}\cup\Pi^{\mathrm{G}}}$. H is an *activation set for* $(\mathcal{A}, L)$ if $L \in \mathfrak{T}_{\mathfrak{L}\cup\Pi}$ for some $\mathfrak{L} \subseteq \mathfrak{T}_{H\cup\mathcal{A}\cup\Pi^{\mathrm{G}}}$. H is a *minimal* [*simplified*] *activation set for* $(\mathcal{A}, L)$ if H is an [simplified] activation set for $(\mathcal{A}, L)$, but no proper subset of H is an [simplified] activation set for $(\mathcal{A}, L)$.

Roughly speaking, an argument is now more (or equivalently) specific than another one if, for each of its activation sets H_1, the same set H_1 is also an activation set for the other argument. Note that we have replaced here the option of some $H_2 \subseteq \mathfrak{T}_{H_1\cup\Pi^{\mathrm{G}}}$ of the first straightforward sketch for a notion of specificity displayed in Sect. 3 with the more restrictive $H_2 = H_1$. Indeed, this simplification applies here because all we consider from any activation set H in Definition 5.1 (such as H_2 in this case) is just the closure $\mathfrak{T}_{H\cup\mathcal{A}\cup\Pi^{\mathrm{G}}} = \mathfrak{T}_{\mathfrak{T}_{H\cup\Pi^{\mathrm{G}}}\cup\mathcal{A}\cup\Pi^{\mathrm{G}}}$.

Activation sets that are not simplified differ from simplified ones by the admission of facts from Π^{F} (in addition to the general rules Π^{G}) after the defeasible part of the derivation is completed (cf. Example 6.6 for an occurrence of this difference).

Our introduction of activation sets that are not simplified is a conceptually important correction of Poole's approach: It must be admitted to use the facts besides the general rules in a purely strict derivation that is based on literals resulting from completed defeasible arguments, simply because the defeasible parts of a derivation (cf. Sect. 3) should not get more specific by the later use of additional facts that do not provide input to the defeasible parts.

5.1 Poole's Specificity Relations P1, P2, P3

In this section we will define the binary relations $\lesssim_{\mathrm{P1}}$, $\lesssim_{\mathrm{P2}}$, $\lesssim_{\mathrm{P3}}$ of "being more or equivalently specific according to David Poole" with implicit reference to our sets of facts and of general and defeasible rules (i.e. to Π^{F}, Π^{G}, and Δ, respectively).

The relation $\lesssim_{\mathrm{P1}}$ of the following definition is precisely Poole's original relation $\geq$ as defined at the bottom of the left column on page 145 of (Poole 1985). See Sect. 4 for our reasons to write "$\gtrsim$" instead of "$\geq$" as a first change. Moreover, as a second change required by mathematical standards, we have replaced the symbol "$\gtrsim$" with the symbol "$\lesssim$" (such that the smaller argument becomes the more specific one), so that the relevant well-foundedness becomes the one of its ordering $<$ instead of the reverse $>$.

Definition 5.2 $(\mathcal{A}_1, L_1) \lesssim_{\mathrm{P1}} (\mathcal{A}_2, L_2)$ if $(\mathcal{A}_1, L_1)$ and $(\mathcal{A}_2, L_2)$ are arguments, and if, for every $H \subseteq \mathfrak{T}_{\Pi\cup\Delta}$ that is a simplified activation set for $(\mathcal{A}_1, L_1)$ but not a simplified activation set for $(\mathcal{A}_2, L_1)$, H is also a simplified activation set for $(\mathcal{A}_2, L_2)$.

The relation $\lesssim_{\mathrm{P2}}$ of the following definition is the relation $\succeq$ of Definition 10 on page 94 of (Stolzenburg et al. 2003) (attributed to Poole 1985). Moreover, the relation $>_{\mathrm{spec}}$ of Definition 2.12 on page 132 of (Simari and Loui 1992) (attributed to (Poole 1985) as well) is the relation $<_{\mathrm{P2}} := \lesssim_{\mathrm{P2}} \setminus \gtrsim_{\mathrm{P2}}$.

Definition 5.3 $(\mathcal{A}_1, L_1) \lesssim_{\mathrm{P2}} (\mathcal{A}_2, L_2)$ if $(\mathcal{A}_1, L_1)$ and $(\mathcal{A}_2, L_2)$ are arguments, and if, for every $H \subseteq \mathfrak{T}_{\Pi\cup\Delta}$ that is a simplified activation set for $(\mathcal{A}_1, L_1)$ but not a simplified activation set for $(\emptyset, L_1)$, H is also a simplified activation set for $(\mathcal{A}_2, L_2)$.

The only change in Definition 5.3 as compared to Definition 5.2 is that "$(\mathcal{A}_2, L_1)$" is replaced with "$(\emptyset, L_1)$". We did not encounter any example yet where this most appropriate correction of the counter-intuitive variant "$(\mathcal{A}_2, L_1)$" of Definition 5.2 makes any difference to today's standard "$(\emptyset, L_1)$" in Definition 5.3.

The relations $\lesssim_{\mathrm{P1}}$ and $\lesssim_{\mathrm{P2}}$ were not meant to compare arguments for literals that do not need any defeasible rules — or at least they do not show an intuitive behavior on such arguments, as shown in Example 5.6 (right after the next definition and its corollary).

To overcome this minor flaw, we finally add an implication as an additional requirement in Definition 5.4. This implication guarantees that no argument that requires defeasible rules can be more specific than an argument not requiring any defeasible rules at all.

Definition 5.4 $(\mathcal{A}_1, L_1) \lesssim_{\mathrm{P3}} (\mathcal{A}_2, L_2)$ if $(\mathcal{A}_1, L_1)$ and $(\mathcal{A}_2, L_2)$ are arguments, $L_2 \in \mathfrak{T}_\Pi$ implies $L_1 \in \mathfrak{T}_\Pi$, and if, for every $H \subseteq \mathfrak{T}_{\Pi\cup\Delta}$ that is a [minimal] simplified activation set for $(\mathcal{A}_1, L_1)$ but not a simplified activation set for $(\emptyset, L_1)$, H is also a simplified activation set for $(\mathcal{A}_2, L_2)$.

Trivially, $\lesssim_{\mathrm{P3}} \subseteq \lesssim_{\mathrm{P2}}$. As every simplified activation set that passes the condition of Definition 5.2 passes the one of Definition 5.3, we have:

Corollary 5.5 $\lesssim_{\mathrm{P3}} \subseteq \lesssim_{\mathrm{P2}} \subseteq \lesssim_{\mathrm{P1}}$.

Example 5.6 (Minor Flaw of $\lesssim_{\mathrm{P1}}$ and $\lesssim_{\mathrm{P2}}$)
$\Pi^{\mathrm{F}}_{5.6} := \{\ \mathsf{thirst}\ \}$, $\Pi^{\mathrm{G}}_{5.6} := \{\ \mathsf{drink} \Leftarrow \mathsf{thirst}\ \}$,
$\Delta_{5.6} := \{\ \mathsf{beer} \leftarrow \mathsf{thirst}\ \}$, $\mathcal{A}_1 := \Delta_{5.6}$.
Let us compare the specificity of the arguments $(\mathcal{A}_1, \mathsf{beer})$ and $(\emptyset, \mathsf{drink})$. $\mathfrak{T}_{\Pi_{5.6}} = \{\mathsf{thirst}, \mathsf{drink}\}$, $\mathfrak{T}_{\Pi_{5.6}\cup\Delta_{5.6}} = \{\mathsf{beer}\} \cup \mathfrak{T}_{\Pi_{5.6}}$.
We have $(\mathcal{A}_1, \mathsf{beer}) \lesssim_{\mathrm{P2}} (\emptyset, \mathsf{drink})$ because for every $H \subseteq \mathfrak{T}_{\Pi_{5.6}\cup\Delta_{5.6}}$ that is a simplified activation set for $(\mathcal{A}_1, \mathsf{beer})$, but not a simplified activation set for $(\emptyset, \mathsf{beer})$, we have $H = \{\mathsf{thirst}\}$, which is a simplified activation set also for $(\emptyset, \mathsf{drink})$.
We have $(\emptyset, \mathsf{drink}) \lesssim_{\mathrm{P2}} (\mathcal{A}_1, \mathsf{beer})$ because there cannot be a simplified activation set for $(\emptyset, \mathsf{drink})$ that is not a simplified activation set for $(\emptyset, \mathsf{drink})$.
All in all, we get $(\mathcal{A}_1, \mathsf{beer}) \approx_{\mathrm{P2}} (\emptyset, \mathsf{drink})$, although $(\emptyset, \mathsf{drink}) <_{\mathrm{P3}} (\mathcal{A}_1, \mathsf{beer})$ should be given according to intuition, because, if beer produces a conflict with our drinking habits, there is no reason to prefer it to another drink. Finally note that by Corollary 5.5, we get $(\mathcal{A}_1, \mathsf{beer}) \approx_{\mathrm{P1}} (\emptyset, \mathsf{drink})$ as well.

Corollary 5.7 *If $(\mathcal{A}_1, L_1)$, $(\mathcal{A}_2, L_2)$ are arguments and we have $\mathcal{A}_1 \subseteq \mathcal{A}_2$ and $L_1 = L_2$, then we have $(\mathcal{A}_1, L_1) \lesssim_{\mathrm{P3}} (\mathcal{A}_2, L_2)$.*

By Corollaries 5.5 and 5.7, $\lesssim_{\mathrm{P1}}, \lesssim_{\mathrm{P2}}, \lesssim_{\mathrm{P3}}$ are reflexive relations, but — as we will show in Example 5.8 and state in Theorem 5.10 — not quasi-orderings in general.

Example 5.8 (Counterexample to Transitivities)
$\Pi^{\mathrm{F}}_{5.8} := \{\ \mathsf{alcohol},\ \mathsf{blessing},\ \mathsf{thirst}\ \}$,
$\Pi^{\mathrm{G}}_{5.8} := \{\ \mathsf{wine} \Leftarrow \mathsf{e}\ \}$, $\Delta_{5.8} := \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$,
$\mathcal{A}_1 := \left\{ \begin{array}{l} \mathsf{e} \leftarrow \mathsf{alcohol} \wedge \mathsf{blessing} \wedge \mathsf{thirst}, \\ \mathsf{beer} \leftarrow \mathsf{e} \end{array} \right\}$,
$\mathcal{A}_2 := \{\ \mathsf{wine} \leftarrow \mathsf{alcohol} \wedge \mathsf{blessing}\ \}$,
$\mathcal{A}_3 := \{\ \mathsf{vodka} \leftarrow \mathsf{alcohol}\ \}$.
Compare the arguments $(\mathcal{A}_1, \mathsf{beer})$, $(\mathcal{A}_2, \mathsf{wine})$, and $(\mathcal{A}_3, \mathsf{vodka})$!

Lemma 5.9 *There are a specification $(\Pi^{\mathrm{F}}, \Pi^{\mathrm{G}}, \Delta)$ without any negative literals (i.e. $\Pi^{\mathrm{F}} \cup \Pi^{\mathrm{G}} \cup \Delta$ is non-contradictory), and arguments $(\mathcal{A}_1, L_1)$, $(\mathcal{A}_2, L_2)$, $(\mathcal{A}_3, L_3)$ with respective minimal sets $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ (i.e. $(\mathcal{A}'_i, L_i)$ is not an argument for any proper subset $\mathcal{A}'_i \subsetneq \mathcal{A}_i$), such that*
$(\mathcal{A}_1, L_1) \lesssim_{\mathrm{P3}} (\mathcal{A}_2, L_2) \lesssim_{\mathrm{P3}} (\mathcal{A}_3, L_3) \not\gtrsim_{\mathrm{P1}} (\mathcal{A}_1, L_1)$ *and*
$(\mathcal{A}_1, L_1) \not\gtrsim_{\mathrm{P1}} (\mathcal{A}_2, L_2) \not\gtrsim_{\mathrm{P1}} (\mathcal{A}_3, L_3)$.

Proof of Lemma 5.9 Looking at Example 5.8, we see that only the quasi-ordering properties in the last two lines of Lemma 5.9 are non-trivial.
$\mathfrak{T}_{\Pi_{5.8}} = \{\mathsf{alcohol}, \mathsf{blessing}, \mathsf{thirst}\}$, $\mathfrak{T}_{\Pi_{5.8}\cup\Delta_{5.8}} = \{\mathsf{e}, \mathsf{beer}, \mathsf{wine}, \mathsf{vodka}\} \cup \mathfrak{T}_{\Pi_{5.8}}$. Thus, regarding the arguments $(\mathcal{A}_1, \mathsf{beer})$, $(\mathcal{A}_2, \mathsf{wine})$, $(\mathcal{A}_3, \mathsf{vodka})$, the additional implication condition of Definition 5.4 as compared to Definitions 5.2 and 5.3 is always satisfied, simply because its condition is always false.
$(\mathcal{A}_3, \mathsf{vodka}) \not\gtrsim_{\mathrm{P1}} (\mathcal{A}_1, \mathsf{beer}) \lesssim_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{wine})$: The minimal simplified activation sets for $(\mathcal{A}_1, \mathsf{beer})$ that are subsets of $\mathfrak{T}_{\Pi_{5.8}\cup\Delta_{5.8}}$ and no simplified activation sets for $(\emptyset, \mathsf{beer})$ (or, without any difference, for $(\mathcal{A}_3, \mathsf{beer})$) are $\{\mathsf{alcohol}, \mathsf{blessing}, \mathsf{thirst}\}$ and $\{\mathsf{e}\}$, which are simplified activation sets for $(\mathcal{A}_2, \mathsf{wine})$ — but $\{\mathsf{e}\}$ is no simplified activation set for $(\mathcal{A}_3, \mathsf{vodka})$.
$(\mathcal{A}_1, \mathsf{beer}) \not\gtrsim_{\mathrm{P1}} (\mathcal{A}_2, \mathsf{wine}) \lesssim_{\mathrm{P3}} (\mathcal{A}_3, \mathsf{vodka})$: The only minimal simplified activation set for $(\mathcal{A}_2, \mathsf{wine})$ that is a subset of $\mathfrak{T}_{\Pi_{5.8}\cup\Delta_{5.8}}$ and no simplified activation set for $(\emptyset, \mathsf{wine})$ (such as $\{\mathsf{e}\}$) (or, without any difference, for $(\mathcal{A}_1, \mathsf{wine})$) is $\{\mathsf{alcohol}, \mathsf{blessing}\}$, which is a simplified activation set for $(\mathcal{A}_3, \mathsf{vodka})$, but not for $(\mathcal{A}_2, \mathsf{beer})$.
$(\mathcal{A}_2, \mathsf{wine}) \not\gtrsim_{\mathrm{P1}} (\mathcal{A}_3, \mathsf{vodka})$: The only minimal simplified activation set for $(\mathcal{A}_3, \mathsf{vodka})$ that is a subset of $\mathfrak{T}_{\Pi_{5.8}\cup\Delta_{5.8}}$ and no simplified activation set for $(\mathcal{A}_2, \mathsf{vodka})$ is $\{\mathsf{alcohol}\}$, which is not a simplified activation set for $(\mathcal{A}_2, \mathsf{wine})$. **Q.e.d. (Lemma 5.9)**

The relations stated in Lemma 5.9 hold not only for the given indices, but — by Corollary 5.5 — actually for all of P1, P2, P3; and so we immediately get:

Theorem 5.10 *There is a specification $(\Pi^{\mathrm{F}}, \Pi^{\mathrm{G}}, \Delta)$, such that $\Pi^{\mathrm{F}} \cup \Pi^{\mathrm{G}} \cup \Delta$ is non-contradictory, but none of $\lesssim_{\mathrm{P1}}$, $\lesssim_{\mathrm{P2}}$, $\lesssim_{\mathrm{P3}}$, $<_{\mathrm{P1}}$, $<_{\mathrm{P2}}$, $<_{\mathrm{P3}}$ is transitive.*

Note that in the given counterexample to transitivity (Example 5.8) all arguments have minimal sets of ground instances of defeasible rules.

As a consequence of Theorem 5.10, the respective relations in (Simari and Loui 1992) and (Stolzenburg et al. 2003) are not transitive. This means that these relations are not quasi-orderings, let alone reflexive orderings. See (Wirth and Stolzenburg 2013, Sect. 5.2) for details.

5.2 Our Novel Specificity Ordering CP1

In the previous section, we have seen that *minor corrections* of Poole's original specificity relation P1 (such as P2, P3) do not cure the (up to our finding of Example 5.8) hidden and even denied formal deficiency of these relations, namely their lack of transitivity. Therefore, in this section, we now define our major correction of Poole's specificity — the binary relation $\lesssim_{\mathrm{CP1}}$ — with implicit reference to our sets of facts and of general and defeasible rules (i.e. to Π^{F}, Π^{G}, and Δ, respectively) as follows.

Definition 5.11 (Our Version of Specificity $\lesssim_{\mathrm{CP1}}$)
$(\mathcal{A}_1, L_1) \lesssim_{\mathrm{CP1}} (\mathcal{A}_2, L_2)$ if $(\mathcal{A}_1, L_1)$ and $(\mathcal{A}_2, L_2)$ are arguments, and we have
1. $L_1 \in \mathfrak{T}_\Pi$ or
2. $L_2 \notin \mathfrak{T}_\Pi$ and every $H \subseteq \mathfrak{T}_\Pi$ that is an [minimal] activation set for $(\mathcal{A}_1, L_1)$ is also an activation set for $(\mathcal{A}_2, L_2)$.

The crucial change in Definition 5.11 as compared to Definition 5.4 is not the merely technical emphasis it puts on the case "$L_1 \in \mathfrak{T}_\Pi$", which has no effect on the extension of the relation as compared to $\lesssim_{\mathrm{P3}}$. The crucial changes actually are

- the replacement of "$H \subseteq \mathfrak{T}_{\Pi\cup\Delta}$" with "$H \subseteq \mathfrak{T}_\Pi$", and the thereby enabled
- omission of the previously technically required, but unintuitive negative condition on derivability ("but not a simplified activation set for $(\emptyset, L_1)$").

An additional minor change, which we have already discussed in Sect. 5, is the one from simplified to (non-simplified) activation sets.

Corollary 5.12 *If $(\mathcal{A}_1, L_1)$, $(\mathcal{A}_2, L_2)$ are arguments and we have $\mathcal{A}_1 \subseteq \mathcal{A}_2$ and $L_1 = L_2$, then we have $(\mathcal{A}_1, L_1) \lesssim_{\mathrm{CP1}} (\mathcal{A}_2, L_2)$.*

Theorem 5.13
$\lesssim_{\mathrm{CP1}}$ is a quasi-ordering on arguments.

The proof of Theorem 5.13 is straightforward and can be found in (Wirth and Stolzenburg 2013, Sect. 5.3, Proof of Theorem 5.13).

Example 5.14 (continuing Example 5.8)
The following holds for our specification of Example 5.8 by Lemma 5.9 and Corollary 5.5:
$(\mathcal{A}_1, \mathsf{beer}) <_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{wine}) <_{\mathrm{P3}} (\mathcal{A}_3, \mathsf{vodka}) \not\gtrsim_{\mathrm{P3}}$
$(\mathcal{A}_1, \mathsf{beer})$. We have now:
$(\mathcal{A}_1, \mathsf{beer}) <_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{wine}) <_{\mathrm{CP1}} (\mathcal{A}_3, \mathsf{vodka}) >_{\mathrm{CP1}}$
$(\mathcal{A}_1, \mathsf{beer})$, simply because the trouble-making set $\{\mathsf{e}\}$ is not to be considered: it is not a subset of $\mathfrak{T}_{\Pi_{5.8}}$!

Obviously, an argument is ranked by $\lesssim_{\mathrm{CP1}}$ firstly on whether its literal is in $\mathfrak{T}_\Pi$, and, if not, secondly on the set of its activation sets, which is an element of the power set of the power set of $\mathfrak{T}_\Pi$. So we get:

Corollary 5.15
If $\mathfrak{T}_\Pi$ is finite, then $<_{\mathrm{CP1}}$ is well-founded.

Theorem 5.16 *$\lesssim_{\mathrm{P3}} \subseteq \lesssim_{\mathrm{CP1}}$ holds if, for each instance $L \Leftarrow L'_0 \wedge \ldots \wedge L'_n$ with $n \geq 1$ of each rule in Π^{G}, we have $L'_j \notin \mathfrak{T}_\Pi$ for all $j \in \{0, \ldots, n\}$.*

Please find the lengthy proof of Theorem 5.16 in (Wirth and Stolzenburg 2013, Sect. 5.4, Theorem 5.15). Here, however, is a typical application of it:

Example 5.17 (continuing Example 2.2)
We have $(\mathcal{A}_2, \mathsf{flies(edna)}) \not\lesssim_{\mathrm{CP1}} (\emptyset, \neg\mathsf{flies(edna)})$ because $\mathsf{flies(edna)} \notin \mathfrak{T}_{\Pi_{2.2}}$ and $\neg\mathsf{flies(edna)} \in \mathfrak{T}_{\Pi_{2.2}}$.
We have $(\emptyset, \neg\mathsf{flies(edna)}) \lesssim_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{flies(edna)})$, because $\neg\mathsf{flies(edna)} \in \mathfrak{T}_{\Pi_{2.2}}$ and because the premise of the last condition in Definition 5.4 is contradictory for $\mathcal{A}_1 := \emptyset$, and cannot be satisfied by any set $H \subseteq \mathfrak{T}_{\Pi_{2.2}\cup\Delta_{2.2}}$. All in all, by Theorem 5.16
we get $(\emptyset, \neg\mathsf{flies(edna)}) <_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{flies(edna)})$
and $(\emptyset, \neg\mathsf{flies(edna)}) <_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{flies(edna)})$.

6 Specificity and Human Intuition

Let us now put the two notions of specificity — as formalized in the two binary relations $\lesssim_{\mathrm{P3}}$ and $\lesssim_{\mathrm{CP1}}$ — to test w.r.t. our changed phase 1 of Sect. 5 in a series of classical examples.

6.1 Preference of the "More Concise"

Example 6.1 *(continuing Example 3.1)*
Let us compare the arguments $(\mathcal{A}_1, \neg\mathsf{flies(edna)})$ and $(\mathcal{A}_2, \mathsf{flies(edna)})$. $\mathfrak{T}_{\Pi_{3.1}} = \{\mathsf{bird(tweety)}, \mathsf{emu(edna)}, \mathsf{bird(edna)}\}$, $\mathfrak{T}_{\Pi_{3.1}\cup\Delta_{3.1}} = \{\neg\mathsf{flies(edna)}, \mathsf{flies(edna)}, \mathsf{flies(tweety)}\} \cup \mathfrak{T}_{\Pi_{3.1}}$. We have
$(\mathcal{A}_2, \mathsf{flies(edna)}) \not\lesssim_{\mathrm{CP1}} (\mathcal{A}_1, \neg\mathsf{flies(edna)})$
because $\mathsf{flies(edna)} \notin \mathfrak{T}_{\Pi_{3.1}}$ and because $\{\mathsf{bird(edna)}\} \subseteq \mathfrak{T}_{\Pi_{3.1}}$ is an activation set for $(\mathcal{A}_2, \mathsf{flies(edna)})$, but not for $(\mathcal{A}_1, \neg\mathsf{flies(edna)})$. We have
$(\mathcal{A}_1, \neg\mathsf{flies(edna)}) \lesssim_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{flies(edna)})$,
because $\mathsf{flies(edna)} \notin \mathfrak{T}_{\Pi_{3.1}}$ and, if $H \subseteq \mathfrak{T}_{\Pi_{3.1}\cup\Delta_{3.1}}$ is a simplified activation set for $(\mathcal{A}_1, \neg\mathsf{flies(edna)})$, but not for $(\emptyset, \neg\mathsf{flies(edna)})$, then we have $\mathsf{emu(edna)} \in H$, and thus H is a simplified activation set also for $(\mathcal{A}_2, \mathsf{flies(edna)})$. By Theorem 5.16, we get $(\mathcal{A}_1, \neg\mathsf{flies(edna)}) <_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{flies(edna)})$
and $(\mathcal{A}_1, \neg\mathsf{flies(edna)}) <_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{flies(edna)})$.

Example 6.2 *(continuing Example 3.2)*
Let us compare the arguments $(\mathcal{A}_1, \neg\mathsf{flies(edna)})$ and $(\mathcal{A}_2, \mathsf{flies(edna)})$. $\mathfrak{T}_{\Pi_{3.2}} = \{\mathsf{emu(edna)}\}$, $\mathfrak{T}_{\Pi_{3.2}\cup\Delta_{3.2}} = \{\mathsf{bird(edna)}, \mathsf{flies(edna)}, \neg\mathsf{flies(edna)}\} \cup \mathfrak{T}_{\Pi_{3.2}}$.
We have $(\mathcal{A}_2, \mathsf{flies(edna)}) \lesssim_{\mathrm{CP1}} (\mathcal{A}_1, \neg\mathsf{flies(edna)})$ because $\neg\mathsf{flies(edna)} \notin \mathfrak{T}_{\Pi_{3.2}}$ and, for every activation set $H \subseteq \mathfrak{T}_{\Pi_{3.2}}$ for $(\mathcal{A}_2, \mathsf{flies(edna)})$, we get $\mathsf{emu(edna)} \in H$, so H is an activation set also for $(\mathcal{A}_1, \neg\mathsf{flies(edna)})$.
$(\mathcal{A}_2, \mathsf{flies(edna)}) \not\lesssim_{\mathrm{P3}} (\mathcal{A}_1, \neg\mathsf{flies(edna)})$ is still the case because $\{\mathsf{bird(edna)}\} \subseteq \mathfrak{T}_{\Pi_{3.2}\cup\Delta_{3.2}}$ is a simplified activation set for $(\mathcal{A}_2, \mathsf{flies(edna)})$, but neither for $(\emptyset, \mathsf{flies(edna)})$, nor for $(\mathcal{A}_1, \neg\mathsf{flies(edna)})$.
We have $(\mathcal{A}_1, \neg\mathsf{flies(edna)}) \lesssim_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{flies(edna)})$, because of $\mathsf{flies(edna)} \notin \mathfrak{T}_{\Pi_{3.2}}$ and because, if $H \subseteq \mathfrak{T}_{\Pi_{3.2}\cup\Delta_{3.2}}$ is a simplified activation set for $(\mathcal{A}_1, \neg\mathsf{flies(edna)})$, but not for $(\emptyset, \neg\mathsf{flies(edna)})$, then we have $\mathsf{emu(edna)} \in H$ and thus H is a simplified activation set also for $(\mathcal{A}_2, \mathsf{flies(edna)})$.
All in all, by Theorem 5.16, this time
we get $(\mathcal{A}_1, \neg\mathsf{flies(edna)}) \approx_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{flies(edna)})$
and $(\mathcal{A}_1, \neg\mathsf{flies(edna)}) <_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{flies(edna)})$.
From a conceptual point of view, we have to ask ourselves, whether we would like a *defeasible* rule instance such as $\mathsf{bird(edna)} \leftarrow \mathsf{emu(edna)}$ to reduce the specificity of $\mathcal{A}_2$ as compared to a system that seems equivalent for the given argument for $\mathsf{flies(edna)}$, namely the argument
$(\{\mathsf{flies(edna)} \leftarrow \mathsf{emu(edna)}\}, \mathsf{flies(edna)})$?
Does the specificity of a defeasible reasoning step really reduce if we introduce intermediate literals?

According to human intuition, this question has a negative answer, as we have already explained at the end of Sect. 3.1. Moreover, Example 6.3 will exhibit a strong reason to deny it.

Finally, see Example 6.5 for another example that makes even clearer why defeasible rules should be considered for their global semantical effect instead of their syntactical fine structure.

6.2 Monotonicity w.r.t. Conjunction

Monotonicity w.r.t. conjunction means for a quasi-ordering $\lesssim_N$ that, in case of $(\mathcal{A}_1^i, L_1^i) \lesssim_N (\mathcal{A}_2^i, L_2^i)$ for $i \in \{1,2\}$, we always have $(\mathcal{A}_1^1 \cup \mathcal{A}_1^2, L_1') \lesssim_N (\mathcal{A}_2^i \cup \mathcal{A}_2^i, L_2')$ for fresh constant literals L_j' with additional general rules $L_j' \Leftarrow L_j^1 \wedge L_j^2 \in \Pi^{\mathrm{G}}$ $(j \in \{1,2\})$. For $\lesssim_{\mathrm{CP1}}$, this property is trivially given in case of $L_1^1, L_1^2 \in \mathfrak{T}_\Pi$, but cannot be expected in case of $L_1^i \notin \mathfrak{T}_\Pi \ni L_1^{3-i}$(for some $i \in \{1,2\}$), simply because then we get $L_1' \notin \mathfrak{T}_\Pi$. Also for the only remaining and most interesting case of $L_1^1, L_1^2 \notin \mathfrak{T}_\Pi$, this property is obviously given. For $\lesssim_{\mathrm{P1}}$, however, monotonicity is not even given in this most interesting case, as already noted in (Poole 1985):

Example 6.3 (Poole 1985, Example 6)
$\Pi_{6.3}^{\mathrm{F}} := \left\{ \begin{array}{l} \mathsf{a}, \\ \mathsf{d} \end{array} \right\}$, $\Pi_{6.3}^{\mathrm{G}} := \left\{ \begin{array}{l} \mathsf{g}_1 \Leftarrow \neg\mathsf{c} \wedge \neg\mathsf{f}, \\ \mathsf{g}_2 \Leftarrow \mathsf{c} \wedge \mathsf{f} \end{array} \right\}$,
$\Delta_{6.3} := \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{A}_1 := \{\ \neg\mathsf{c} \leftarrow \mathsf{a},\ \neg\mathsf{f} \leftarrow \mathsf{d}\ \}$,
$\mathcal{A}_2 := \{\ \mathsf{b} \leftarrow \mathsf{a},\ \mathsf{c} \leftarrow \mathsf{b},\ \mathsf{e} \leftarrow \mathsf{d},\ \mathsf{f} \leftarrow \mathsf{e}\ \}$
Let us compare the specificity of the arguments $(\mathcal{A}_1, \mathsf{g}_1)$ and $(\mathcal{A}_2, \mathsf{g}_2)$. $\mathfrak{T}_{\Pi_{6.3}} = \{\mathsf{a}, \mathsf{d}\}$. $\mathfrak{T}_{\Pi_{6.3} \cup \Delta_{6.3}} =$ $\{\mathsf{b},\mathsf{c},\mathsf{e},\mathsf{f}, \mathsf{g}_1, \mathsf{g}_2, \neg\mathsf{c}, \neg\mathsf{f}\} \cup \mathfrak{T}_{\Pi_{6.3}}$. $(\mathcal{A}_1, \mathsf{g}_1) \approx_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{g}_2)$ because $H \subseteq \mathfrak{T}_{\Pi_{6.3}}$ is an activation set for $(\mathcal{A}_i, \mathsf{g}_i)$ if and only if $H = \{\mathsf{a}, \mathsf{d}\}$. We have $(\mathcal{A}_1, \mathsf{g}_1) \triangle_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{g}_2)$: $\{\mathsf{a}, \neg\mathsf{f}\} \subseteq \mathfrak{T}_{\Pi_{6.3} \cup \Delta_{6.3}}$ is a simplified activation set for $(\mathcal{A}_1, \mathsf{g}_1)$, but neither for $(\emptyset, \mathsf{g}_1)$, nor for $(\mathcal{A}_2, \mathsf{g}_2)$. $\{\mathsf{a}, \mathsf{f}\} \subseteq \mathfrak{T}_{\Pi_{6.3} \cup \Delta_{6.3}}$ is a simplified activation set for $(\mathcal{A}_2, \mathsf{g}_2)$, but neither for $(\emptyset, \mathsf{g}_2)$, nor for $(\mathcal{A}_1, \mathsf{g}_1)$.

In (Poole 1985), the same result for $\lesssim_{\mathrm{P1}}$ is described as "seemingly unintuitive", because, as we have seen in the isomorphic sub-specification of Example 3.2, we have both $(\mathcal{A}_1, \neg\mathsf{c}) <_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{c})$ and $(\mathcal{A}_1, \neg\mathsf{f}) <_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{f})$. Indeed, as already listed as an essential requirement in Sect. 4, the conjunction of two respectively more specific derivations should be more specific. On the other hand, considering $\lesssim_{\mathrm{CP1}}$ instead of $\lesssim_{\mathrm{P3}}$, the conjunction of two equivalently specific derivations results in an equivalently specific derivation — exactly as one intuitively expects.

Example 6.4 (1st Variation of Example 6.3)
$\Pi_{6.4}^{\mathrm{F}} := \Pi_{6.3}^{\mathrm{F}}$, $\mathcal{A}_1 := \{\ \neg\mathsf{c} \leftarrow \mathsf{a},\ \neg\mathsf{f} \leftarrow \mathsf{d}\ \}$,
$\Pi_{6.4}^{\mathrm{G}} := \{\ \mathsf{g}_1 \Leftarrow \neg\mathsf{c} \wedge \neg\mathsf{f},\ \mathsf{g}_2 \Leftarrow \mathsf{c} \wedge \mathsf{f},\ \mathsf{b} \Leftarrow \mathsf{a}\ \}$,
$\Delta_{6.4} := \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{A}_2 := \{\mathsf{c} \leftarrow \mathsf{b},\ \mathsf{e} \leftarrow \mathsf{d},\ \mathsf{f} \leftarrow \mathsf{e}\}$.
Let us compare the specificity of the arguments $(\mathcal{A}_1, \mathsf{g}_1)$ and $(\mathcal{A}_2, \mathsf{g}_2)$. $\mathfrak{T}_{\Pi_{6.4}} = \{\mathsf{a}, \mathsf{b}, \mathsf{d}\}$. $\mathfrak{T}_{\Pi_{6.4} \cup \Delta_{6.4}} =$ $\{\mathsf{c}, \mathsf{e}, \mathsf{f}, \mathsf{g}_1, \mathsf{g}_2, \neg\mathsf{c}, \neg\mathsf{f}\} \cup \mathfrak{T}_{\Pi_{6.4}}$. $(\mathcal{A}_2, \mathsf{g}_2) \not\lesssim_{\mathrm{CP1}} (\mathcal{A}_1, \mathsf{g}_1)$ because $\{\mathsf{b}, \mathsf{d}\} \subseteq \mathfrak{T}_{\Pi_{6.4}}$ is an activation set for $(\mathcal{A}_2, \mathsf{g}_2)$, but not for $(\mathcal{A}_1, \mathsf{g}_1)$. $(\mathcal{A}_1, \mathsf{g}_1) \lesssim_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{g}_2)$ because, for any activation set $H \subseteq \mathfrak{T}_{\Pi_{6.4}}$ for $(\mathcal{A}_1, \mathsf{g}_1)$, we have $\{\mathsf{a}, \mathsf{b}\} \subseteq H$; so H is also an activation set for $(\mathcal{A}_2, \mathsf{g}_2)$.

Again $(\mathcal{A}_1, \mathsf{g}_1) \triangle_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{g}_2)$, for the same reason as in Example 6.3. Thus, the situation for $\lesssim_{\mathrm{P3}}$ is just as in Example 6.3, and just as "seemingly unintuitive" for exactly the same reason.

We have $(\mathcal{A}_1, \mathsf{g}_1) <_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{g}_2)$, which is intuitive because the conjunction of a more specific and an equivalently specific element, respectively, should be more specific. Indeed, from the isomorphic sub-specifications in Examples 3.1 and 3.2, we know that $(\mathcal{A}_1, \neg\mathsf{c}) <_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{c})$ and $(\mathcal{A}_1, \neg\mathsf{f}) \approx_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{f})$, resp.

All in all, $\lesssim_{\mathrm{P3}}$ fails in this example again, whereas $\lesssim_{\mathrm{CP1}}$ satisfies the monotonicity w.r.t. conjunction required in Sect. 4.

6.3 Preference of the "More Precise"

As primary sources of differences in specificity, the previous examples illustrated the effect of a chain of implications. We now consider examples where the primary source is an essentially required condition that is a super-conjunction of the condition of another rule.

Example 6.5 (2nd Variation of Example 6.3)
$\Pi_{6.5}^{\mathrm{F}} := \Pi_{6.3}^{\mathrm{F}}$, $\Pi_{6.5}^{\mathrm{G}} := \{\ \mathsf{g}_1 \Leftarrow \neg\mathsf{c},\ \mathsf{g}_2 \Leftarrow \mathsf{c} \wedge \mathsf{f}\ \}$,
$\Delta_{6.5} := \mathcal{A}_1 \cup \mathcal{A}_2$, $\mathcal{A}_1 := \{\ \neg\mathsf{c} \leftarrow \mathsf{a}\ \}$,
$\mathcal{A}_2 := \{\ \mathsf{b} \leftarrow \mathsf{a},\ \mathsf{c} \leftarrow \mathsf{b},\ \mathsf{e} \leftarrow \mathsf{d},\ \mathsf{f} \leftarrow \mathsf{e}\ \}$
Let us compare the specificity of the arguments $(\mathcal{A}_1, \mathsf{g}_1)$ and $(\mathcal{A}_2, \mathsf{g}_2)$. $\mathfrak{T}_{\Pi_{6.5}} = \{\mathsf{a}, \mathsf{d}\}$. $\mathfrak{T}_{\Pi_{6.5} \cup \Delta_{6.5}} = \{\mathsf{b}, \mathsf{c}, \mathsf{e}, \mathsf{f}, \mathsf{g}_1, \mathsf{g}_2, \neg\mathsf{c}\} \cup \mathfrak{T}_{\Pi_{6.5}}$. We have $(\mathcal{A}_1, \mathsf{g}_1) \not\lesssim_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{g}_2)$ because $\{\mathsf{a}\} \subseteq \mathfrak{T}_{\Pi_{6.5}}$ is an activation set for $(\mathcal{A}_1, \mathsf{g}_1)$, but not for $(\mathcal{A}_2, \mathsf{g}_2)$. We have $(\mathcal{A}_2, \mathsf{g}_2) \lesssim_{\mathrm{CP1}} (\mathcal{A}_1, \mathsf{g}_1)$ because any activation set for $(\mathcal{A}_2, \mathsf{g}_2)$ that is a subset of $\mathfrak{T}_{\Pi_{6.5}}$ includes a, and so is also an activation set for $(\mathcal{A}_1, \mathsf{g}_1)$. Considering Theorem 5.16 and the the activation set $\{\mathsf{b}, \mathsf{d}\}$ for $(\mathcal{A}_2, \mathsf{g}_2)$, we see $(\mathcal{A}_1, \mathsf{g}_1) \triangle_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{g}_2)$.

All in all, $\lesssim_{\mathrm{CP1}}$ realizes the intuition that the super-conjunction $\mathsf{a} \wedge \mathsf{d}$ — which is essential to derive $\mathsf{c} \wedge \mathsf{f}$ according to $\mathcal{A}_2$ — is more specific than the "less precise" a.

Just like Example 3.2, this example shows again that $\lesssim_{\mathrm{P3}}$ does not really implement the intuition that defeasible rules should be considered for their global semantical effect instead of their syntactical fine structure.

Example 6.6 (Stolzenburg et al. 2003, p. 96)
$\Pi_{6.6}^{\mathrm{F}} := \{\ \mathsf{c}, \mathsf{d}, \mathsf{e}\ \}$, $\Pi_{6.6}^{\mathrm{G}} := \{\ \mathsf{x} \Leftarrow \mathsf{a} \wedge \mathsf{f}\ \}$,
$\Delta_{6.6} := \mathcal{A}^1 \cup \mathcal{A}^2 \cup \mathcal{A}^3 \cup \mathcal{A}^4 \cup \mathcal{A}^5$,
$\mathcal{A}^1 := \{\ \mathsf{x} \leftarrow \mathsf{a} \wedge \mathsf{b} \wedge \mathsf{c}\ \}$, $\mathcal{A}^2 := \{\ \neg\mathsf{x} \leftarrow \mathsf{a} \wedge \mathsf{b}\ \}$,
$\mathcal{A}^3 := \{\mathsf{f} \leftarrow \mathsf{e}\}$, $\mathcal{A}^4 := \{\mathsf{a} \leftarrow \mathsf{d}\}$, $\mathcal{A}^5 := \{\mathsf{b} \leftarrow \mathsf{e}\}$.
Let us compare the specificity of the arguments $(\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x})$, $(\mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \neg\mathsf{x})$, $(\mathcal{A}^3 \cup \mathcal{A}^4, \mathsf{x})$. $\mathfrak{T}_{\Pi_{6.6}} = \{\mathsf{c}, \mathsf{d}, \mathsf{e}\}$. $\mathfrak{T}_{\Pi_{6.6} \cup \Delta_{6.6}} = \{\mathsf{a}, \mathsf{b}, \mathsf{f}, \mathsf{x}, \neg\mathsf{x}\} \cup \mathfrak{T}_{\Pi_{6.6}}$.
We have $(\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x}) <_{\mathrm{CP1}} (\mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \neg\mathsf{x}) \approx_{\mathrm{CP1}} (\mathcal{A}^3 \cup \mathcal{A}^4, \mathsf{x})$, because of $\mathsf{x}, \neg\mathsf{x} \notin \mathfrak{T}_{\Pi_{6.6}}$, and because any activation set $H \subseteq \mathfrak{T}_{\Pi_{6.6}}$ for any of $(\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x})$, $(\mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \neg\mathsf{x})$, $(\mathcal{A}^3 \cup \mathcal{A}^4, \mathsf{x})$ contains $\{\mathsf{d}, \mathsf{e}\}$, which is an activation set only for the latter two. This matches our intuition well, because the first of these arguments essentially requires the "more precise" $\mathsf{c} \wedge \mathsf{d} \wedge \mathsf{e}$ instead of the less specific $\mathsf{d} \wedge \mathsf{e}$.

We have $(\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x}) \triangle_{\mathrm{P3}} (\mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \neg\mathsf{x}) \triangle_{\mathrm{P3}} (\mathcal{A}^3 \cup \mathcal{A}^4, \mathsf{x}) \triangle_{\mathrm{P3}} (\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x})$, however. This means that $\lesssim_{\mathrm{P3}}$ cannot compare these counterarguments and cannot help us to pick the more specific argument. What is most interesting under the computational aspect is that, for realizing

$$(\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x}) \not\lesssim_{\mathrm{P3}} (\mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \neg\mathsf{x}),$$

we have to consider the defeasible rule of $\mathcal{A}^3$ (implicitly via $\{\mathsf{d}, \mathsf{f}\} \subseteq \mathfrak{T}_{\Pi_{6.6} \cup \Delta_{6.6}}$), which is not part of any of the two arguments under comparison. Note that such considerations are not required, however, for realizing the properties of $\lesssim_{\mathrm{CP1}}$, because defeasible rules not in the given argument can be completely ignored when calculating the minimal activation sets as subsets of $\mathfrak{T}_\Pi$ instead of $\mathfrak{T}_{\Pi \cup \Delta}$. This means in particular that the complication of *pruning* — as discussed in detail by Stolzenburg et al. (2003, Sect. 3.3) — does not have to be considered for the operationalization of $\lesssim_{\mathrm{CP1}}$.

Example 6.7 (Variation of Example 6.6)
$\Pi^{\mathrm{F}}_{6.7} := \{\ \mathsf{c}, \mathsf{d}, \mathsf{e}\ \}$, $\Pi^{\mathrm{G}}_{6.7} := \{\ \mathsf{x} \Leftarrow \mathsf{a} \wedge \mathsf{f},\ \mathsf{f} \Leftarrow \mathsf{e}\ \}$,
$\Delta_{6.7} := \mathcal{A}^1 \cup \mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5$,
$\mathcal{A}^1 := \{\ \mathsf{x} \leftarrow \mathsf{a} \wedge \mathsf{b} \wedge \mathsf{c}\ \}$, $\mathcal{A}^2 := \{\ \neg\mathsf{x} \leftarrow \mathsf{a} \wedge \mathsf{b}\ \}$,
$\mathcal{A}^4 := \{\ \mathsf{a} \leftarrow \mathsf{d}\ \}$, $\mathcal{A}^5 := \{\ \mathsf{b} \leftarrow \mathsf{e}\ \}$.
Let us compare the specificity of the arguments $(\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x})$, $(\mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \neg\mathsf{x})$, $(\mathcal{A}^4, \mathsf{x})$. $\mathfrak{T}_{\Pi_{6.7}} = \{\mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}\}$. $\mathfrak{T}_{\Pi_{6.7} \cup \Delta_{6.7}} = \{\mathsf{a}, \mathsf{b}, \mathsf{x}, \neg\mathsf{x}\} \cup \mathfrak{T}_{\Pi_{6.7}}$.
Obviously, $\mathsf{x}, \neg\mathsf{x} \notin \mathfrak{T}_{\Pi_{6.7}}$. Moreover, $\{\mathsf{d}\} \subseteq \mathfrak{T}_{\Pi_{6.7}}$ is an activation set for $(\mathcal{A}^4, \mathsf{x})$ (but not a simplified one) and, *a fortiori* (by Corollary 5.12), for $(\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x})$, but not for $(\mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \neg\mathsf{x})$. Furthermore, every activation set $H \subseteq \mathfrak{T}_{\Pi_{6.7}}$ for $(\mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \neg\mathsf{x})$ satisfies $\{\mathsf{d}, \mathsf{e}\} \subseteq H$, which is an activation set for $(\mathcal{A}^4, \mathsf{x})$ and $(\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x})$. Furthermore, every activation set $H \subseteq \mathfrak{T}_{\Pi_{6.7}}$ for $(\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x})$ satisfies $\{\mathsf{d}\} \subseteq H$ which is an activation set for $(\mathcal{A}^4, \mathsf{x})$. All in all, we have
$(\mathcal{A}^4, \mathsf{x}) \approx_{\mathrm{CP1}} (\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x}) >_{\mathrm{CP1}} (\mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \neg\mathsf{x})$.
This is intuitively sound because $(\mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \neg\mathsf{x})$ is activated only by the more specific $\mathsf{d} \wedge \mathsf{e}$, whereas $(\mathcal{A}^4, \mathsf{x})$ is activated also by the "less precise" d. Moreover, $\mathsf{c} \wedge \mathsf{d} \wedge \mathsf{e}$ is not essentially required for $(\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x})$, which thus is equivalent to $(\mathcal{A}^4, \mathsf{x})$.
We have $(\mathcal{A}^4, \mathsf{x}) <_{\mathrm{P3}} (\mathcal{A}^1 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \mathsf{x}) \triangle_{\mathrm{P3}} (\mathcal{A}^2 \cup \mathcal{A}^4 \cup \mathcal{A}^5, \neg\mathsf{x}) \triangle_{\mathrm{P3}} (\mathcal{A}^4, \mathsf{x})$, however. This means that $\lesssim_{\mathrm{P3}}$ fails here completely w.r.t. Poole's intuition.

Example 6.8 *(continuing Example 3.3)*
Let us compare the specificity of the arguments $(\mathcal{A}_1, \neg\mathsf{p}(\mathsf{a}))$ and $(\mathcal{A}_2, \mathsf{p}(\mathsf{a}))$. $\mathfrak{T}_{\Pi_{3.3}} = \{\mathsf{q}(\mathsf{a}), \mathsf{s}(\mathsf{a})\}$, $\mathfrak{T}_{\Pi_{3.3} \cup \Delta_{3.3}} = \{\mathsf{p}(\mathsf{a}), \neg\mathsf{p}(\mathsf{a})\} \cup \mathfrak{T}_{\Pi_{3.3}}$.
We have $(\mathcal{A}_1, \neg\mathsf{p}(\mathsf{a})) \approx_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{p}(\mathsf{a}))$, because of $\mathsf{p}(\mathsf{a}), \neg\mathsf{p}(\mathsf{a}) \notin \mathfrak{T}_{\Pi_{3.3}}$, and because, for $H \subseteq \mathfrak{T}_{\Pi_{3.3} \cup \Delta_{3.3}}$, $i \in \{1, 2\}$, $L_1 := \neg\mathsf{p}(\mathsf{a})$, and $L_2 := \mathsf{p}(\mathsf{a})$, we have the logical equivalence of $H = \{\mathsf{q}(\mathsf{a})\}$ on the one hand, and of H being a minimal simplified activation set for $(\mathcal{A}_i, L_i)$ but not for $(\emptyset, L_i)$, on the other hand. By Theorem 5.16, we also get $(\mathcal{A}_1, \neg\mathsf{p}(\mathsf{a})) \approx_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{p}(\mathsf{a}))$. This makes perfect sense because $\mathsf{q}(\mathsf{a}) \wedge \mathsf{s}(\mathsf{a})$ is not at all strictly "more precise" than $\mathsf{q}(\mathsf{a})$ in the context of $\Pi_{3.3}$.
Note that nothing is changed here if $\mathsf{s}(x) \Leftarrow \mathsf{q}(x)$ is replaced by setting $\Pi^{\mathrm{G}}_{3.3} := \{\mathsf{s}(\mathsf{a})\}$. If $\mathsf{s}(x) \Leftarrow \mathsf{q}(x)$ is replaced, however, by setting $\Pi^{\mathrm{G}}_{3.3} := \emptyset$ and $\Pi^{\mathrm{F}}_{3.3} := \{\mathsf{q}(\mathsf{a}), \mathsf{s}(\mathsf{a})\}$, then we get both $(\mathcal{A}_1, \neg\mathsf{p}(\mathsf{a})) <_{\mathrm{P3}} (\mathcal{A}_2, \mathsf{p}(\mathsf{a}))$ and $(\mathcal{A}_1, \neg\mathsf{p}(\mathsf{a})) <_{\mathrm{CP1}} (\mathcal{A}_2, \mathsf{p}(\mathsf{a}))$.

7 Relation to Other Approaches

Our new notion of specificity $\lesssim_{\mathrm{CP1}}$ follows the lines of Poole (1985). It is a transitive relation that provides us a quasi-ordering on arguments, which is as monotonic w.r.t. conjunction as can be expected (cf. Sect. 6.2). In addition, the effort for computing $\lesssim_{\mathrm{CP1}}$ is lower than that of $\lesssim_{\mathrm{P3}}$ because of $\mathfrak{T}_\Pi \subseteq \mathfrak{T}_{\Pi \cup \Delta}$, though not w.r.t. asymptotic complexity: In both cases already the number of possible (simplified) activation sets is exponential in the number of literals in the respective sets $\mathfrak{T}_\Pi$ and $\mathfrak{T}_{\Pi \cup \Delta}$, because in principle each possible subset has to be tested.

Stolzenburg et al. (2003, Definition 12) introduce the concept of pruning derivation trees because, for the case of $\lesssim_{\mathrm{P2}}$, attention cannot be restricted to derivations which make use only of the instances of defeasible rules given in the arguments. The reason for this is that the specificity notions of (Poole 1985) and (Simari and Loui 1992) admit literals L in activation sets that cannot be derived solely by strict rules, i.e. $L \in \mathfrak{T}_{\Pi \cup \Delta} \backslash \mathfrak{T}_\Pi$. Since this is not possible with the relation $\lesssim_{\mathrm{CP1}}$, this problem vanishes with our new version of specificity. See also Example 6.6.

Yet still, the new relation $\lesssim_{\mathrm{CP1}}$ inherits several properties from $\lesssim_{\mathrm{P3}}$. For instance, in general the specificity criterion requires us to compare sets of derivations, in principle all possible derivations for a given argument. This is true for both versions of the specificity relation. The reason for this complication is that we consider a very general setting of defeasible reasoning here, because — in contrast to other approaches (Gelfond and Przymusinska 1990; Dung and Son 1996; Benferhat and Garcia 1997) — we admit more than one antecedent in rules, i.e. bodies containing more than one literal, and (possibly) non-empty sets of background knowledge, namely the general rules in Π^{G} in addition to the facts in Π^{F}.

All in all, there are numerous frameworks for argumentation in logic. The overall process usually includes a dialectical process used for answering queries. Different arguments are pro or contra a certain answer. By means of an attack relation conflicts between contradicting arguments can be determined in abstract argumentation frameworks (Dung 1995; Prakken and Vreeswijk 2002). A concrete specificity or similar relation helps then to decide among conflicting arguments. There are a lot of works on abstract argumentation frameworks. However, as the discussion in this paper demonstrates, it is not that easy to find an effective concrete specificity relation. The main problem is that the orderings tend to be either computationally highly complex (Kern-Isberner and Thimm 2012) or not really appropriate for specificity (Besnard and Hunter 2001).

8 Conclusion

We would need further discussions on our most surprising new findings — after all, defeasible reasoning with Poole's notion of specificity is being applied now for over a quarter of century, and it was not to be expected that our investigations could discover a flaw in its foundations.

One remedy for the discovered lack of transitivity of $\lesssim_{\mathrm{P3}}$ could be to consider the transitive closure of the non-transitive relation $\lesssim_{\mathrm{P3}}$. Only under the condition of Theorem 5.16, the transitive closure of $\lesssim_{\mathrm{P3}}$ is a subset of $\lesssim_{\mathrm{CP1}}$, and therefore a possible choice. Moreover, it will still have all the intuitive shortcomings made obvious in Sect. 6. We do not see how this transitive closure could be decided efficiently. Furthermore, this transitive closure lacks a direct intuitive motivation, and after the first extension step from $\lesssim_{\mathrm{P3}}$ to its transitive closure, we had better take the second extension step to the more intuitive $\lesssim_{\mathrm{CP1}}$ immediately.

Finally, contrary to the transitive closure of $\lesssim_{\mathrm{P3}}$, our novel relation $\lesssim_{\mathrm{CP1}}$ also solves the problem of non-monotonicity of specificity w.r.t. conjunction (cf. Sect. 6.2) insofar as it was realized as a problem of $\lesssim_{\mathrm{P1}}$ in (Poole 1985).

Further work is needed to improve efficiency considerably. As a first step, we have narrowed the concept of $\lesssim_{\mathrm{CP1}}$ further down according to Sect. 3, resulting in a similar (no difference in any of the examples presented in this paper!), but more tractable relation $\lesssim_{\mathrm{CP2}}$. We are currently developing and testing strong methods for efficient operationalization of $\lesssim_{\mathrm{CP2}}$, which can hardly be found for any of $\lesssim_{\mathrm{P1}}, \lesssim_{\mathrm{P2}}, \lesssim_{\mathrm{P3}}, \lesssim_{\mathrm{CP1}}$.

References

Benferhat, S., and Garcia, L. 1997. A coherence-based approach to default reasoning. In Gabbay, D.; Kruse, R.; Nonnengart, A.; and Ohlbach, H.-J., eds., *Proc. 1st Int. Joint Conf. on Qualitative and Quantitative Practical Reasoning, 1997, June 9–12, Bad Honnef (Germany)*, number 1244 in Lecture Notes in Computer Science, 43–57. Springer. http://dx.doi.org/10.1007/BFb0035611.

Besnard, P., and Hunter, A. 2001. A logic-based theory of deductive arguments. *Artif. Intell.* 128(1-2):203–235.

Chesñevar, C. I.; Dix, J.; Stolzenburg, F.; and Simari, G. R. 2003. Relating defeasible and normal logic programming through transformation properties. *Theoretical Computer Sci.* 290:499–529. Received Jan. 8, 2001; rev. Nov. 9, 2001. http://dx.doi.org/10.1016/S0304-3975(02)00033-6.

Dung, P. M., and Son, T. C. 1996. An argumentation-theoretic approach to reasoning with specificity. In Aiello, L. C.; Doyle, J.; and Shapiro, S. C., eds., *Proc. 5th Int. Conf. on Principles of Knowledge Representation and Reasoning, 1996, Nov. 5–8, Cambridge (MA)*, 506–517. Morgan Kaufmann, Los Altos (CA) (Elsevier).

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77(2):321–358.

García, A. J., and Simari, G. R. 2004. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming, Cambridge Univ. Press* 4:95–138.

Gelfond, M., and Przymusinska, H. 1990. Formalization of inheritance reasoning in autoepistemic logic. *Fundamenta Informaticae* XIII:403–443.

Gillman, L. 1987. *Writing Mathematics Well*. The Mathematical Association of America.

Kern-Isberner, G., and Thimm, M. 2012. A ranking semantics for first-order conditionals. In Raedt, L. D.; Bessière, C.; Dubois, D.; Doherty, P.; Frasconi, P.; Heintz, F.; and Lucas, P. J. F., eds., *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, 456–461. IOS Press.

Poole, D. L. 1985. On the comparison of theories: Preferring the most specific explanation. In Joshi, A., ed., *Proc. 9th Int. Joint Conf. on Artificial Intelligence (IJCAI), 1985, Aug. 18–25, Los Angeles (CA)*. Morgan Kaufmann, Los Altos (CA) (Elsevier). http://ijcai.org/Past%20Proceedings.

Prakken, H., and Vreeswijk, G. 2002. Logics for defeasible argumentation. In Gabbay, D., ed., *Handbook of Philosophical Logic*. Kluwer Academic Publisher, 2nd edition. 218–319.

Simari, G. R., and Loui, R. P. 1992. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence* 53:125–157. Received Feb. 1990, rev. April 1991.

Stolzenburg, F.; García, A. J.; Chesñevar, C. I.; and Simari, G. R. 2003. Computing generalized specificity. *J. Applied Non-Classical Logics* 13:87–113. http://www.tandfonline.com/doi/abs/10.3166/jancl.13.87-113.

Wirth, C.-P., and Stolzenburg, F. 2013. *David Poole's Specificity Revised*. SEKI-Report SR–2013–01 (ISSN 1437–4447). DFKI Bremen GmbH, Safe and Secure Cognitive Systems, Cartesium, Enrique Schmidt Str. 5, D–28359 Bremen, Germany: SEKI Publications. pp. ii+22, http://arxiv.org/abs/1308.4943.

Axiomatizing Rationality

Adam Bjorndahl
Dept. of Mathematics
Cornell University
Ithaca, NY 14853
abjorndahl@math.cornell.edu

Joseph Y. Halpern
Dept. of Computer Science
Cornell University
Ithaca, NY 14853
halpern@cs.cornell.edu

Rafael Pass
Dept. of Computer Science
Cornell University
Ithaca, NY 14853
rafael@cs.cornell.edu

Abstract

We provide a sound and complete axiomatization for a class of logics appropriate for reasoning about the rationality of players in games. Essentially the same axiomatization applies to a wide class of decision rules.

1 Introduction

Decision-making in an uncertain environment is a fundamental component of game theory: players must choose what to do without necessarily knowing what their opponents will do. Under certainty, decision-making is straightforward; one simply chooses the course of action that leads to the most preferred outcome. Under uncertainty, however, a player must evaluate many possible outcomes in a manner that somehow takes into account her relative degrees of belief. There are many ways to do this. The *maximin* decision rule, for example, focuses entirely on worst-case scenarios. An *expected utility maximizer*, on the other hand, weights each outcome according to her (subjective) probability and chooses the course of action that maximizes the corresponding expected value.

One can argue about which decision rules are reasonable and which are not, and the word "rational" is often invoked to denote this very divide, but this is not the notion of rationality that concerns us here. Rather, assuming that we have fixed a decision rule for a given player, we can ask whether that player is, in fact, making choices in accordance with it, and call her *rational* precisely when she is. In classical game theory, for example, rationality is typically identified with expected utility maximization: a player is rational precisely if she is acting to maximize her expected utility.

Another fundamental component of game theory is higher-order belief: what you believe your opponent will *do* is connected with what you believe your opponent *believes*. In other words, your beliefs about their beliefs matter. Many notions of equilibrium, for example, require not only that each player is rational, but also that each player believes their opponents to be rational as well. Reasoning about games therefore requires reasoning about beliefs.

Epistemic game theory takes this as its motto, and concerns itself with models expressive enough to represent the complexities of higher-order beliefs in a game-theoretic context. Formal logic furnishes a powerful and versatile class of such models; namely, modal logics of belief appropriately specialized for reasoning about games. Rationality is no less important in this setting; however, while it has been incorporated into these models both syntactically and semantically, no axiomatization of the resulting logical systems has been provided. This paper fills this gap.

We take as our point of departure axioms for rationality in the sense of expected utility maximization given in (Bjorndahl, Halpern, & Pass 2011). We extend these axioms to arbitrary decision rules, under the assumption that the players' uncertainty is represented by a probability measure. This allows us to deal with not just expected utility maximization, but other standard rules such as *maximin* and *minimax regret* (see (Halpern 2003) for a discussion of all the decision rules mentioned in this paper). We then go on to consider what happens when the players' uncertainty is represented by a *set* of probabilities, which allows us to capture well-known decision rules such as *maxmin expected utility* and *minimax expected regret*. Finally, we consider situations where a player might be uncertain about which decision rules his opponents are using.

The rest of this paper is organized as follows. In Section 2, we define the core concepts formally: games, modal logics of belief appropriate for reasoning about games, and the incorporation of rationality into these logics. Section 3 gives the axiomatization, and proves that it is sound and complete. In Section 4, we provide sound and complete axiomatizations for the cases where the players' uncertainty is represented by sets of probabilities, and where players may be uncertain about the decision rules used by other players. In Section 5 we discuss the role of language. Section 6 concludes with a discussion of future work. More detailed proofs and further discussion can be found in the full paper, which is available at http://www.math.cornell.edu/∼abjorndahl/Site/CV_files/Axiomatizing%20Rationality.pdf.

2 Reasoning about games

Given a tuple $(X_i)_{i\in I}$ over some finite index set $I = \{1,\dots,n\}$, we adopt the usual notational convention of writing $X \coloneqq \prod_{i\in I} X_i$ and $X_{-i} \coloneqq \prod_{j\neq i} X_j$. We also

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

write $X_i' \times X_{-i}$ for

$$X_1 \times \cdots \times X_{i-1} \times X_i' \times X_{i+1} \times \cdots \times X_n$$

and similarly (x_i', x_{-i}) for $(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n)$.

A **(normal-form) game** is a tuple $\Gamma = (I, (\Sigma_i)_{i \in I}, (u_i)_{i \in I})$ where $I = \{1, \ldots, n\}$ is the set of *players*, Σ_i is the (finite) set of *strategies available to player* i, and $u_i : \Sigma \to \mathbb{R}$ is *player* i*'s utility function*, where $\Sigma = \prod_i \Sigma_i$ denotes the set of *strategy profiles*.

2.1 Syntax

One way of reasoning formally about a game is to build a logical language that is expressive enough to capture the aspects of play that we are interested in analyzing. To this end, given a game Γ, we begin by defining a propositional modal language of belief and then specializing the primitive propositions to correspond to the strategies available to the players.

Given an arbitrary set Φ of primitive propositions, let $\mathcal{L}_B(\Phi)$ be the language recursively generated by the grammar

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid B_i\varphi,$$

where $p \in \Phi$ and $i \in I$. As usual, we read $B_i\varphi$ as "player i believes φ". We define $\vee$, $\rightarrow$, and $\leftrightarrow$ as usual in terms of $\wedge$ and $\neg$, and write $\widehat{B}_i\varphi$ for $\neg B_i \neg\varphi$ ("player i consider it possible that φ"). We also write

$$\begin{aligned} EB^1\varphi &\equiv B_1\varphi \wedge \cdots \wedge B_n\varphi \text{ and} \\ EB^k\varphi &\equiv EB^1(EB^{k-1}\varphi) \end{aligned}$$

for "everyone believes φ" and its k-fold iteration, respectively. Set

$$\Phi_\Gamma \coloneqq \{play_i(\sigma_i) \,:\, i \in I,\, \sigma_i \in \Sigma_i\},$$

where we read $play_i(\sigma_i)$ as "player i is playing strategy σ_i"; we write

$$\begin{aligned} play(\sigma) &\equiv play_1(\sigma_1) \wedge \cdots \wedge play_n(\sigma_n) \text{ and} \\ play_{-i}(\sigma_{-i}) &\equiv \bigwedge_{j \neq i} play_j(\sigma_j) \end{aligned}$$

for "the players are playing according to the strategy profile σ" and the analogous statement regarding the players other than i. The language $\mathcal{L}_B(\Phi_\Gamma)$ is thus appropriate for reasoning about the beliefs of the players with respect to the strategies they are playing.

2.2 Semantics

A language of belief can be interpreted using Kripke-style possible world semantics, where associated to each world ω and each player i is a probability measure on the set of all worlds, used to interpret player i's beliefs at ω. In the case of a language like $\mathcal{L}_B(\Phi_\Gamma)$, we also must take care to interpret the primitive propositions appropriately.

We restrict our attention to finite models. A **finite Γ-structure** is a tuple $M = (\Omega, (Pr_i)_{i \in I}, s)$ satisfying the following conditions:

(C1) Ω is a nonempty, finite set;

(C2) Pr_i associates with each $\omega \in \Omega$ a probability measure $Pr_i(\omega)$ on Ω;

(C3) $Pr_i(\omega)(\{\omega' \in \Omega \,:\, Pr_i(\omega') = Pr_i(\omega)\}) = 1$;

(C4) $s : \Omega \to \Sigma$ satisfies $Pr_i(\omega)(\{\omega' \in \Omega \,:\, s_i(\omega') = s_i(\omega)\}) = 1$.

Conditions (C1) and (C2) set the stage to interpret player i's beliefs at ω by the measure $Pr_i(\omega)$. Condition (C3) then ensures that at each world ω, each player is sure of (i.e. assigns probability 1 to) her own beliefs. Finally, condition (C4) establishes that the *strategy function* s assigns to each world ω a strategy profile $s(\omega)$ in game Γ—intuitively, the strategy that each player is playing at ω—and moreover, that each player is sure of her own strategy.

A finite Γ-structure M induces an **interpretation** $[\![\cdot]\!]_M : \mathcal{L}_B(\Phi_\Gamma) \to 2^\Omega$ defined recursively as follows:

$$\begin{aligned} [\![play_i(\sigma_i)]\!]_M &\coloneqq \{\omega \in \Omega \,:\, s_i(\omega) = \sigma_i\} \\ [\![\varphi \wedge \psi]\!]_M &\coloneqq [\![\varphi]\!]_M \cap [\![\psi]\!]_M \\ [\![\neg\varphi]\!]_M &\coloneqq \Omega \setminus [\![\varphi]\!]_M \\ [\![B_i\varphi]\!]_M &\coloneqq \{\omega \in \Omega \,:\, Pr_i(\omega)([\![\varphi]\!]_M) = 1\}. \end{aligned}$$

Thus, the primitive propositions are interpreted in the obvious way using the strategy function (s_i denotes the ith component function of s), the Boolean connectives are interpreted classically, and the formula $B_i\varphi$ holds at all and only those worlds ω at which $Pr_i(\omega)$ assigns probability 1 to φ. As is standard, we often write $(M, \omega) \models \varphi$ or just $\omega \models \varphi$ to indicate that $\omega \in [\![\varphi]\!]_M$. Similarly, we write $M \models \varphi$ if $[\![\varphi]\!]_M = \Omega$, and when $(M, \omega) \not\models \varphi$, we say that M *refutes* φ *(at* ω*)*, or just that ω refutes φ.

2.3 Rationality

Informally, a player is *rational* if the strategy she is playing is a best response to her beliefs about the outcome of the game, given her preferences. But there is no single conception of what constitutes a "best response"; a variety of principles of decision-making have been proposed and studied.

One influential notion, particularly in game theory, is that of expected utility maximization. Given a game Γ and a probability measure μ on Σ_{-i} (thought of as representing player i's beliefs about the strategies her opponents will play), the **expected utility** of a strategy $\sigma_i \in \Sigma_i$ is just the expected value of the function $u_i(\sigma_i, \cdot) : \Sigma_{-i} \to \mathbb{R}$ with respect to μ:

$$EU_i(\sigma_i; \mu) \coloneqq \sum_{\sigma_{-i} \in \Sigma_{-i}} u_i(\sigma_i, \sigma_{-i}) \cdot \mu(\sigma_{-i}).$$

A best response for an expected utility maximizer is a strategy that maximizes this value. Abstractly, we might identify the mandate "maximize expected utility" for player i (in the game Γ) with a function $\mathfrak{d}_i^{eu}$ that takes as input a belief μ on Σ_{-i} and returns as output the set of strategies $\sigma_i \in \Sigma_i$ that maximize player i's expected utility given μ:

$$\mathfrak{d}_i^{eu}(\mu) \coloneqq \{\sigma_i \in \Sigma_i \,:\, (\forall \sigma_i' \in \Sigma_i)(EU_i(\sigma_i; \mu) \geq EU_i(\sigma_i'; \mu))\}.$$

Let $\Delta(\Sigma_{-i})$ denote the set of all probability measures on Σ_{-i}. Then, generalizing the above, we define a **decision**

rule for player i **(in** Γ**)** to be a function $\mathfrak{d}_i : \Delta(\Sigma_{-i}) \to 2^{\Sigma_i} \setminus \{\emptyset\}$. Intuitively, $\sigma_i \in \mathfrak{d}_i(\mu)$ just in case σ_i is a best response for player i to the belief μ according to the decision rule $\mathfrak{d}_i$. For example, the "maximin" mandate for player i, which says to maximize the worst-case outcome among those considered possible, corresponds to the decision rule defined as follows: let

$$WC_i(\sigma_i; \mu) \coloneqq \min\{u_i(\sigma_i, \sigma_{-i}) \ : \ \mu(\sigma_{-i}) > 0\},$$

and set

$$\mathfrak{d}_i^m(\mu) \coloneqq \{\sigma_i \in \Sigma_i \ : \ (\forall \sigma_i' \in \Sigma_i)(\, WC_i(\sigma_i; \mu) \geq \ WC_i(\sigma_i'; \mu))\}.$$

Decision rules can be interpreted in Γ-structures; roughly speaking, for each world ω, we can define the set of $\mathfrak{d}_i$-best responses for player i at ω, and thus determine whether or not player i is being $\mathfrak{d}_i$-rational (i.e., acting in accordance with the decision rule $\mathfrak{d}_i$) at ω. Formally, given a Γ-structure M, for each player i and each world ω, the probability measure $Pr_i(\omega)$ induces a probability measure $\mu_{i,\omega}$ defined on Σ_{-i} as follows:

$$\mu_{i,\omega}(\sigma_{-i}) \coloneqq Pr_i(\omega)([\![play_{-i}(\sigma_{-i})]\!]_M).$$

In fact, $\mu_{i,\omega}$ is the *pushforward* of $Pr_i(\omega)$ by s_{-i}. Since $Pr_i(\omega)$ is interpreted as representing player i's beliefs at ω, it makes sense to apply her decision rule to $\mu_{i,\omega}$. This leads us to define the set of $\mathfrak{d}_i$**-best responses for player** i **at** ω to be $\mathfrak{d}_i(\mu_{i,\omega})$. We say that player i is $\mathfrak{d}_i$**-rational at** ω just in case $s_i(\omega) \in \mathfrak{d}_i(\mu_{i,\omega})$.

Since we wish to reason formally about games using the logic developed above, and rationality is an important concept for such analyses, we expand the language to include primitive propositions denoting rationality of the players. Given a profile of decision rules $\mathfrak{d} = (\mathfrak{d}_i)_{i \in I}$, let

$$\Phi_\Gamma^{\mathfrak{d}} \coloneqq \Phi_\Gamma \cup \{RAT_i^{\mathfrak{d}_i} \ : \ i \in I\},$$

where $RAT_i^{\mathfrak{d}_i}$ is read "player i is $\mathfrak{d}_i$-rational". Henceforth, except where noted otherwise, we work with an arbitrary but fixed profile of decision rules $\mathfrak{d}$. For notational convenience, we omit the $\mathfrak{d}_i$ when possible, referring to "best responses" and "rationality" instead of "$\mathfrak{d}_i$-best responses" and "$\mathfrak{d}_i$-rationality", and writing RAT_i instead of $RAT_i^{\mathfrak{d}_i}$. We also make use of the syntactic abbreviation

$$RAT \quad \equiv \quad RAT_1 \wedge \cdots \wedge RAT_n$$

for "everyone is rational".

Given a Γ-structure M, the interpretation $[\![\cdot]\!]_M$ is extended to $\mathcal{L}_B(\Phi_\Gamma^{\mathfrak{d}})$ in the obvious way, by setting

$$[\![RAT_i]\!]_M \quad \coloneqq \quad \{\omega \in \Omega \ : \ s_i(\omega) \in \mathfrak{d}_i(\mu_{i,\omega})\}.$$

When $\mathfrak{d} = \mathfrak{d}^{eu}$, so each player's decision rule is given by expected utility maximization, we regain the traditional notion of rationality in game theory. Rationality so defined can be used to characterize several well-known solution concepts in terms of Γ-structures. For example, as shown by Tan and Werlang (1988) and Brandenburger and Dekel (1987), given a game Γ, a strategy σ_i is *rationalizable* if and only if there exists a Γ-structure M and a state ω therein such that $\omega \models play_i(\sigma_i)$ and for every $k \in \mathbb{N}$, $\omega \models EB^k(RAT)$ (i.e. it is common belief that everyone is rational).

It is worth noting that decision rules are general enough to represent processes that fall outside the traditional purview of "rationality". For example, suppose that player i is a computer system and Σ_i is a collection of actions it can execute. Suppose also that the system maintains a database consisting of estimates of the values of certain variables, which can be represented as beliefs about the strategies of "opponents". In this context, a decision rule can be thought of as a nondeterministic process that the computer system might use to choose which action to execute on the basis of the information in its database. In particular, $RAT_i^{\mathfrak{d}_i}$ asserts that the system has executed an action consistent with the nondeterministic process $\mathfrak{d}_i$.[1]

2.4 Example

In order to help solidify the framework just introduced, we present a simple example. Consider the standard Bach-or-Stravinsky game Γ (Osborne & Rubinstein 1994), in which each of two players must choose which of two concerts to attend this evening: one featuring the music of Bach, and one of Stravinsky. Player 1 prefers to attend the Bach concert, while player two prefers the Stravinsky; however, each much prefers to attend the same concert as the other. We can represent these preferences with the utility functions summarized in Table 1.

	Bach	Stravinsky
Bach	3,2	0,0
Stravinsky	0,0	2,3

Table 1: Bach or Stravinsky?

We now describe a Γ-structure in which we can reason about the beliefs and rationality of the players. Let $\Omega = \{\omega_0, \omega_1, \omega_2\}$ and let $Pr_1(\omega_0)$ be the uniform distribution on $\{\omega_1, \omega_2\}$, so player 1 considers ω_1 and ω_2 equally likely. Consider a strategy function s defined such that $s_2(\omega_1) =$ Bach and $s_2(\omega_2) =$ Stravinsky. It is easy to see that in this case, the induced measure μ_{1,ω_0} assigns probability .5 to Bach and .5 to Stravinsky. It follows easily that player 1's expected utility on choosing Bach is 1.5, while his expected utility on choosing Stravinsky is 1. Thus,

$$\omega_0 \models RAT_1^{\mathfrak{d}_1^{eu}} \leftrightarrow play_1(\text{Bach}).$$

On the other hand, the worst-case outcome for player 1 on choosing either Bach or Stravinsky yields a utility of 0, so we have

$$\omega_0 \models RAT_1^{\mathfrak{d}_1^m},$$

regardless of which strategy player 1 actually chooses at ω_0.

[1] In this sense, we can think of a decision rule as a *knowledge-based program* (Fagin *et al.* 1995). Knowledge-based programs have previously been used to characterize rationality and solution concepts (Halpern & Moses 2007).

3 Axiomatization

Although reasoning formally about beliefs and rationality has long been recognized as important to game theory (see, e.g., (**?**)), to the best of our knowledge, rationality has not been axiomatized in a logic of belief. In this section we provide an axiomatization and prove that it is sound and complete.

3.1 The axioms

While most of the axioms are easy to state and understand, two require some preliminary definitions. Fix a game Γ and a profile of decision rules $\mathfrak{d}$. Given a subset $S \subseteq \Sigma_{-i}$, let

$$\delta_{i,S} \equiv \bigwedge_{\sigma_{-i} \in S} \widehat{B}_i \, play_{-i}(\sigma_{-i}) \wedge \bigwedge_{\sigma_{-i} \notin S} \neg \widehat{B}_i \, play_{-i}(\sigma_{-i}).$$

Intuitively, the formula $\delta_{i,S}$ says that player i considers possible *all* and *only* the strategy profiles for his opponents that are elements of S. It is easy to see that for each player i and any world ω, exactly one of the formulas $\delta_{i,S}$ holds.

Given a measure μ on a finite set X, let

$$supp(\mu) \coloneqq \{x \in X \,:\, \mu(x) > 0\},$$

the **support** of μ. For each player i and each $\sigma_i \in \Sigma_i$, let $\mathcal{S}_i^+(\sigma_i)$ denote the collection of all $S \subseteq \Sigma_{-i}$ such that there exists a probability measure μ with $supp(\mu) = S$ and with respect to which σ_i is a best response; that is, $\sigma_i \in \mathfrak{d}_i(\mu)$. Similarly, define $\mathcal{S}_i^-(\sigma_i)$ to be the collection of all $S \subseteq \Sigma_{-i}$ such that there exists a probability measure μ with $supp(\mu) = S$ and $\sigma_i \notin \mathfrak{d}_i(\mu)$.

Consider the following axiom schemes:

G1. $\bigvee_{\sigma_i \in \Sigma_i} play_i(\sigma_i)$

G2. $\neg(play_i(\sigma_i) \wedge play_i(\sigma_i'))$, for $\sigma_i \neq \sigma_i'$

G3. $play_i(\sigma_i) \leftrightarrow B_i \, play_i(\sigma_i)$

G4. $RAT_i \leftrightarrow B_i(RAT_i)$

G5. $(play_i(\sigma_i) \wedge RAT_i) \rightarrow \bigvee_{S \in \mathcal{S}_i^+(\sigma_i)} \delta_{i,S}$

G6. $(play_i(\sigma_i) \wedge \neg RAT_i) \rightarrow \bigvee_{S \in \mathcal{S}_i^-(\sigma_i)} \delta_{i,S}.$

G1–G4 are straightforward. **G1** and **G2** say that, at each state, each player plays exactly one strategy. **G3** and **G4** say that each player is certain of her strategy and of whether or not she is rational. The interesting axioms are **G5** and **G6**. Intuitively, **G5** says that if RAT_i holds and player i is playing σ_i, then player i must consider possible a collection of strategy profiles[2] on which she could place a probability that would justify her playing σ_i. **G6** is interpreted analogously. Notice that player i's actual (i.e. quantitative) beliefs are not fully specified. This may be somewhat surprising, given that the semantic interpretation of RAT_i is defined in terms of the quantitative probabilities that constitute player i's beliefs. Nevertheless, as we show in Section 3.2, these axioms completely characterize the logical properties of rationality as expressible in the language $\mathcal{L}_B(\Phi_\Gamma^{\mathfrak{d}})$; we return to this point in Section 5. Axioms **G1–G4** and the specialization of axioms **G5** and **G6** to the rule of maximizing expected utility appear in (Bjorndahl, Halpern, & Pass 2011).

Let $\mathsf{GL}_\Gamma^{\mathfrak{d}}$ be the axiom system that results from adding **G1–G6** to the standard $\mathsf{KD45}$ axioms and rules of inference of belief logic (see, e.g., (Fagin *et al.* 1995)). It bears emphasizing that $\mathsf{GL}_\Gamma^{\mathfrak{d}}$ is parametrized by two variables: the underlying game Γ that determines the players, their strategies, and their preferences, and the profile of decision rules $\mathfrak{d}$ that determines the meaning of "rationality" for each player. Thus, what we are axiomatizing here is not a single logic but a class of logics.

For each fixed Γ and $\mathfrak{d}$, the corresponding axiom system $\mathsf{GL}_\Gamma^{\mathfrak{d}}$ is trivially decidable (i.e., we can effectively determine whether a formula is an instance of an axiom scheme) because $\mathsf{KD45}$ is decidable and there are only finitely-many instances of the axiom schemes **G1–G6** (the implicit quantification in these schemes ranges in some cases over players and in others over strategies, all of which are finite sets).

But there is arguably a more interesting question. Up to now we have fixed a game Γ, and considered decision rules in the game Γ. But rules like maximizing expected utility, maximin, or minimizing regret can be applied in all games in a uniform way. To capture this, we define a **decision paradigm** to be a function that maps each game Γ to a decision rule in Γ. Suppose that we are given decision paradigms $\mathcal{D}_i$ for each player associating with each game Γ a decision rule $\mathcal{D}_i(\Gamma)$ for that player in Γ. We might want to know, given the profile $\mathcal{D} = (\mathcal{D}_i)_{i \in I}$, whether the mapping

$$\Gamma \mapsto \mathsf{GL}_\Gamma^{\mathcal{D}(\Gamma)}$$

is decidable; in other words, given as input a game Γ, can we effectively determine whether a formula belongs to the axiom system $\mathsf{GL}_\Gamma^{\mathcal{D}(\Gamma)}$? For each game Γ, this requires determining membership in the sets $\mathcal{S}_i^+(\sigma_i)$ and $\mathcal{S}_i^-(\sigma_i)$, which are defined by existential quantification over simplices $\Delta(\Sigma_{-i})$, subject to constraints based on the decision rules $\mathcal{D}_i(\Gamma)$. In the case of familiar decision paradigms like maximin or expected utility maximization, computing the sets $\mathcal{S}_i^+(\sigma_i)$ and $\mathcal{S}_i^-(\sigma_i)$ boils down to solving systems of linear inequalities. In general, however, we must impose certain computability requirements on the decision paradigms in order to be able to decide whether a formula is an instance of an axiom. At this point, we have not found an interesting characterization of when we get decidability.

3.2 Soundness and completeness

Theorem 3.1: $\mathsf{GL}_\Gamma^{\mathfrak{d}}$ *is a sound axiomatization of the language* $\mathcal{L}_B(\Phi_\Gamma^{\mathfrak{d}})$ *with respect to the class of all finite* Γ*-structures.*

[2]Note that the sets $\mathcal{S}_i^+(\sigma_i)$ and $\mathcal{S}_i^-(\sigma_i)$ could be empty; by convention, we define the empty disjunction to be $\bot$ (or *false*). If, for example, $\mathcal{S}_i^+(\sigma_i) = \emptyset$, this means that σ_i is not a best response to any beliefs, in which case the corresponding axiom $(play_i(\sigma_i) \wedge RAT_i) \rightarrow \bot$ is intuitively correct.

Proof: The proof is much in the same spirit as the proof of the corresponding result in (Bjorndahl, Halpern, & Pass 2011) (except that now we consider rules more general than just maximizing expected utility). Soundness of the axioms and rules of KD45 can be proved as usual. It therefore suffices to show that **G1**–**G6** are valid in all finite Γ-structures.

Fix a finite Γ-structure $M = (\Omega, (Pr_i)_{i\in I}, s)$. Soundness of $\mathbf{G1}$ and $\mathbf{G2}$ is an immediate consequence of the fact that s is a (total) function. Soundness of $\mathbf{G3}$ is a straightforward consequence of condition (C4), while soundness of $\mathbf{G4}$ follows easily from the combination of conditions (C3) and (C4).

Now suppose that $\omega \models play_i(\sigma_i) \wedge RAT_i$; then $\sigma_i \in \mathfrak{d}_i(\mu_{i,\omega})$. Set $S := supp(\mu_{i,\omega})$, and observe that $S \in \mathcal{S}_i^+(\sigma_i)$. For each $\sigma_{-i} \in S$, it is easy to see that $\omega \models \widehat{B}_i\, play_{-i}(\sigma_{-i})$; therefore, we must have

$$\omega \models \bigwedge_{\sigma_{-i}\in S} \widehat{B}_i\, play_{-i}(\sigma_{-i}).$$

Similarly, for each $\sigma_{-i} \notin S$, we have $\omega \models \neg\widehat{B}_i\, play_{-i}(\sigma_{-i})$, so

$$\omega \models \bigwedge_{\sigma_{-i}\notin S} \neg\widehat{B}_i\, play_{-i}(\sigma_{-i}).$$

In other words, $\omega \models \delta_{i,S}$; this establishes the soundness of $\mathbf{G5}$.

Finally, suppose that $\omega \models play_i(\sigma_i) \wedge \neg RAT_i$, which implies that $\sigma_i \notin \mathfrak{d}_i(\mu_{i,\omega})$. Set $S := supp(\mu_{i,\omega})$; then $S \in \mathcal{S}_i^-(\sigma_i)$ and, as above, we have $\omega \models \delta_{i,S}$, which establishes soundness of $\mathbf{G6}$. ∎

We prove completeness by what is essentially the *canonical model method*, a standard method for proving completeness of modal logics (see, e.g., (Fagin *et al.* 1995) or any standard text on modal logic). Of course, the full canonical model is not finite (for $n > 1$), so we modify the construction by restricting attention to finite sub-languages. More precisely, given a formula $\varphi \in \mathcal{L}_B(\Phi_\Gamma^{\mathfrak{d}})$, we identify a finite sub-language of $\mathcal{L}_B(\Phi_\Gamma^{\mathfrak{d}})$ such that the corresponding canonical model refutes φ just in case $\mathsf{GL}_\Gamma^{\mathfrak{d}} \nvdash \varphi$. This technique is sometimes called *filtration*.

Fix a formula $\varphi \in \mathcal{L}_B(\Phi_\Gamma^{\mathfrak{d}})$. Let $Sub_\Gamma(\varphi)$ denote the collection of all subformulas of φ together with all subformulas of instances of the axiom schemes $\mathbf{G1}$ through $\mathbf{G6}$. Define

$$Sub_\Gamma^+(\varphi) := Sub_\Gamma(\varphi) \cup \{\neg\psi \,:\, \psi \in Sub_\Gamma(\varphi)\}.$$

Note that there are only finitely many instances of $\mathbf{G1}$ through $\mathbf{G6}$, and therefore $Sub_\Gamma^+(\varphi)$ is finite.

Let Ω^φ be the collection of all maximal, consistent (with respect to $\mathsf{GL}_\Gamma^{\mathfrak{d}}$) subsets of $Sub_\Gamma^+(\varphi)$. Clearly Ω^φ is a finite set. Given $X \subseteq \mathcal{L}_B(\Phi_\Gamma^{\mathfrak{d}})$, set

$$X^{B_i} := \{\psi \,:\, B_i\psi \in X\};$$

and, for $F \in \Omega^\varphi$, define

$$\mathrm{Bel}_i(F) := \{G \in \Omega^\varphi \,:\, G \supseteq F^{B_i} \text{ and } G^{B_i} = F^{B_i}\}.$$

For each $i \in I$ and each $F \in \Omega^\varphi$, we will define a probability measure $Pr_i^\varphi(F)$ on Ω^φ such that the support of this measure is precisely $\mathrm{Bel}_i(F)$. Loosely speaking, $\mathrm{Bel}_i(F)$ is the set of all $G \in \Omega^\varphi$ that are compatible with the beliefs of player i in F. More precisely, $G \in \mathrm{Bel}_i(F)$ if and only if:

(a) $B_i\psi \in F$ implies $\psi \in G$, and

(b) $B_i\psi \in F$ iff $B_i\psi \in G$.

Condition (a) just says that everything player i believes in F is true in G; condition (b) says that player i's beliefs in F are the same as her beliefs in G, which is reasonable in light of the fact that we are working in a system with positive and negative introspection. (In the full canonical model, (b) follows from (a); here we must impose this condition explicitly because of the way the language has been restricted.)

Since our aim is to define probability measures with the sets $\mathrm{Bel}_i(F)$ as their supports, we must show that these sets are never empty.

Lemma 3.2: *For each $i \in I$ and $F \in \Omega^\varphi$, $\mathrm{Bel}_i(F) \neq \emptyset$.*

Proof: Given $F \in \Omega^\varphi$, set

$$\begin{aligned}\Lambda \;:=\; & \{\psi \,:\, B_i\psi \in F\} \cup \{B_i\psi \,:\, B_i\psi \in F\} \\ & \cup \{\neg B_i\psi \,:\, \neg B_i\psi \in F\}.\end{aligned}$$

It is easy to see that $\Lambda \subset Sub_\Gamma^+(\varphi)$. In addition, we show that Λ is consistent. For suppose not; then

$$\begin{aligned}\mathsf{GL}_\Gamma^{\mathfrak{d}} \vdash \neg\bigwedge_{\xi\in\Lambda}\xi \;&\Longrightarrow\; \mathsf{GL}_\Gamma^{\mathfrak{d}} \vdash B_i\neg\bigwedge_{\xi\in\Lambda}\xi \\ &\Longrightarrow\; \mathsf{GL}_\Gamma^{\mathfrak{d}} \vdash \neg B_i\bigwedge_{\xi\in\Lambda}\xi \\ &\Longrightarrow\; \mathsf{GL}_\Gamma^{\mathfrak{d}} \vdash \neg\bigwedge_{\xi\in\Lambda} B_i\xi,\end{aligned}$$

which is a contradiction, since each formula $B_i\xi$ with $\xi \in \Lambda$ is logically equivalent to a formula in F, and F is consistent.

From this we can conclude that there exists a $G \in \Omega^\varphi$ such that $G \supseteq \Lambda$. It follows immediately that $G \supseteq F^{B_i}$ and that $G^{B_i} \supseteq F^{B_i}$. Moreover, if $\psi \in G^{B_i}$, then $B_i\psi \in G$ and so certainly $\neg B_i\psi \notin G$, from which it follows that $\neg B_i\psi \notin \Lambda$ and thus $\neg B_i\psi \notin F$. Maximality of F then guarantees that $B_i\psi \in F$, whence $\psi \in F^{B_i}$, and so $G^{B_i} \subseteq F^{B_i}$. This establishes that $G \in \mathrm{Bel}_i(F)$, as desired. ∎

In the classical canonical model construction, it is sufficient to define $Pr_i^\varphi(F)$ to be the uniform distribution on $\mathrm{Bel}_i(F)$. In the present context, however, we need to be more careful, since $Pr_i^\varphi(F)$ is used not only to interpret the belief modalities B_i, but also the primitive propositions RAT_i. In essence, we must define $Pr_i^\varphi(F)$ in a manner that agrees with whether or not player i is best responding to her beliefs at F; not surprisingly, this is precisely where the axiom schemes **G5** and **G6** come into play. At the same time, we have to define Pr_i^φ on Ω^φ in a systematic way so as to preserve the introspection condition (C3). What follows is a formalization of this basic recipe, for which several more lemmas and definitions are needed.

Lemma 3.3: *Let $F \in \Omega^\varphi$. If $\widehat{B}_i\psi \in F$, then there exists a $G \in \mathrm{Bel}_i(F)$ with $\psi \in G$; if $\neg\widehat{B}_i\psi \in F$, then for all $G \in \mathrm{Bel}_i(F)$ we have $\psi \notin G$.*

Proof: See the full paper. ∎

For each $\sigma_{-i} \in \Sigma_{-i}$, define

$$\mathrm{Bel}_i(F;\sigma_{-i}) \coloneqq \{G \in \mathrm{Bel}_i(F) \,:\, play_{-i}(\sigma_{-i}) \in G\}.$$

Given $F \in \Omega^\varphi$, it is easy to see, using **G1** and **G2**, that there is a unique $\sigma_i \in \Sigma_i$ with $play_i(\sigma_i) \in F$. If, in addition, $RAT_i \in F$, then by **G5** we know that for some $S \in \mathcal{S}_i^+(\sigma_i)$, $\delta_{i,S} \in F$ (or, in the case where $\mathcal{S}_i^+(\sigma_i) = \emptyset$, we know that no such F exists). Otherwise, if $RAT_i \notin F$, then by **G6** we know that for some $S \in \mathcal{S}_i^-(\sigma_i)$, $\delta_{i,S} \in F$ (or again, in the case where $\mathcal{S}_i^-(\sigma_i) = \emptyset$, that no such F exists). Thus, for each $i \in I$, there is a unique set $S_i(F) \subseteq \Sigma_{-i}$ such that $\delta_{i,S_i(F)} \in F$, and moreover, $S_i(F) \in \mathcal{S}_i^+(\sigma_i)$ if $RAT_i \in F$, and $S_i(F) \in \mathcal{S}_i^-(\sigma_i)$ if $RAT_i \notin F$.

Lemma 3.4: *The collection* $\{\mathrm{Bel}_i(F;\sigma_{-i}) \,:\, \sigma_{-i} \in \Sigma_{-i}\}$ *partitions* $\mathrm{Bel}_i(F)$*; moreover,* $\mathrm{Bel}_i(F;\sigma_{-i}) \neq \emptyset$ *if and only if* $\sigma_{-i} \in S_i(F)$.

Proof: The first statement is a straightforward consequence of **G1** and **G2**, while the second is an immediate corollary of Lemma 3.3 together with the fact that

$$\begin{aligned}\delta_{i,S_i(F)} \;\equiv\; & \textstyle\bigwedge_{\sigma_{-i} \in S_i(F)} \widehat{B}_i\, play_{-i}(\sigma_{-i}) \wedge \\ & \textstyle\bigwedge_{\sigma_{-i} \notin S_i(F)} \neg \widehat{B}_i\, play_{-i}(\sigma_{-i}) \in F. \;\blacksquare\end{aligned}$$

For each $\sigma_i \in \Sigma_i$ and $S \in \mathcal{S}_i^+(\sigma_i)$, let $\mu^+_{\sigma_i,S}$ be a fixed probability measure witnessing the fact that $S \in \mathcal{S}_i^+(\sigma_i)$; that is, $supp(\mu^+_{\sigma_i,S}) = S$ and $\sigma_i \in \mathfrak{d}_i(\mu^+_{\sigma_i,S})$. Likewise, for each $\sigma_i \in \Sigma_i$ and $S \in \mathcal{S}_i^-(\sigma_i)$, let $\mu^-_{\sigma_i,S}$ be a fixed probability measure witnessing the fact that $S \in \mathcal{S}_i^-(\sigma_i)$.

Let $F \in \Omega^\varphi$, and suppose that $play_i(\sigma_i) \in F$. In light of Lemma 3.4, we can define $Pr_i^\varphi(F)$ to be the unique probability measure on $\mathrm{Bel}_i(F)$ such that, for all $\sigma_{-i} \in \Sigma_{-i}$,

$$Pr_i^\varphi(F)(\mathrm{Bel}_i(F;\sigma_{-i})) = \begin{cases} \mu^+_{\sigma_i,S_i(F)}(\sigma_{-i}) & \text{if } RAT_i \in F \\ \mu^-_{\sigma_i,S_i(F)}(\sigma_{-i}) & \text{if } RAT_i \notin F, \end{cases}$$

and which is uniform within each (nonempty) set $\mathrm{Bel}_i(F;\sigma_{-i})$.

Proposition 3.5: *Pr_i^φ satisfies the following:*

(a) $Pr_i^\varphi(F)(G) > 0$ iff $G \in \mathrm{Bel}_i(F)$, and

(b) $G^{B_i} = F^{B_i}$ implies $Pr_i^\varphi(G) = Pr_i^\varphi(F)$.

Proof: See the full paper. ∎

Finally, we define a strategy function $s^\varphi : \Omega^\varphi \to \Sigma$ by assigning to each $F \in \Omega^\varphi$ the unique strategy profile $\sigma \in \Sigma$ such that $play(\sigma) \in F$.

Lemma 3.6: *The tuple $M^\varphi \coloneqq (\Omega^\varphi, (Pr_i^\varphi)_{i\in I}, s^\varphi)$ is a Γ-structure.*

Proof: Conditions (C1) and (C2) have already been established. By Lemma 3.5(b), in order to see that (C3) holds it suffices to observe that $Pr_i^\varphi(F)(G) > 0$ implies that $G^{B_i} = F^{B_i}$, which follows from Lemma 3.5(a). This same observation also establishes (C4), since by **G3** we know that $G^{B_i} = F^{B_i}$ implies $s_i^\varphi(G) = s_i^\varphi(F)$. ∎

Lemma 3.7: *For all formulas ψ, for all $F \in \Omega^\varphi$, if $\psi \in Sub_\Gamma^+(\varphi)$ then $F \in [\![\psi]\!]_{M^\varphi}$ if and only if $\psi \in F$.*

Proof: The proof proceeds by induction on the structure of ψ. We prove here the base cases corresponding to the primitive propositions; the inductive steps can be proved in the standard way (see, e.g., (Fagin *et al.* 1995)).

First consider the primitive proposition $play_i(\sigma_i)$. We have

$$\begin{aligned} F \in [\![play_i(\sigma_i)]\!]_{M^\varphi} \quad & \text{iff} \quad s_i^\varphi(F) = \sigma_i \\ & \text{iff} \quad play_i(\sigma_i) \in F, \end{aligned}$$

as a direct consequence of the definition of s^φ. Next consider the primitive proposition RAT_i; we have

$$\begin{aligned} F \in [\![RAT_i]\!]_{M^\varphi} \quad & \text{iff} \quad s_i^\varphi(F) \in \mathfrak{d}_i(\mu_{i,F}) \\ & \text{iff} \quad RAT_i \in F, \end{aligned}$$

the last equivalence being a consequence of the definition of Pr_i^φ, which ensures that $s_i^\varphi(F)$ is a best response to (the pushforward of) $Pr_i^\varphi(F)$ precisely when $RAT_i \in F$. This completes the proof. ∎

Theorem 3.8: $\mathsf{GL}_\Gamma^\mathfrak{d}$ *is a complete axiomatization of the language $\mathcal{L}_B(\Phi_\Gamma^\mathfrak{d})$ with respect to the class of all finite Γ-structures.*

Proof: Suppose that $\mathsf{GL}_\Gamma^\mathfrak{d} \nvdash \varphi$. Then $\{\neg\varphi\}$ is consistent and so can be extended to a maximal consistent set $F \in \Omega^\varphi$. By Lemma 3.7, this implies that $F \notin [\![\varphi]\!]_{M^\varphi}$ and so, in particular, that $M^\varphi \not\models \varphi$, as desired. ∎

4 Extensions

4.1 Belief as lower probability

In this paper we have taken for granted that each player's uncertainty is represented by a probability measure. While this is a very standard assumption, it is by no means the only framework that has been considered; see (Halpern 2003) for an overview of different ways of modeling uncertainty. Here we show that, with very minor modifications, the results of Section 3.2 hold also in the more general context where beliefs are represented using *sets* of probability measures.

Given a set $\mathcal{P}$ of probability measures, the **lower probability** of an event E, denoted $\mathcal{P}_*(E)$, is defined to be the infimum of the probabilities assigned to E by members of $\mathcal{P}$:

$$\mathcal{P}_*(E) \coloneqq \inf\{\mu(E) \,:\, \mu \in \mathcal{P}\}.$$

Fix a game Γ. A **finite lower Γ-structure** is a tuple $M = (\Omega, (\mathcal{PR}_i)_{i\in I}, s)$ satisfying the following conditions:

(L1) Ω is a nonempty, finite set;

(L2) $\mathcal{PR}_i$ associates to each $\omega \in \Omega$ a set $\mathcal{PR}_i(\omega)$ of probability measures on Ω;

(L3) $\mathcal{PR}_i(\omega)_*(\{\omega' \in \Omega \,:\, \mathcal{PR}_i(\omega') = \mathcal{PR}_i(\omega)\}) = 1$;

(L4) $s : \Omega \to \Sigma$ satisfies $\mathcal{PR}_i(\omega)_*(\{\omega' \in \Omega \,:\, s_i(\omega') = s_i(\omega)\}) = 1$.

These conditions are simply the analogues of conditions (C1) through (C4) where uncertainty is represented by sets of probability measures and certainty is identified with lower probability 1. Accordingly, we define the interpretation $[\![\cdot]\!]_M$ as before, except for the clause corresponding to the belief modalities, which is replaced by the following:

$$[\![B_i\varphi]\!]_M \quad := \quad \{\omega \in \Omega \,:\, \mathcal{PR}_i(\omega)_*([\![\varphi]\!]_M) = 1\}.$$

Finally, a **decision rule** for player i in this context is a function $\mathfrak{d}_i : 2^{\Delta(\Sigma_{-i})} \to 2^{\Sigma_i}$, since player i must make her choice based on the uncertainty given by a set of probability measures. For example, the "maximin expected utility" decision rule for player i would be given by the following:

$$\mathfrak{d}_i^{meu}(\mathcal{P}) := \big\{\sigma_i \in \Sigma_i \,:\, (\forall \sigma_i' \in \Sigma_i) \\ (\min_{\mu\in\mathcal{P}}\{EU_i(\sigma_i;\mu)\} \geq \min_{\mu\in\mathcal{P}}\{EU_i(\sigma_i';\mu)\})\big\}.$$

Other rules, such as minimax expected regret (Hayashi 2008), can also easily be defined in this setting.

As before, such decision rules makes sense in a Γ-structure M: for each player i and each world ω, the probability measures in the set $\mathcal{PR}_i(\omega)$ can be pushed forward by s_{-i} to probability measures on Σ_{-i}. Let $\mathcal{P}_{i,\omega}$ denote the set of all such pushforwards:

$$\mathcal{P}_{i,\omega} := \{\mu_{i,\omega} \,:\, \mu \in \mathcal{PR}_i(\omega)\}.$$

Then we can define $\mathfrak{d}_i$**-rationality** for player i at ω in the obvious way, namely, by the requirement that $s_i(\omega) \in \mathfrak{d}_i(\mathcal{P}_{i,\omega})$.

The axiomatization of Section 3 can be generalized as well. First observe that the dual belief modality, $\widehat{B}_i \equiv \neg B_i \neg$, is interpreted as positive *upper* probability, where the **upper probability** of an event E with respect to a set $\mathcal{P}$ of probability measures, denoted $\mathcal{P}^*(E)$, is given by

$$\mathcal{P}^*(E) := \sup\{\mu(E) \,:\, \mu \in \mathcal{P}\}.$$

Accordingly, given a set $\mathcal{P}$ of probability measures on a finite space X, we define

$$supp(\mathcal{P}) := \{x \in X \,:\, (\exists \mu \in \mathcal{P})(\mu(x) > 0)\}.$$

For each $\sigma_i \in \Sigma_i$, let $\mathcal{S}_i^+(\sigma_i)$ denote the collection of all $S \subseteq \Sigma_{-i}$ such that there exists a set of probability measures $\mathcal{P}$ with $supp(\mathcal{P}) = S$ and $\sigma_i \in \mathfrak{d}_i(\mathcal{P})$. Similarly, define $\mathcal{S}_i^-(\sigma_i)$ to be the collection of all $S \subseteq \Sigma_{-i}$ such that there exists a set $\mathcal{P}$ of probability measures with $supp(\mathcal{P}) = S$ and $\sigma_i \notin \mathfrak{d}_i(\mathcal{P})$. It is not hard to see that all of these definitions generalize what was presented in Sections 2 and 3; indeed, by considering the special case where all sets of probability measures are singletons, we recover that framework exactly. Moreover, the axiom system $\mathsf{GL}_\Gamma^{\mathfrak{d}}$, interpreted in this more general setting using the definitions above, remains sound and complete.

Theorem 4.1: $\mathsf{GL}_\Gamma^{\mathfrak{d}}$ *is a sound and complete axiomatization of the language* $\mathcal{L}_B(\Phi_\Gamma^{\mathfrak{d}})$ *with respect to the class of all finite lower* Γ*-structures.*

Proof: The proof given in Section 3.2 works here as well, modulo the obvious minor alterations in keeping with the generalized definitions given above. In particular, for each $\sigma_i \in \Sigma_i$ and $S \in \mathcal{S}_i^+(\sigma_i)$, we define $\mathcal{P}^+_{\sigma_i,S}$ to be a fixed *set* of probability measures such that $supp(\mathcal{P}^+_{\sigma_i,S}) = S$ and $\sigma_i \in \mathfrak{d}_i(\mathcal{P}^+_{\sigma_i,S})$; likewise, for each $\sigma_i \in \Sigma_i$ and $S \in \mathcal{S}_i^-(\sigma_i)$, define $\mathcal{P}^-_{\sigma_i,S}$ to be a fixed set of probability measures witnessing the fact that $S \in \mathcal{S}_i^-(\sigma_i)$. Then, given $F \in \Omega^\varphi$ with $play_i(\sigma_i) \in F$, define $\mathcal{PR}_i^\varphi(F)$ as follows: for each $\mu \in \mathcal{P}^+_{\sigma_i,S} \cup \mathcal{P}^-_{\sigma_i,S}$, let $\tilde{\mu}$ be the unique probability measure on $\mathrm{Bel}_i(F)$ such that, for all $\sigma_{-i} \in \Sigma_{-i}$,

$$\tilde{\mu}(\mathrm{Bel}_i(F;\sigma_{-i})) = \mu(\sigma_{-i}),$$

and which is uniform on each set $\mathrm{Bel}_i(F;\sigma_{-i})$; then set

$$\mathcal{PR}_i^\varphi(F) := \begin{cases} \{\tilde{\mu} \,:\, \mu \in \mathcal{P}^+_{\sigma_i,S}\} & \text{if } RAT_i \in F \\ \{\tilde{\mu} \,:\, \mu \in \mathcal{P}^-_{\sigma_i,S}\} & \text{if } RAT_i \notin F. \end{cases} \blacksquare$$

4.2 Multiple decision rules

In general, we may wish to reason about players who are uncertain about which decision rules their opponents are using. For example, player i might believe that if player j is maximizing her expected utility, she will play σ_j, but if she plays σ_j', then she might be minimizing the worst-case outcome. One way to try to model such uncertainty is to expand the logic so that there is a *set* $\mathfrak{D}_i$ of decision rules associated with each player i. Consider the collection of primitive propositions

$$\Phi_\Gamma^{\mathfrak{D}} := \Phi_\Gamma \cup \{RAT_i^{\mathfrak{d}_i} \,:\, i \in I,\, \mathfrak{d}_i \in \mathfrak{D}_i\}.$$

Interpret $RAT_i^{\mathfrak{d}_i}$ as before. Then $\mathcal{L}_B(\Phi_\Gamma^{\mathfrak{D}})$ is a language for reasoning about the strategies and beliefs of the players $i \in I$ as well as their adherence to the various decision rules $\mathfrak{d}_i \in \mathfrak{D}_i$.

To obtain a complete axiomatization of this language, it is not sufficient to simply let **G4**–**G6** range over all decision rules $\mathfrak{d}_i \in \mathfrak{D}_i$ for each player i; in general, this system is sound but not complete. Roughly speaking, this is because it is possible for both $play_i(\sigma_i) \wedge RAT_i^{\mathfrak{d}_i}$ and $play_i(\sigma_i) \wedge RAT_i^{\mathfrak{d}_i'}$ to be consistent with $\delta_{i,S}$ for some $S \subseteq \Sigma_{-i}$, yet no measure μ with $supp(\mu) = S$ is such that $\sigma_i \in \mathfrak{d}_i(\mu) \cap \mathfrak{d}_i'(\mu)$. In this case, the formula

$$play_i(\sigma_i) \wedge RAT_i^{\mathfrak{d}_i} \wedge RAT_i^{\mathfrak{d}_i'} \wedge \delta_{i,S}$$

is not satisfiable, but there is no way to prove its negation from the axioms. However, provided that each set $\mathfrak{D}_i$ is finite, we can deal with this problem by replacing **G5** and **G6** with the following collection of axioms for each player i, each strategy $\sigma_i \in \Sigma_i$, and every subset $D \subseteq \mathfrak{D}_i$:

$$\Big(play_i(\sigma_i) \wedge \bigwedge_{\mathfrak{d}_i \in D} RAT_i^{\mathfrak{d}_i} \wedge \bigwedge_{\mathfrak{d}_i \notin D} \neg RAT_i^{\mathfrak{d}_i}\Big) \to \bigvee_{S \in \mathcal{S}_i^D(\sigma_i)} \delta_{i,S},$$

where $\mathcal{S}_i^D(\sigma_i)$ is the collection of all $S \subseteq \Sigma_{-i}$ such that there exists a probability measure μ on S such that $supp(\mu) = S$ and for every $\mathfrak{d}_i \in D$, $\sigma_i \in \mathfrak{d}_i(\mu)$, and for every $\mathfrak{d}_i \notin D$, $\sigma_i \notin \mathfrak{d}_i(\mu)$. **G5** and **G6** are special cases occurring when $|\mathfrak{D}_i| = 1$, corresponding to $D = \mathfrak{D}_i$ and $D = \emptyset$, respectively.

These modified axioms give a sound and complete axiomatization of $\mathcal{L}_B(\Phi_\Gamma^{\mathfrak{D}})$ with respect to the class of all finite Γ-structures. Nevertheless, there is something unsatisfying about using this language to model players' uncertainty about decision rules: the propositions $RAT_i^{\mathfrak{d}_i}$ say that player i is playing a $\mathfrak{d}_i$-best-response, but *not* that player i is actually *using* the rule $\mathfrak{d}_i$ to decide her strategy. To see the difference, consider a player i who is trying to maximize her expected utility (i.e. using $\mathfrak{d}_i^{eu}$), and happens to also play a maximin strategy; contrast this with a scenario in which she is actively seeking to maximize the worst-case outcome (i.e. using $\mathfrak{d}_i^{m}$), and in so doing happens to play a strategy that maximizes her expected utility. Although player i is following different decision rules in these two cases, the language $\mathcal{L}_B(\Phi_\Gamma^{\mathfrak{D}})$ cannot express this difference; the formula $RAT_i^{\mathfrak{d}_i^{eu}} \wedge RAT_i^{\mathfrak{d}_i^{m}}$ holds either way. For instance, in the example presented in Section 2.4, it is not hard to check that

$$\omega_0 \models play_1(\text{Bach}) \rightarrow \left(RAT_1^{\mathfrak{d}_1^{eu}} \wedge RAT_1^{\mathfrak{d}_1^{m}}\right).$$

In sum, the propositions $RAT_i^{\mathfrak{d}_i}$ do not say anything about *how* player i is making her decision, but simply record whether or not the decision she does make is compatible with the rule $\mathfrak{d}_i$. What we want is a different kind of proposition, say $rule_i(\mathfrak{d}_i)$, that says that player i really is *using* the rule $\mathfrak{d}_i$ in deciding her strategy.

Decision rules interpreted in this sense are particularly relevant in a dynamic setting. When an opponent does something unexpected and seemingly irrational, there is the question of how to update your beliefs. One option is to abandon the belief that your opponent is rational, but this is unsatisfying both conceptually and methodologically. An alternative response is to update your beliefs about your opponent's beliefs: what they did actually was rational with respect to their beliefs, you had just misjudged what those beliefs were (see, e.g., (Battigalli & Siniscalchi 2002)). But in some cases, this too is unsatisfying or simply unavailable. However, when decision rules are formally present as objects of belief, a third option becomes available: abandon the belief that your opponent is $\mathfrak{d}_i$-rational, but not that they are behaving rationally with respect to *some* decision rule. Though an analysis of decision rules in extensive-form games is beyond the scope of this paper, the groundwork for such a study can be laid by formalizing them in a static context.

Fix a game Γ and a profile $\mathfrak{D} = (\mathfrak{D}_i)_{i\in I}$ of sets of decision rules for each player $i \in I$. Let

$$\Phi_{\Gamma,\mathfrak{D}} \coloneqq \Phi_\Gamma \cup \{rule_i(\mathfrak{d}_i) \,:\, i \in I,\, \mathfrak{d}_i \in \mathfrak{D}_i\}.$$

In order to interpret the primitive propositions $rule_i(\mathfrak{d}_i)$, we must extend the semantic model so that it associates with each world ω the decision rule that each player i is using at that world; furthermore, we must constrain the strategies used at each world so that they are compatible with the corresponding decision rules. Formally, a **Γ-structure with decision functions** is a tuple $M = (\Omega, (Pr_i)_{i\in I}, s, r)$ satisfying (C1) through (C4) as well as the following additional conditions:

(C5) $r : \Omega \rightarrow \mathfrak{D}$ satisfies $Pr_i(\omega)(\{\omega' \in \Omega \,:\, r_i(\omega') = r_i(\omega)\}) = 1$;

(C6) $s_i(\omega) \in r_i(\omega)(\mu_{i,\omega})$.

Condition (C5) says that the *decision function* r assigns to each world ω a profile of decision rules $r(\omega)$—intuitively, $r_i(\omega) \in \mathfrak{D}_i$ is the rule that player i is using at ω—and moreover, each player is sure of her own decision rule. Condition (C6) requires that, at each world ω, the strategy $s_i(\omega)$ is an $r_i(\omega)$-best response for player i; in other words, player i really is following the decision rule $r_i(\omega)$ at ω.

The language $\mathcal{L}_B(\Phi_{\Gamma,\mathfrak{D}})$ can be interpreted in a Γ-structure with decision functions M as before, with the additional clause

$$[\![rule_i(\mathfrak{d}_i)]\!]_M \;\coloneqq\; \{\omega \in \Omega \,:\, r_i(\omega) = \mathfrak{d}_i\}.$$

The resulting logic can be axiomatized using essentially the same technique as in Section 3. Consider the following axiom schemes:

P1. $\bigvee_{\sigma_i\in\Sigma_i} play_i(\sigma_i)$

P2. $\neg(play_i(\sigma_i) \wedge play_i(\sigma_i'))$, for $\sigma_i \neq \sigma_i'$

P3. $play_i(\sigma_i) \leftrightarrow B_i\, play_i(\sigma_i)$

P4. $\bigvee_{\mathfrak{d}_i\in\mathfrak{D}_i} rule_i(\mathfrak{d}_i)$

P5. $\neg(rule_i(\mathfrak{d}_i) \wedge rule_i(\mathfrak{d}_i'))$, for $\mathfrak{d}_i \neq \mathfrak{d}_i'$

P6. $rule_i(\mathfrak{d}_i) \leftrightarrow B_i\, rule_i(\mathfrak{d}_i)$

P7. $(play_i(\sigma_i) \wedge rule_i(\mathfrak{d}_i)) \rightarrow \bigvee_{S\in\mathcal{S}_i^+(\sigma_i)} \delta_{i,S}$.

Here, as before, $\mathcal{S}_i^+(\sigma_i)$ denotes the collection of all $S \subseteq \Sigma_{-i}$ such that there exists a probability measure μ with $supp(\mu) = S$ and such that $\sigma_i \in \mathfrak{d}_i(\mu)$. Note that there is no need for a symmetric axiom involving $\mathcal{S}_i^-(\sigma_i)$ for this logic, because the formula $\neg rule_i(\mathfrak{d}_i)$, unlike $\neg RAT_i$, does not say that σ_i is *incompatible* with player i's beliefs and the decision rule $\mathfrak{d}_i$; it simply says that player i did not *use* the rule $\mathfrak{d}_i$ to help choose her strategy σ_i (though she may, coincidentally, have beliefs with respect to which σ_i is a $\mathfrak{d}_i$-best response).

Let $\mathsf{GL}_{\Gamma,\mathfrak{D}}$ be the axiom system that results from adding **P1**–**P7** to the KD45 axioms and rules of inference of belief logic. Then we have the following result, the proof of which proceeds analogously to that given in Section 3.2.

Theorem 4.2: *$\mathsf{GL}_{\Gamma,\mathfrak{D}}$ is a sound and complete axiomatization of the language $\mathcal{L}_B(\Phi_{\Gamma,\mathfrak{D}})$ with respect to the class of all finite Γ-structures with decision functions.*

5 The role of language

In this section we focus on the profile of decision rules $\mathfrak{d} = \mathfrak{d}^{eu}$, with respect to which each player is rational precisely if they are playing a strategy that maximizes their expected utility. As noted, it is somewhat surprising that **G5** and **G6** are sufficient to capture this notion of rationality. Whether or not a player is maximizing their expected utility depends on their *quantitative* beliefs; however, while **G5** and **G6** specify the possible supports for player i's beliefs,

they say nothing about the actual weights placed on the individual outcomes. Nor could they—the language $\mathcal{L}_B(\Phi_\Gamma^{\mathfrak{d}})$ cannot express anything beyond such qualitative properties of the measures $Pr_i(\omega)$.

Expressivity, however, is a double-edged sword: when working with a less expressive language, though we are more limited in the possible axioms we have available, there are also fewer validities to worry about proving. This, in essence, is why $\mathsf{GL}_\Gamma^{\mathfrak{d}}$ can be a complete axiomatization: the properties of rationality it fails to encode are precisely those properties that are not expressible in the language at all.

A richer language—in particular, one with a finer-grained representation of belief—may not be axiomatizable at all, or at least not using the techniques in this paper. Consider, for example, a language with belief modalities B_i^α for each $\alpha \in [0,1]$, where $B_i^\alpha\varphi$ is interpreted as saying that player i assigns probability α to φ. In this case, $\mathsf{GL}_\Gamma^{\mathfrak{d}}$ (replacing B_i by B_i^1) is sound but certainly not complete. It can easily happen, for example, that in the game Γ it is only rational for player i to play σ_i if she assigns probability $\frac{1}{2}$ to σ_{-i}; however, the corresponding validity

$$(play_i(\sigma_i) \wedge RAT_i) \to B_i^{\frac{1}{2}}\, play_{-i}(\sigma_{-i})$$

is clearly not a theorem of $\mathsf{GL}_\Gamma^{\mathfrak{d}}$. Moreover, extending $\mathsf{GL}_\Gamma^{\mathfrak{d}}$ to this richer language runs into difficulties. The axiom schemes **G5** and **G6** essentially work by insisting that the players' beliefs be compatible with rationality or its negation, respectively. In the language $\mathcal{L}_B(\Phi_\Gamma^{\mathfrak{d}})$, this amounts to specifying the possible supports for the players' beliefs, which can be written using finite formulas since each Σ_{-i} is finite and therefore has only finitely-many subsets. By contrast, in the language with belief modalities B_i^α for every $\alpha \in [0,1]$, the "formula" that says that player i's beliefs are compatible with rationality may be infinitely long.

A still richer language, however, can circumvent these difficulties entirely. Fix a game Γ and consider the language of *linear likelihood inequalities* defined by the grammar

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid a_1\ell_i(\varphi_1) + \cdots + a_k\ell_i(\varphi_k) \geq b,$$

where $p \in \Phi_\Gamma$, $i \in I$, $k \in \mathbb{N}$, and $a_1, \ldots, a_k, b \in \mathbb{R}$. The *likelihood terms* $\ell_i(\varphi)$ are meant to be read as "the probability of φ according to player i", and a likelihood formula $a_1\ell_i(\varphi_1) + \cdots + a_k\ell_i(\varphi_k) \geq b$ should be thought of as asserting the corresponding inequality. More precisely, we interpret such formulas in a Γ-structure M as follows:

$$\begin{aligned} &[\![a_1\ell_i(\varphi_1) + \cdots + a_k\ell_i(\varphi_k) \geq b]\!]_M \\ :=\ &\{\omega \in \Omega \,:\, \textstyle\sum_{j=1}^k a_j Pr_i(\omega)([\![\varphi_j]\!]_M) \geq b\}. \end{aligned}$$

For example, the formula $\ell_i(\varphi) \geq 1$ says that player i assigns probability at least (and therefore exactly) 1 to φ, while the formula $(\ell_i(\varphi) \geq \frac{1}{2}) \wedge (\ell_i(\neg\varphi) \geq \frac{1}{2})$ says that player i assigns probability $\frac{1}{2}$ to φ. See (Halpern 2003) for a thorough discussion of this and related logics; a sound and complete axiomatization is given in (Fagin, Halpern, & Megiddo 1990).

In this language, rationality in the sense of expected utility maximization can be *defined*, thus obviating the need for a separate axiomatization. Indeed, if we let $\sigma_i \succeq \sigma_i'$ be an abbreviation for the formula

$$\begin{aligned} &\textstyle\sum_{\sigma_{-i}\in\Sigma_{-i}} u_i(\sigma_i, \sigma_{-i})\ell_i(play_{-i}(\sigma_{-i})) - \\ &\quad \textstyle\sum_{\sigma_{-i}\in\Sigma_{-i}} u_i(\sigma_i', \sigma_{-i})\ell_i(play_{-i}(\sigma_{-i})) \geq 0, \end{aligned}$$

which says that the expected value according to player i of playing σ_i is no less than the expected value of playing σ_i', then it is easy to see that

$$RAT_i^{\mathfrak{d}_i^{eu}} \leftrightarrow \bigvee_{\sigma_i\in\Sigma_i} \Big(play_i(\sigma_i) \wedge \bigwedge_{\sigma_i'\in\Sigma_i} \sigma_i \succeq \sigma_i' \Big)$$

is valid. Since rationality is expressible in this language, it is axiomatized as well.

6 Further work

Equilibrium analysis in game theory is predicated on the notion of rationality: solution concepts like Nash equilibrium are formulated in terms of best responding to beliefs. As such, one natural application of a logic of rationality is to the analysis of equilibria.

The language $\mathcal{L}_B(\Phi_\Gamma)$ has formulas that represent pure strategy choices, but not mixed strategies. In the context of Nash equilibrium, this difference turns out to be (at least formally) innocuous: one can view a mixed strategy $\ell_i \in \Delta(\Sigma_i)$ either as a conscious randomization on the part of player i, or as the common conjecture of the players $j \neq i$ about what pure strategy i will choose—either way, the set of mixed strategy Nash equilibria stays the same. However, this insensitivity is, in part, dependent on that fact that rationality in the sense of expected utility maximization plays well with mixing: $\ell_i \in \Delta(\Sigma_i)$ maximizes player i's expected utility (with respect to some fixed beliefs) if and only if every pure strategy σ_i in the support of ℓ_i maximizes expected utility. But this correspondence breaks down when "expected utility maximization" is replaced with the generalized notion of rationality presented in Section 2.3: in the context of an arbitrary decision rule $\mathfrak{d}_i : \Delta(\Sigma_{-i}) \to 2^{\Sigma_i}$, there is no principled way to extend the notion of "best response" from Σ_i to $\Delta(\Sigma_i)$. This suggests that further research into the interaction between pure and mixed strategies under general decision rules may be fruitful.

One might also wonder whether there are useful logical characterizations of when a given strategy constitutes a Nash equilibrium. Similar to the above, a related line of inquiry arises by considering generalized decision rules and the correspondingly generalized notions of equilibria they induce. In principle, it makes sense to consider "$\mathfrak{d}$-Nash equilibria", defined by replacing $\mathfrak{d}^{eu}$ with an arbitrary profile of decision rules $\mathfrak{d}$ in the definition of Nash equilibrium. It is certainly too much to hope that Nash's famous existence theorem apply in full force to this wider concept; however, properties of $\mathfrak{d}$ that suffice to guarantee the existence of equilibria are of interest, and potentially admit a logical characterization. Such questions are the subject of ongoing research.

Although in this paper we focused on rationality, our approach to axiomatizing rationality applies to a wider class of properties. Define an *i-subjective property* to be any property of i that i is sure of; that is, a formula φ such that

$\varphi \leftrightarrow B_i\varphi$ is valid. Rationality is i-subjective, as is whether i is jealous (e.g., believes somebody has some property that she doesn't have, but wishes she did), whether i is cautious (e.g., assigns positive probability to all her opponents' actions), etc. We believe that the techniques developed in this paper could be used to provide a sound and complete axiomatization of any i-subjective property whose truth is determined by i's beliefs and what i plays. We hope to return to this topic in future work.

7 Acknowledgements

Bjorndahl is supported in part by NSF grants IIS-0812045, CCF-1214844, DMS-0852811, and DMS-1161175, and ARO grant W911NF-09-1-0281. Halpern is supported in part by NSF grants IIS-0812045, IIS-0911036, and CCF-1214844, by MURI grant FA9550-12-1-0040, and by ARO grant W911NF-09-1-0281. Pass is supported in part by an Alfred P. Sloan Fellowship, a Microsoft Research Faculty Fellowship, NSF Awards CNS-1217821 and CCF-1214844, NSF CAREER Award CCF-0746990, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2-0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

References

Battigalli, P., and Siniscalchi, M. 2002. Strong beliefs and forward-induction reasoning. *Journal of Economic Theory* 106:356–391.

Bjorndahl, A.; Halpern, J. Y.; and Pass, R. 2011. Reasoning about justified belief. In *Theoretical Aspects of Rationality and Knowledge: Proc. Thirteenth Conference (TARK 2011)*, 221–227.

Brandenburger, A., and Dekel, E. 1987. Rationalizability and correlated equilibria. *Econometrica* 55:1391–1402.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning About Knowledge*. Cambridge, Mass.: MIT Press. A slightly revised paperback version was published in 2003.

Fagin, R.; Halpern, J. Y.; and Megiddo, N. 1990. A logic for reasoning about probabilities. *Information and Computation* 87(1/2):78–128.

Halpern, J. Y., and Moses, Y. 2007. Characterizing solution concepts in games using knowledge-based programs. In *Proc. Twentieth International Joint Conference on Artificial Intelligence (IJCAI '07)*, 1300–1307.

Halpern, J. Y. 2003. *Reasoning About Uncertainty*. Cambridge, Mass.: MIT Press.

Hayashi, T. 2008. Regret aversion and opportunity dependence. *Journal of Economic Theory* 139(1):242–268.

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. Cambridge, Mass.: MIT Press.

Tan, T., and Werlang, S. 1988. The Bayesian foundation of solution concepts of games. *Journal of Economic Theory* 45(45):370–391.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Dynamic Causal Calculus

Alexander Bochman
Computer Science Department,
Holon Institute of Technology,
Israel

Abstract

We introduce dynamic causal calculus, a nonmonotonic formalism that can be viewed as a direct logical counterpart of the action description language C+ from (Giunchiglia et al. 2004). We formulate a nonmonotonic semantics of the associated causal language, and compare this semantics with the indirect, two-stage semantics for C+, given in (Giunchiglia et al. 2004). It will be shown, in particular, that the suggested semantics allows us to alleviate syntactic distinctions between propositional atoms, maintained by C+, as well as type restrictions imposed on its causal laws. We will describe also a logical formalism of dynamic causal inference that constitutes a complete description of the logic that is adequate for this dynamic calculus.

Introduction

Unlike general descriptions of temporal dynamics in temporal logics and related logical formalisms, action theories in AI have to deal primarily with two quite specific reasoning tasks, namely the *prediction* task (what are the results of a given sequence of actions from an initial state) and the *planning* task (what sequence of actions could lead from an initial state to a target goal state). These reasoning tasks immediately lead to a triple of famous problems, known as the frame, ramification and qualification problems (see (Shanahan 1997)). It was realized quite early that classical logic and its temporal/dynamic extensions, taken by themselves, encounter difficulties in resolving these problems. More precisely, it has become clear that these problems have an essentially *nonmonotonic* character, so their proper solution requires augmenting purely logical, monotonic reasoning with an appropriate mechanism for making nonmonotonic conclusions[1].

In recent years a dominant approach to solving these problems has been based, in one form or another, on what can be broadly termed causal reasoning. Given a set of action and (causal) rules describing the domain, the causal approach employs a distinction between facts that hold in a situation versus facts that are caused (or explained) by other facts and the rules. In this context, the corresponding causal closure assumption (see. e.g., (Reiter 2001)) can be viewed as a particular form of an old philosophical principle of *universal causation*, which amounts to the requirement that all facts that hold in a situation should be either caused by other occurrent facts, or else preserve their truth-values in time (due to the accompanying *inertia* assumption). A direct incorporation of such causal assertions into the language of the situation calculus has been proposed in (Lin 1995; 1996), and has been shown to provide a natural account of both the frame and ramification problems. Subsequently, a general formal framework for this kind of causal reasoning, called a causal calculus, has been suggested in (McCain and Turner 1997).

An elaborate implementation of the above causal principles in reasoning about actions has been given in (Giunchiglia et al. 2004). The formalism of (Giunchiglia et al. 2004), however, is a multi-sorted and multi-layered representation framework. As its top layer, it employs a causal action description language C+ that provides high-level descriptions of action domains in terms of three kinds of propositional atoms (actions, simple fluents and statically determined fluents) and three different kinds of causal laws (static laws, action dynamic laws and fluent dynamic laws). Domain descriptions in this language are then instantiated by assigning temporal stamps to propositions, and incorporating the resulting descriptions into an atemporal causal calculus of (McCain and Turner 1997). The models of the resulting causal theories are viewed then as intended models of the source, higher-level action descriptions.

In this study[2] we will attempt to single out and 'streamline' the logical framework behind the language C+. To this end, we will introduce a dynamic generalization of the original causal calculus, which will be formulated, ultimately, in terms of a single basic kind of *dynamic* causal rules. This dynamic calculus will provide a direct and uniform logical description for the language C+. In addition, we will describe also a logical (monotonic) system of dynamic causal inference that will constitute a concise logical framework for causal reasoning in dynamic domains.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] Even the well-known "monotonic" solution to the frame problem, described in (Reiter 1991), can be seen as a result of compiling this nonmonotonic information into the successor state equivalences.

[2] Preliminary version of this paper has appeared in (Bochman 2013).

The plan of the paper is as follows. After an overview of the original (atemporal) causal calculus and its use in the framework of (Giunchiglia et al. 2004), we will introduce a dynamic causal calculus as a direct reformulation of the action description language C+. We will describe a modified nonmonotonic semantic for this dynamic causal language that is based on a comprehensive principle of universal causation, and show that this modification allows us, in particular, to alleviate most of the syntactic restrictions imposed in C+ on the form of its causal rules, as well as syntactic distinctions between the propositions of the language. Finally, we will describe a logical formalism of dynamic causal inference that constitutes an underlying logic for causal reasoning in action domains.

The Causal Calculus

Throughout this study, we will assume that our basic language is a classical propositional language with the usual connectives and constants $\{\wedge, \vee, \neg, \rightarrow, \mathbf{t}, \mathbf{f}\}$. $\models$ and Th will stand, respectively, for the classical entailment and the associated logical closure operator. In what follows, we will also identify propositional interpretations ('worlds') with the sets of propositional formulas that hold in them.

A *causal rule* is a rule of the form $A \Rightarrow B$, where A and B are classical propositions. We will informally interpret such rules as saying plainly *"A causes B"*.[3]

By a *causal theory* we will mean an arbitrary set of causal rules. For a set u of propositions and a causal theory Δ, we will denote by $\Delta(u)$ the set of all propositions that are caused by u in Δ, that is,

$$\Delta(u) = \{B \mid A \Rightarrow B \in \Delta, \text{ for some } A \in u\}$$

Then the nonmonotonic semantics of a causal theory can be defined as follows.

Definition 1. A world (= propositional interpretation) α is an *exact model* of a causal theory Δ if it is a unique model of $\Delta(\alpha)$. The set of exact models forms a *nonmonotonic semantics* of Δ.

The above semantics of causal theories coincides, in effect, with the semantics for such theories, described in (McCain and Turner 1997) and (Giunchiglia et al. 2004). It can also be verified that exact models of a causal theory are precisely the worlds that satisfy the following condition:

$$\alpha = \mathrm{Th}(\Delta(\alpha)).$$

Accordingly, exact worlds are not only closed with respect to the causal rules, but also such that any proposition that holds in them is caused (that is, explained) ultimately by other propositions.

An overview of the Language C+

The underlying propositional language of the action description language C+, introduced in (Giunchiglia et al. 2004), is somewhat more general than the standard classical propositional language in that it is based on a multi-valued propositional signature that consists of a set of constants, along with a function Dom assigning every constant c a nonempty finite set $Dom(c)$ of values. Propositional atoms in this signature are expressions of the form $c = v$, where c is a constant, while v is one of its possible values. Still, propositional formulas in this language are defined as usual combinations of atoms with the help of the ordinary classical connectives. Moreover, *Boolean* constants are defined in this setting as a special kind of constants whose domain is the set $\{\mathbf{f}, \mathbf{t}\}$ of truth values.

An *action description* in C+ is defined as a set of causal laws. There are, however, three kinds of causal laws in C+, and the differences between them are based, ultimately, on a distinction between three kinds of constants, and thereby three kinds of atoms, stipulated by the theory. First, propositional atoms are partitioned into *action* atoms and *fluent* atoms, while the latter are further partitioned into *simple* and *statically determined* fluents. A *fluent formula* is defined as a formula such that all constants occurring in it are fluent constants, whereas an *action formula* is a formula that contains at least one action constant and no fluent constants.

Granted the above syntactic distinctions among propositional atoms, the following three kinds of causal laws are defined in C+:

- *Static laws* are expressions of the form

  $$\mathbf{caused}\ F\ \mathbf{if}\ G,$$

  where F and G are fluent formulas;
- *Action dynamic laws* are expressions of the form

  $$\mathbf{caused}\ F\ \mathbf{if}\ G,$$

  where F is an action formula and G is a formula;
- *Fluent dynamic laws* are expressions of the form

  $$\mathbf{caused}\ F\ \mathbf{if}\ G\ \mathbf{after}\ H,$$

  where F is a fluent formula that does not contain statically determined constants, G is a fluent formula, and H is an arbitrary formula.

Static laws are used in C+ to talk about causal dependencies between fluents in the same state, while action dynamic laws are purported to express causal dependencies between concurrently executed actions. In accordance with their very name, statically determined fluent constants are allowed in the heads of static laws, but not in the heads of dynamic laws. As we will see in what follows, the necessity of introducing a separate syntactic sort of statically determined fluents stems from a particular definition of a state of a transition system, which has been used in interpreting action descriptions in C+.

The language C+ employs also a number of abbreviations and names for special kinds of causal laws. Two such abbreviations play an especially important role in the descriptions of action domains that serve to illustrate the general theory. If c is a simple fluent constant, then

$$\mathbf{inertial}\ c$$

[3] (Giunchiglia et al. 2004) adopted a more cautious informal reading of such rules, namely *"If A holds, then B is caused"*.

stands for the fluent dynamic laws

$$\textbf{caused } c = v \textbf{ if } c = v \textbf{ after } c = v$$

for all $v \in Dom(c)$. These laws provide an encoding of the *inertia assumption* for (simple) fluent atoms. Similarly, if c is an action constant, the expression

$$\textbf{exogenous } c$$

stands for the action dynamic laws

$$\textbf{caused } c = v \textbf{ if } c = v$$

for all $v \in Dom(c)$. These laws make action atoms exogenous in an action domain, which exempt them, in effect, from explanation.

Interpretations and models of action descriptions in C+ are defined indirectly by translating them into plain causal theories. To begin with, for every natural number m, an action description D is transformed into an atemporal causal theory D_m as follows. First, 'time stamps' i: for $i \in \{0, \ldots, m\}$ are inserted in front of every occurrence of every atom in propositional formulas. Then each static law is translated into the following set of causal rules, for every $i \leq m$:

$$i : A \Rightarrow i : B,$$

where $i : F$ is the result of inserting i: in front of every occurrence of every atom in a formula F. Similarly, any action dynamic law is translated into a set of causal rules of the same form, but only for $i < m$.

Finally, any fluent dynamic law is translated into the following set of causal rules:

$$(i : C) \wedge (i + 1 : A) \Rightarrow i + 1 : B,$$

for every $i < m$.

As a concluding step, in order to deal with the initial states, the following causal rules are added to the resulting causal theory:

$$0 : l \Rightarrow 0 : l$$

for every *simple* fluent atom l. These rules make simple fluent atoms exogenous (self-explained) in the initial state. As a result, we obtain an ordinary causal theory, and the exact models of this theory are considered to be the models of the original action description in C+. Such models can be visualized as histories of length m of the source dynamic domain. More precisely, these are histories in which the initial state is 'self-explainable', but every subsequent state is already causally explained by the preceding state and actions taken in it.

(Giunchiglia et al. 2004) contains also a more general semantic construction, according to which an action description D in C+ describes, in effect, a *transition model* (i.e., a set of states with a set of transitions among them) in which states are the models of the 'smallest' (static) causal theory D_0 (which is a theory D_m for $m = 0$)[4], while transitions correspond precisely to the models of the minimal dynamic causal theory D_1 (that is, a theory D_m for $m = 1$). It has been shown in (Giunchiglia et al. 2004, Proposition 8) that, for any $m > 0$, models of a causal theory D_m are exactly histories (paths) of length m in this transition model.

[4]Note that, by the definition of the translation, D_0 includes no dynamic laws, and only '0-stamped' static causal laws.

On statically determined fluents

The above construction of a canonical transition model for an action description implicitly relied on the property (stated as Proposition 7 in (Giunchiglia et al. 2004)) that for any transition $\langle s, e, t\rangle$ in the above sense (that is, for any model of D_1), both s and t are states (models of D_0). As has been observed in the paper, the validity of this property depends essentially on the fact that the heads of fluent dynamic laws were not allowed to contain statically determined fluent constants, which explained the very need for a *syntactic* (type) separation between simple and statically determined fluent constants in the framework of the general theory described in (Giunchiglia et al. 2004).

It should be noted, however, that, taken by itself, the distinction between simple and statically determined fluents is not as clear-cut as it may seem. Consider, for example, a fluent predicate *Safe* (for a baby), discussed first in (Myers and Smith 1988). As was observed in (Giunchiglia and Lifschitz 1995), there might be different kinds of safety in this sense. An object in question may be a heavy hammer, so it is safe for a baby only if it is far from the reach, say on a table. This is a clear example of a statically determined fluent. But the object may be a doll, in which case its safety is clearly an ordinary, simple inertial fluent. Furthermore, we can easily imagine an action like *Isolate* that wrap up a potentially non-safe hammer with a material that makes it fully safe for a baby. On an account of (Giunchiglia et al. 2004) this would require, however, a syntactic shift for this fluent from statically determined to simple one.

The distinction between simple and statically determined fluents is only a very special case of a broader distinction between inertial and non-inertial fluents. Such distinctions between fluents play an important role in reasoning about actions and change in AI, so they should not be obliterated, or neglected. Still, they need not be defined as syntactic distinctions between types of fluents. Rather, they could be viewed as a by-product of the actual ways these fluents are described, or defined, in (causal) rules. Thus, we can still maintain that a particular fluent is *statically determined* in a given action description if it happens to appear only in heads of its static rules. But this characterization could change, for example, with an addition of new dynamic rules which involve this fluent in their heads. Such a 'dynamic' characterization would be unfeasible, however, in the framework of (Giunchiglia et al. 2004) because statically determined fluents play an essential role in the definition of the very notion of a state of a transition model that determines, in turn, the semantics of C+. Accordingly, in order to make the language of C+ syntactically uniform, we should change also its semantic description. This will be done in the next section where we will describe an alternative semantics for the dynamic causal calculus that will provide a logical representation for the action descriptions in C+.

As a final note here, we should mention that there are also other, perhaps even more serious, problems with the representation of statically determined fluents in C+. Thus, many fluents of this kind are naturally defined using recursive (Prolog-like) rules. It turns out, however, that such definitions are not expressible in C+, though they are nat-

urally expressible in an alternative action description language, namely the language B (see (Gelfond and Lifschitz 1998)). As a matter of fact, representation of such recursively determined fluents creates similar difficulties also for the dynamic causal calculus, described in this study, since the latter closely follows the general semantic framework of C+[5].

Dynamic Causal Calculus

In this section we are going to describe a dynamic causal calculus that is intended to provide a direct logical reformulation of the action description language C+ and its semantics. As a first step, we will use a more convenient, 'logic-oriented' notation, namely, we will rewrite a dynamic law $\mathbf{caused}\ B\ \mathbf{if}\ A\ \mathbf{after}\ C$ as a causal rule of the form

$$C.A \Rightarrow B.$$

We will call such rules *dynamic causal rules*. Moreover, extending our previous re-interpretation of atemporal causal rules, we will assign a more 'active' informal reading to such rules, namely *"After C, A causes B"*. By the intended interpretation, a rule $C.A \Rightarrow B$ describes a dynamic transition from a state that satisfies proposition C to a subsequent state in which B is caused by A.

Remark. As a matter of fact, the above dynamic causal rules are somewhat ambiguous from a syntactic point of view. On a more abstract level, such a rule can be viewed simply as an instantiation of a primitive ternary propositional operator. There are, however, at least two other, more articulated possibilities[6]. Thus, a dynamic causal rule $C.A \Rightarrow B$ could be viewed as a plain causal rule $A \Rightarrow B$ that is conditioned by a preceding context C. In fact, this 'parsing' agrees with the informal reading of such rules, given above. Furthermore, we will see that this understanding of dynamic causal rules as conditional static rules provides a natural justification for the postulates of the associated dynamic causal inference that will be given below. Still, a different parsing possibility consists in viewing such rules as binary causal rules, though with complex premises consisting of pairs of propositions (C, A). Again, we will see later that this reading can also be given a formal support, due to a possible translation of such rules as propositions of the form $(C \circ A) \rightarrow B$ in arrow logic.

As a second step in our reformulation, we will uniformly view both static and action dynamic laws of the form $\mathbf{caused}\ B\ \mathbf{if}\ A$ as plain (static) causal rules $A \Rightarrow B$. Moreover, in the version of the dynamic causal calculus that we will present in this study, we will make one further step and identify such rules with a special kind of dynamic rules of the form $\mathbf{t}.A \Rightarrow B$, where $\mathbf{t}$ is the truth constant. In other words, we will adopt the following definition:

$$A \Rightarrow B \ \equiv_{df}\ \mathbf{t}.A \Rightarrow B.$$

According to this definition, static causal rules are precisely rules that are valid after any legitimate transition. The consequences and variations created by this identification will be discussed below.

The above reduction of static causal rules to a special kind of dynamic rules makes the dynamic causal rules the only kind of rules of the dynamic calculus. In accordance with this, a *dynamic causal theory* will be defined below simply as a set of dynamic causal rules.

Now we are going to provide a direct description of the nonmonotonic semantics of dynamic causal theories. The guiding principle behind this nonmonotonic semantics will be a thorough enforcement of the *principle of universal causation*, according to which every state of a dynamic model should be explained (i.e., caused) by a preceding state and the causal rules of the domain.

Given a dynamic causal theory Δ and worlds α, β, we will denote by $\Delta(\alpha.\beta)$ the set

$$\{C \mid A.B \Rightarrow C \in \Delta \text{ for some } A \in \alpha, B \in \beta.\}$$

For any given state β, the set $\Delta(\alpha.\beta)$ is precisely the set of propositions in β that are caused due to a transition from α.

Definition 2.(i) A pair (α, β) of worlds will be called an *exact transition* with respect to a dynamic causal theory Δ if β is the unique model of $\Delta(\alpha.\beta)$, that is

$$\beta = \mathrm{Th}(\Delta(\alpha.\beta)).$$

(ii) An *exact transition model* of a dynamic causal theory Δ is a set of worlds $\mathfrak{I}$ such that, for any $\beta \in \mathfrak{I}$ there is $\alpha \in \mathfrak{I}$ such that (α, β) is a an exact transition wrt Δ.

An exact transition is a transition between two states in which the resulting state is fully explained (caused), given the preceding state and the causal laws of the domain. It is important to note that if (α, β) is an exact transition, then the output world β is always closed with respect to the static causal rules (for our definition of the latter), namely if $A \Rightarrow B$ (that is, $\mathbf{t}.A \Rightarrow B$) belongs to Δ and $A \in \beta$, then $B \in \beta$. Note now that any world of an exact transition model is also an output of some exact transition. Accordingly, we immediately obtain the following

Lemma 1. *Any world of an exact transition model of a dynamic causal theory Δ is closed with respect to the static causal rules of Δ.*

It can be easily verified that the union of two exact transition models is also an exact transition model. Consequently, if a dynamic causal theory has at least one exact transition model, it has a unique maximal such model. We will call the latter the *canonical exact transition model* of a dynamic causal theory.

In the next section we are going to compare the above dynamic causal calculus with the theory presented in (Giunchiglia et al. 2004).

[5]Recently, (Lee, Lifschitz, and Yang 2013) have suggested a generalization of the language C+, called a language BC, that incorporates expressive capabilities of the language B. Instead of a causal interpretation, however, the authors have provided only formal translations of this language directly into the framework of logic programming.

[6]Compare with a similar compositional analysis of general and relevant conditionals, discussed in (Beall et al. 2012).

Comparisons with C+

In accordance with the reformulation procedure sketched in the preceding section, any action description in the action description language C+ can be immediately transformed into a dynamic causal theory in our sense. Despite the obvious similarities, however, this reformulation reveals also some basic differences between the two formalisms. To begin with, the language of the above dynamic causal calculus is thoroughly *uniform* in that it does not presuppose any a priori, *syntactic* distinctions between its propositional atoms, and it employs only a single form of causal rules, instead of three kinds of causal laws used in C+. On the other hand, the above notion of an exact transition model is somewhat different, and apparently more restricted, than the corresponding notion of a model for C+. In what follows we will discuss these differences in more details.

Fluents versus Actions

Many action theories in AI, as well as some general dynamic formalisms, maintain rigid semantic and syntactic distinctions between *fluent* propositions that describe particular states or situations and *actions* that describe transitions between states. Thus, the situation calculus (Reiter 2001) treats actions essentially as modifiers, or functions on situations. Similarly, Propositional Dynamic Logic (see (Harel, Kozen, and Tiuryn 2000)) interprets actions as binary relations on states.

However, in contrast to the above formalisms, it can be immediately observed that there are no *inherent* syntactic differences between fluent and action atoms in C+; both kinds are translated as (temporally indexed) propositional atoms in the underlying causal calculus. Of course, there are still important differences between these two kinds of propositions: (simple) fluents are governed by the inertia principle, while action atoms are normally treated in C+ as exogenous (see above). However, these distinctions are semantical, and they can be secured by incorporating causal rules that characterize, respectively, inertia and exogeneity for these atoms directly into the corresponding action descriptions. Once we add such rules to the action descriptions, there is no need to maintain a syntactic distinction between fluents and actions in C+, and there is no need to maintain a separate category of action dynamic laws in addition to static laws.

Remark. The treatment of actions in C+ makes the latter much similar to *temporal* formalisms for analyzing general computation processes, such as Linear Temporal Logic (LTL). In the latter, computational processes are represented as plain temporal sequences of states (which are characterized by fluent propositions), and they do not use explicit action descriptions. Still, actions are usually encoded in these formalisms by using associated 'action fluents' that appropriately constrain such temporal sequences (see, e.g., (Calvanese, De Giacomo, and Vardi 2002; De Giacomo and Vardi 2013)).

State Constraints and the Principle of Universal Causation

An exact transition model of a dynamic causal theory was defined above as a set of states in which every state is caused as a result of some exact transition. As a result, any state of such a model will be closed with respect to the static laws on our reformulation of the latter. Moreover, it can be verified that any such state will be a state in the sense of (Giunchiglia et al. 2004). Consequently, we will obtain that any exact model in our sense will correspond to a model (transition system) in the sense of (Giunchiglia et al. 2004):

Theorem 2. *If Δ is an action description in C+ and Δ_C its corresponding dynamic causal theory, then any exact transition model of Δ_C is a transition model of Δ.*

Still, our semantics for dynamic causal theories is more restrictive, since it requires that any state of the model, *including the initial state*, should be an output of some transition, whereas (Giunchiglia et al. 2004) adopted more relaxed requirements on initial states. As we have seen earlier, the construction of a causal theory for a given action description in C+ necessarily involves an addition of causal rules that make simple fluents exogenous (self-explainable) in the initial state. As a result, the framework (Giunchiglia et al. 2004) exempts to some extent initial states from the need of explanation, though it still requires that such states should be closed with respect to the static laws (which are completely separated from dynamic ones) and, moreover, that any statically determined fluent literal that holds in an initial state should still be explained (caused) by the static laws.

Remark. As a matter of fact, our definition of an exact transition almost coincides with the corresponding definition of a *causally explained transition*, given in (Giunchiglia and Lifschitz 1998) for a more restricted action description language C, a predecessor of C+[7]. In fact, the only difference between the two definitions is that (Giunchiglia and Lifschitz 1998) required further that both the initial and resulting states of such a transition should be closed with respect to the static causal laws. On our construction, this additional requirement is accounted for, respectively, as a by-product of our definition of static causal rules on the one hand (for the resulting states), and an exact transition model on the other hand (for the initial states).

In order to make a proper assessment of the above discrepancy, we should distinguish two aspects of the difference, conceptual and practical one. On the conceptual side, we believe that an ultimate reason for imposing even the above minimal restrictions on initial states in C+ stems from a broader requirement that such states should be somehow accessible in accordance with the laws of the domain. In other words, any state of a dynamic system should be consistently viewed as a result of some legitimate transition (including possible 'loops' in this state). Speaking more generally, we contend that static laws and constraints should be viewed as constraints that are effective after every legitimate transition, and vice versa, any constraint that happens to hold after any

[7] I am grateful to Vladimir Lifschitz for pointing this out (personal correspondence).

possible transition should be considered as a static law of the domain.

The suggested semantics of exact transition models allows us to treat static causal rules as a special case of general dynamic causal rules. In addition, it allows us to remove the syntactic distinction between simple and statically determined fluents. Furthermore, the principle of universal causation, used as a conceptual basis of this semantics, implies also some important consequences concerning general principles of reasoning in dynamic domains.

Generally speaking, the principle of universal causation provides logical foundations for *abductive reasoning*, namely for backward reasoning from effects to their causes (see (Bochman 2007)). This kind of reasoning constitutes an essential part of our commonsense reasoning, especially in reasoning about actions and change, and it occupies also an important part of reasoning in current action theories in AI. For example, the well-known regression method ((Reiter 2001)) can be viewed as a systematic implementation of abductive reasoning in the situation calculus.

Due to its causal foundations, abductive inference is also an essential, though implicit, part of the representation framework of C+; this is because every state of a transition model for an action description, *except the initial one*, is explained as an output of some causal transition. The associated abductive explanation may sanction, in particular, some further static constraints for such states, constraints that arise as a by-product of the fact that the state in question is a result of a particular action with some further effects. By the same token, however, the initial states in C+ are exempted from the abductive explanation of this kind, which, at least in some regular domains, may lead to a loss of important information about these states.

Persistent Action Domains

In case we don't accept the above understanding of static laws, we should maintain two separate kinds of state constraints, dynamic and purely static ones, as is actually done in (Giunchiglia et al. 2004)), as well as in many other action formalisms. This separation would allow us to include a broader class of transition models as admissible models for action descriptions, namely models that involve (initial) states that need not be a result of some transition, though they still satisfy the static laws of the domain. Moreover, we can easily construct some artificial, though logically consistent, action domains for which such a distinction would be necessary, namely action domains in which there are distinguished initial states that lack some property that holds for any state that results from a transition. Still, to find 'real-life' examples of such domains is really difficult (beyond the famous 'Big Bang' exception in Physics). This naturally brings us to the practical side of the difference between the two formalisms, namely to the question whether, and if so, how much we miss in restricting the semantics of action descriptions to exact transition models in our sense.

It turns out that for a fairly broad class of action descriptions in the language C+ (including all the examples given in (Giunchiglia et al. 2004)), we can *guarantee* in advance that any transition model of C+ that satisfies a given action description can be extended to an exact transition model in our sense. For such action descriptions, our dynamic causal calculus provides the same answers to the queries as the original theory of (Giunchiglia et al. 2004)). In what follows, we will illustrate this correspondence for a broad class of what we will call persistent action domains.

Informally speaking, a persistent action domain is a dynamic domain that does not involve involuntary, 'natural' actions that lead to unavoidable, necessary changes of some state. For such action domains, any legitimate state either remains persistent in the absence of any further actions upon it (alias after an action *Wait*), or else it can be forced to persist by using suitable (voluntary) actions.

Formally, by a *persistent action domain* we will mean any action description D in the language C+ such that for any state s of D (that is, for any model of D_0), there is a consistent transition (i.e., a model of D_1) from s to s.

It turns out to be surprisingly difficult to express the above semantic property of transition models in terms of some syntactic restrictions on action descriptions in C+. Still, it is easy to verify the validity of this persistence property for many descriptions used in action theories. Thus, in the central Monkey and Bananas example from (Giunchiglia et al. 2004) every simple fluent is inertial, so any state of the transition model can be shown to persist in the absence of actions. On the other hand, for the case of non-inertial fluents that tend to change by themselves, the corresponding action descriptions usually contain actions that make them persist. Thus, the pendulum domain from (Giunchiglia et al. 2004) involves an action *Hold* that keeps the position of the pendulum (which tends to sway otherwise).

Now, in persistent action domains, any state of a transition model can be consistently viewed as a result of some transition, which immediately leads to the following

Theorem 3. *If Δ is a persistent action description in C+ and Δ_C its corresponding dynamic causal theory, then the canonical transition model of Δ coincides with the exact transition model of Δ_C.*

As a matter of fact, the above correspondence between C+ and the dynamic causal calculus can be extended, though in a somewhat weaker form, even beyond persistent action domains. To this end, we should note that practically all queries that are usually formulated in action formalisms are 'future-oriented', namely they ask whether there is a path in a transition model from a given initial state that satisfies certain further requirements (such as whether it ends in a target goal state). Now, in many cases we can extend the source action description in C+ with some auxiliary actions that will allow us again to reconstruct any given initial state as an output of some exact transition (though in the extended action description). Moreover, this can be done without changing the future of the original states of the transition model, but only by augmenting their 'past'. As an immediate consequence, we will obtain that, for such action domains, the dynamic causal calculus will provide the same answer to such queries as the original action descriptions in C+.

Causal Inference

The framework described in the preceding section is a typical example of a nonmonotonic formalism; conclusions that can be obtained on the basis of the exact transition semantics can change non-monotonically if we add some further facts or causal rules to the original dynamic causal theory. Still, as with other formalisms for nonmonotonic reasoning (see (Bochman 2011)), the causal rules of the dynamic causal calculus presuppose a certain underlying logic that agrees with the above nonmonotonic semantics. Such a logic will provide us with a formal description of the associated dynamic causal inference.

Causal Inference Relations

The original causal calculus of (McCain and Turner 1997) has been defined only semantically, but (Bochman 2004) has described a logical formalism of causal inference relations, which has been shown to provide a complete formalization of a logical (monotonic) reasoning in causal theories. From a logical point of view, causal inference relations were defined as sets of causal rules that were required to satisfy almost all the usual postulates of classical inference, except Reflexivity $A \Rightarrow A$. The latter feature has turned out to be essential for an adequate representation of causal reasoning.

Definition 3. A *causal inference relation* is a relation $\Rightarrow$ on the set of propositions satisfying the following conditions:

(Strengthening) If $A \vDash B$ and $B \Rightarrow C$, then $A \Rightarrow C$;

(Weakening) If $A \Rightarrow B$ and $B \vDash C$, then $A \Rightarrow C$;

(And) If $A \Rightarrow B$ and $A \Rightarrow C$, then $A \Rightarrow B \wedge C$;

(Or) If $A \Rightarrow C$ and $B \Rightarrow C$, then $A \vee B \Rightarrow C$;

(Cut) If $A \Rightarrow B$ and $A \wedge B \Rightarrow C$, then $A \Rightarrow C$;

(Truth) $\mathbf{t} \Rightarrow \mathbf{t}$;

(Falsity) $\mathbf{f} \Rightarrow \mathbf{f}$.

The rule Or permits reasoning by cases; this feature can be seen as one of the main advantages of causal reasoning as compared with, say, default logic. It indicates that the causal logic is an *objective* (extensional) logical system, a system of reasoning about the world. In this respect, it is similar to classical logic, and distinct from modal (intensional) formalisms that deal primarily with beliefs and knowledge.

Yet another important feature of causal inference stems from the validity of the following rule:

(Coherence) If $A \Rightarrow \neg A$, then $A \Rightarrow \mathbf{f}$.

The above rule says that if a proposition causes propositions that are incompatible with it, then it is causally inconsistent. This feature indicates, in effect, that the above notion of causal inference is *atemporal*. For example, the rule $p \wedge q \Rightarrow \neg q$ cannot be understood as saying that p and q jointly cause $\neg q$ (afterwards) in a temporal sense; instead, by Coherence it implies $p \wedge q \Rightarrow \mathbf{f}$, which means, in effect, that $p \wedge q$ cannot hold. Just as in classical logic, however, a representation of temporal domains in this formalism can be obtained by adding explicit temporal arguments to propositions; this is what has been actually done in the action description framework of (Giunchiglia et al. 2004).

A possible worlds semantics

A semantic interpretation of causal inference relations can be given in terms of ordinary possible worlds (or Kripke) models (W, R, V), where W is a set of possible worlds, R a binary accessibility relation on W, and V a function assigning each world a propositional interpretation. Intuitively, $R\alpha\beta$ means that α is an initial state, and β a possible output state of a causal process. A Kripke model is *quasi-reflexive* if it satisfies the condition that if $R\alpha\beta$, then $R\alpha\alpha$.

Definition 4. A rule $A \Rightarrow B$ is *valid* in a Kripke model (W, R, V) if, for any worlds α, β such that $R\alpha\beta$, if A holds in α, then B holds in β.

By a set of causal rules *determined* by a Kripke model we will mean the set of rules that is valid in it. It can be verified that such a set satisfies all the postulates of causal inference. Moreover, as for other modal formalisms, a suitable construction of a canonical semantics allows us to obtain the corresponding completeness result:

Proposition 4. *A set of causal rules forms a causal inference relation if and only if it is determined by some quasi-reflexive Kripke model.*

As a by-product, the above semantics immediately sanctions a simple modal representation of causal rules. Namely, let $\Box$ be the usual modal operator definable in a possible worlds model: $\Box A$ holds in α iff A holds in all β such that $R\alpha\beta$. Then the validity of $A \Rightarrow B$ in a possible worlds model is equivalent to validity of the formula $A \rightarrow \Box B$. Consequently, causal rules are representable by modal formulas of the latter form. As a matter of fact, this modal representation has actually been used in many approaches to formalizing causation in action theories (see, e.g., (Geffner 1990; Turner 1999; Giordano, Martelli, and Schwind 2000; Zhang and Foo 2001)).

The nonmonotonic semantics of causal inference

Causal inference relations are just a special kind of causal theories, so they also possess a nonmonotonic semantics. Moreover, due to the logical properties of causal inference, the description of this nonmonotonic semantics can be simplified as follows.

To begin with, we extend causal rules to rules having arbitrary sets of propositions as premises: given a causal inference relation $\Rightarrow$ and an arbitrary set u of propositions, $u \Rightarrow A$ will be taken to hold if, for some finite $a \subseteq u$, $\bigwedge a \Rightarrow A$ belongs to $\Rightarrow$. $\mathcal{C}(u)$ will denote the set of propositions caused by u, that is $\mathcal{C}(u) = \{A \mid u \Rightarrow A\}$. Then a world α is an *exact world* of a causal inference relation if and only if

$$\alpha = \mathcal{C}(\alpha).$$

Given an arbitrary causal theory Δ, we will denote by $\Rightarrow_\Delta$ the least causal inference relation that includes Δ. It has been shown in (Bochman 2003) that Δ has the same nonmonotonic semantics as $\Rightarrow_\Delta$, which means that the rules of causal inference are adequate for causal reasoning with respect to the nonmonotonic semantics of causal theories. Moreover, it has been shown that causal inference relations constitute in this respect a maximal such logic (see (Bochman 2004) for details).

Dynamic Causal Inference

By a *dynamic causal inference relation* we will mean a set of dynamic causal rules of the form $A.B \Rightarrow C$ that satisfies the conditions described below.

The first group of postulates states that a set of dynamic causal rules with a fixed first premise (D) should satisfy the postulates of an 'ordinary' causal inference:

(Strengthening) If $A \vDash B$ and $D.B \Rightarrow C$, then $D.A \Rightarrow C$;

(Weakening) If $D.A \Rightarrow B$ and $B \vDash C$, then $D.A \Rightarrow C$;

(And) If $D.A \Rightarrow B$ and $D.A \Rightarrow C$, then $D.A \Rightarrow B \wedge C$;

(Or) If $D.A \Rightarrow C$ and $D.B \Rightarrow C$, then $D.A \vee B \Rightarrow C$;

(Cut) If $D.A \Rightarrow B$ and $D.A \wedge B \Rightarrow C$, then $D.A \Rightarrow C$;

(Truth) $\mathbf{t}.\mathbf{t} \Rightarrow \mathbf{t}$;

(Falsity) $\mathbf{t}.\mathbf{f} \Rightarrow \mathbf{f}$.

In view of the above postulates, dynamic causal rules $C.A \Rightarrow B$ can be seen as ordinary, binary causal rules $A \Rightarrow B$ that are conditioned by the preceding context C.

The next two postulates describe the logical properties of this preceding context in dynamic causal rules:

(Left-Str) If $A \vDash B$ and $B.D \Rightarrow C$, then $A.D \Rightarrow C$;

(Left-Or) If $A.D \Rightarrow C$ and $B.D \Rightarrow C$, then $A \vee B.D \Rightarrow C$.

The combined effect of the above pair of postulates is that the associated semantic interpretation of dynamic causal inference (described in the next section) will be again a kind of a possible world semantics, in which both the two premises and conclusion of a dynamic causal rule are evaluated with respect to worlds (complete states).

Finally, the last postulate is a formal expression of the requirement that any state that can be an input of some transition, is also an output state of at least one transition:

(Transition) If $\mathbf{t}.A \Rightarrow \mathbf{f}$, then $A.\mathbf{t} \Rightarrow \mathbf{f}$.

Recall that we have decided to identify static causal laws $A \Rightarrow B$ with dynamic causal laws of the form $\mathbf{t}.A \Rightarrow B$. Then the above postulate can be rewritten as

(Transition1) If $A \Rightarrow \mathbf{f}$, then $A.\mathbf{t} \Rightarrow \mathbf{f}$.

On this reformulation, the above postulate stipulates, in effect, that any input state of a consistent transition should be (statically) causally consistent. Combined with the other postulates, this will immediately imply that both the input and output state of a transition should be closed with respect to the valid static laws.

In the next section we will describe a possible-worlds semantics for the above logical formalism of dynamic inference.

A possible worlds semantics

A possible worlds semantics of dynamic causal relations can be obtained by generalizing an accessibility relation on possible worlds to ternary relations.

A *causal possible world model* of dynamic causal inference is a triple (W, R, V), where W is a set of possible worlds, R a ternary accessibility relation on W, and V a function assigning each world a propositional interpretation. The accessibility relation will be required to satisfy the following two conditions:

(Quasi-reflexivity) If $R\alpha\beta\gamma$, then $R\alpha\beta\beta$.

(Transition) If $R\alpha\beta\beta$, then $R\delta\alpha\alpha$, for some $\delta \in W$.

Definition 5. A rule $A.B \Rightarrow C$ is *valid* in a model (W, R, V) if, for any worlds α, β, γ such that $R\alpha\beta\gamma$, if A holds in α and B holds in β, then C holds in γ.

Given the above definition of validity, it is easy to verify the following

Lemma 5. *The set of dynamic causal rules valid in a causal possible world model forms a dynamic causal inference relation.*

Moreover, using a suitable construction of a canonical semantics for a dynamic causal inference relation, the following completeness result can be established:

Theorem 6. *A set of dynamic causal rules forms a dynamic causal inference relation if and only if it is determined by a causal possible world model.*

Proof. (A sketch) Due to the connection between dynamic causal rules and the original, atemporal causal rules, the proof is a relatively straightforward generalization of the corresponding completeness proof for causal inference relations, given in (Bochman 2004, Theorem 7.4). More precisely, given a dynamic causal relation $\Rightarrow$, we can construct the corresponding canonical model (W, R_c) by taking W to be the set of all maximal consistent sets of propositions, and defining R_c as follows[8]:

$$R_c\alpha\beta\gamma \;\equiv\; C(\alpha.\beta) \subseteq \beta \cap \gamma.$$

Notice that this definition directly implies quasi-reflexivity of R_c. Moreover, the use of the Transition postulate allows us to prove the transition property of R_c. Finally, it can be shown that $A.B \Rightarrow C$ holds for the source dynamic causal relation if and only if it is valid in (W, R_c). □

Remark. One of the interesting consequences of the above semantic characterization of dynamic causal inference is that, similarly to a straightforward modal translation of ordinary causal rules as formulas of the form $A \rightarrow \Box B$, dynamic causal rules can be represented as formulas of *arrow logic* (see, e.g., (Venema 1997)). As a matter of fact, one of the principal motivations behind arrow logic has also consisted in providing an abstract description of dynamic (transition) models (cf. (van Benthem 1994)). Moreover, semantic interpretation of arrow logic is also based on a possible world semantics with a ternary accessibility relation, and it can be easily verified that, by the above semantic description, a dynamic causal rule $A.B \Rightarrow C$ turns out to be equivalent to a formula

$$A \circ B \rightarrow C$$

of arrow logic, where $\circ$ is a binary 'arrow conjunction' operator having the following semantic interpretation: $A \circ B$ holds in a world α if and only if there are worlds β, γ such that $R\beta\gamma\alpha$, A holds in β and B holds in γ.

[8] See the next section for a definition of $C(\alpha.\beta)$.

Correspondences

Recall that a dynamic causal theory is an arbitrary set of dynamic causal rules. For any dynamic causal theory Δ there exists a least dynamic causal inference relation that includes Δ. We will denote it by $\Rightarrow_\Delta$. Clearly, $\Rightarrow_\Delta$ is the set of all dynamic causal rules that can be derived from Δ using the postulates of dynamic causal inference.

As before, we will extend the notation of dynamic causal rules to sets of propositional formulas in premises: for sets u, v of propositional formulas, $u.v \Rightarrow A$ will be taken to hold if $\bigwedge a. \bigwedge b \Rightarrow A$, for some finite $a \subseteq u, b \subseteq v$. In addition, $\mathcal{C}(u.v)$ will denote the set of propositions $\{A \mid u.v \Rightarrow A\}$.

Due to the logical properties of a dynamic causal inference relation, the definition of an exact transition can now be simplified, namely a pair of worlds (α, β) will be an *exact transition* with respect to a dynamic causal inference relation if and only if

$$\beta = \mathcal{C}(\alpha.\beta).$$

Then the following key result of this study shows, in effect, that the logic of causal dynamic inference is adequate for reasoning with respect to the exact semantics of dynamic causal theories, since it preserves the latter.

Theorem 7. *The exact transition models of a dynamic causal theory Δ coincide with the exact transition models of $\Rightarrow_\Delta$.*

Proof sketch. It can be shown that if C_Δ is the provability operator of $\Rightarrow_\Delta$, then, for any worlds α, β, if $C_\Delta(\alpha.\beta)$ is a consistent set, then it coincides with $\mathrm{Th}(\Delta(\alpha.\beta))$. Consequently, $\alpha = C_\Delta(\alpha.\beta)$ iff $\alpha = \mathrm{Th}(\Delta(\alpha.\beta))$, and therefore exact transitions of Δ will coincide with exact transitions of $\Rightarrow_\Delta$. Hence the result. □

Moreover, it can be shown that the logic of causal dynamic inference constitutes a maximal logic that is adequate for reasoning with exact causal models.

Definition 6. Two dynamic causal theories Δ and Γ are *strongly equivalent* if, for any set Φ of causal rules, $\Delta \cup \Phi$ has the same exact transition models as $\Gamma \cup \Phi$.

Strongly equivalent theories are 'equivalent forever', that is, they are interchangeable in any larger causal theory without changing the nonmonotonic semantics. Consequently, strong equivalence can be seen as an equivalence with respect to the background monotonic logic of causal theories. And the next result shows that this logic is precisely the logic of dynamic causal inference.

Theorem 8. *Dynamic causal theories Δ and Γ are strongly equivalent if and only if $\Rightarrow_\Delta = \Rightarrow_\Gamma$.*

The above result states that dynamic causal theories are strongly equivalent if and only if each of them can be obtained from the other using the postulates of dynamic causal inference.

Summary and Perspectives

The primary objective of this study consisted in showing that causal reasoning in dynamic action domains can be given a direct and concise logical representation. Moreover, being combined with the wealth of representation capabilities of such a reasoning, demonstrated in (Giunchiglia et al. 2004), the results of this study strongly indicate that a theory of dynamic causal inference can be viewed as a self-subsistent logical theory that provides an adequate and comprehensive representation framework for reasoning in dynamic domains. The study creates also obvious incentives for broader questions about the role and scope of causation in commonsense reasoning, as well as in knowledge representation in AI.

Causation has always been one of the most discussed concepts in the philosophy of science. It is intimately related to practically all notions that are essential both for a commonsense and scientific view of the world, such as laws, counterfactuals, explanation and abduction. On the other hand, causation and related notions have shown to be extremely elusive and problematic concepts. Efforts of many philosophers and logicians in the past have been focused on a formal, logical explication of these notions, but the task has turned out to be surprisingly difficult. Furthermore, starting with David Hume, an influential line of philosophical thought has argued, in effect, that causation should be expelled from the language of Science and Logic.

In recent years, however, we witness a revival of interest in the concept of causation, accompanied with new, more practical, insights about its role in our reasoning. Most prominent in this respect is Pearl's theory of causal reasoning (Pearl 2000) and its applications in statistics, economics, cognitive and social sciences.

An important alternative source of the new understanding of causation and its role in our reasoning comes from Artificial Intelligence, especially from theories of action and change. In these theories causation is a working concept allowing us to single out intended models of commonsense action descriptions. They make especially vivid the fact that causal reasoning, that is, asking why and seeking explanations, is germane to our reasoning about the world. However, these theories have also made evident that, though causal reasoning includes an important logical part, it is *not reducible* to a plain logical derivation in some ingenious causal logic. Instead, causal reasoning should be viewed as an important case of general nonmonotonic (assumption-based) reasoning. Accordingly, the tools and formalisms of nonmonotonic reasoning should hopefully provide us with a more adequate understanding of the concept of causation itself.

References

Beall, J.; Brady, R.; Dunn, J.; Hazen, A.; Mares, E.; Meyer, R.; Priest, G.; Restall, G.; Ripley, D.; Slaney, J.; and Sylvan, R. 2012. On the ternary relation and conditionality. *Journal of Philosophical Logic* 41(3):595–612.

Bochman, A. 2003. A logic for causal reasoning. In *IJCAI-03, Proceedings of the Eighteenth International Joint Con-*

ference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003, 141–146. Acapulco: Morgan Kaufmann.

Bochman, A. 2004. A causal approach to nonmonotonic reasoning. *Artificial Intelligence* 160:105–143.

Bochman, A. 2007. A causal theory of abduction. *Journal of Logic and Computation* 17:851–869.

Bochman, A. 2011. Logic in nonmonotonic reasoning. In Brewka, G.; Marek, V. W.; and Truszczynski, M., eds., *Nonmonotonic Reasoning. Essays Celebrating its 30th Anniversary*. College Publ. 25–61.

Bochman, A. 2013. Causation in flux. In *Commonsense-2013, 11th International Symposium on Logical Formalization of Commonsense Reasoning*.

Calvanese, D.; De Giacomo, G.; and Vardi, M. Y. 2002. Reasoning about actions and planning in LTL action theories. In *Proc. Eights Int. Conf. on Principles of Knowledge Representation and Reasoning (KR-02), Toulouse, France*, 593–602. Morgan Kaufmann.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*.

Geffner, H. 1990. Causal theories for nonmonotonic reasoning. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, 524–530. AAAI Press.

Gelfond, M., and Lifschitz, V. 1998. Action languages. *Electronic Transactions on Artificial Intelligence* 3:195–210.

Giordano, L.; Martelli, A.; and Schwind, C. 2000. Ramification and causality in a modal action logic. *J. Log. Comput.* 10(5):625–662.

Giunchiglia, E., and Lifschitz, V. 1995. Dependent fluents. In *Proceeedings IJCAI-1995*, 1964–1969.

Giunchiglia, E., and Lifschitz, V. 1998. An action language based on causal explanation: preliminary report. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, 623–630. AAAI Press.

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2004. Nonmonotonic causal theories. *Artificial Intelligence* 153:49–104.

Harel, D.; Kozen, D.; and Tiuryn, J. 2000. *Dynamic Logic*. MIT Press.

Lee, J.; Lifschitz, V.; and Yang, F. 2013. Action language BC: Preliminary report. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*.

Lin, F. 1995. Embracing causality in specifying the inderect effect of actions. In *Proc. Int. Joint Conf. on Artificial Intelligence, IJCAI-95*, 1985–1991. Montreal: Morgan Kaufmann.

Lin, F. 1996. Embracing causality in specifying the indeterminate effects of actions. In *Proceedings AAAI-96*, 670–676.

McCain, N., and Turner, H. 1997. Causal theories of action and change. In *Proceedings AAAI-97*, 460–465.

Myers, K. L., and Smith, D. E. 1988. The persistence of derived information. In *Proceedings AAAI-88*, 496–500.

Pearl, J. 2000. *Causality*. Cambridge UP.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*. Academic Press. 318–420.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamic Systems*. MIT Press.

Shanahan, M. P. 1997. *Solving the Frame Problem*. The MIT Press.

Turner, H. 1999. A logic of universal causation. *Artificial Intelligence* 113:87–123.

van Benthem, J. 1994. A note on dynamic arrow logic. In van Eijck, J., and Visser, A., eds., *Logic and Information Flow*. Cambridge, MA, USA: MIT Press. 15–29.

Venema, Y. 1997. A crash course in arrow logic. In Marx, M.; Pólos, L.; and Masuch, M., eds., *Arrow Logic and Multi-Modal Logic*. Stanford, CA, USA: Center for the Study of Language and Information. 3–34.

Zhang, D., and Foo, N. 2001. EPDL: A logic for causal reasoning. In *Proc Int. Joint Conf. on Artificial Intelligence, IJCAI-01*, 131–136. Seattle: Morgan Kaufmann.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Appropriate Causal Models and Stability of Causation

Joseph Y. Halpern*
Cornell University
halpern@cs.cornell.edu

Abstract

Causal models defined in terms of structural equations have proved to be quite a powerful way of representing knowledge regarding causality. However, a number of authors have given examples that seem to show that the Halpern-Pearl (HP) definition of causality (Halpern & Pearl 2005) gives intuitively unreasonable answers. Here it is shown that, for each of these examples, we can give two stories consistent with the description in the example, such that intuitions regarding causality are quite different for each story. By adding additional variables, we can disambiguate the stories. Moreover, in the resulting causal models, the HP definition of causality gives the intuitively correct answer. It is also shown that, by adding extra variables, a modification to the original HP definition made to deal with an example of Hopkins and Pearl (2003) may not be necessary. Given how much can be done by adding extra variables, there might be a concern that the notion of causality is somewhat unstable. Can adding extra variables in a "conservative" way (i.e., maintaining all the relations between the variables in the original model) cause the answer to the question "Is $X = x$ a cause of $Y = y$?" to alternate between "yes" and "no"? Here it is shown that adding an extra variable can change the answer from "yes' to "no", but after that, it cannot cannot change back to "yes".

1 Introduction

Causal models defined in terms of structural equations have proved to be quite a powerful way of representing knowledge regarding causality. For example, they have been used to find causes of errors in software (Beer *et al.* 2012) and have been shown to be useful in predicting human attributions of responsibility (Gerstenberg & Lagnado 2010; Lagnado, Gerstenberg, & Zultan 2013). However, a number of authors authors (Glymour *et al.* 2010; Livengood 2013; Spohn 2008; Weslake 2014) have given examples that seem to show that the Halpern-Pearl (HP) definition of causality (Halpern & Pearl 2005) gives intuitively unreasonable answers. One contribution of this paper is to show that these "problematic" examples can be dealt with in a relatively uniform way, by being a little more careful about the choice of causal model.

The need to choose the causal model carefully has been pointed out frequently (Blanchard & Schaffer 2013; Hall 2007; Halpern & Pearl 2005; Halpern & Hitchcock 2010; Hitchcock 2001; 2007). A causal model is characterized by the choice of variables, the equations relating them, and which variables we choose to make exogenous and endogenous (roughly speaking, which are the variables we choose to take as given and which we consider to be modifiable). Different choices of causal model for a given situation can lead to different conclusions regarding causality. The choices are, to some extent, subjective. While some suggestions have been made for good rules of thumb for choosing random variables (e.g., in (Halpern & Hitchcock 2010)), they are certainly not definitive. Moreover, the choice of variables may also depend in part on the variables that the modeler is aware of.

In this paper, I consider the choice of representation in more detail in four examples. I show that in all these examples, the model originally considered (which I call the "naive" model) does not correctly model all the relevant features of the situation. I argue that we can see this because, in all these cases, there is another story that can be told, also consistent with the naive model, for which we have quite different intuitions regarding causality, This suggests that a more careful model is needed to disambiguate the stories. In the first four cases, what turns out to arguably be the best way to do the disambiguation is to add (quite well motivated) extra variables, which, roughly speaking, capture the mechanism of causality. In the final example, what turns out to be most relevant is the decision as to which variables to make exogenous. Once we model things more carefully, the HP approach gives the expected answer in all cases.

As already observed by Halpern and Hitchcock (2014), adding extra variables also lets us deal with two other concerns that resulted in changes to the original HP definition. In Section 4, I consider an example due Hopkins and Pearl (2003) that motivated one of the changes. After showing

*Supported in part by NSF grants IIS-0911036 and CCF-1214844, AFOSR grant FA9550-08-1-0438 and by the DoD Multidisciplinary University Research Initiative (MURI) program administered by AFOSR under grant FA9550-12-1-0040. Thanks to Chris Hitchcock and Jonathan Livengood for interesting discussions and useful comments. I also thank Jonathan for a careful proofreading of the paper, which uncovered many typos. Finally, I think Thomas Blanchard for pointing out a serious problem in an earlier version of Theorem 5.1.
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

how this example can be dealt with by adding an extra variable in a natural way (without modifying the original HP definition), I show that this approach generalizes: we can always add extra variables so as to get a model where the original HP definition can be used. In the full paper, I discuss an example due to Hiddleston (2005) that motivated the addition of normality considerations to the basic HP framework (see Section 2). Again, adding an extra variable deals with this example.

All these examples show that adding extra variables can result in a cause becoming a non-cause. Can adding variables also result in a non-cause becoming a cause? Of course, without constraints, this can easily happen. Adding extra variables can fundamentally change the model. However, as I show, if variables are added in a conservative way (so as to maintain all the relations between the variables in the original model), this cannot happen. If $X = x$ is not a cause of $Y = y$, then adding extra variables to the model cannot make $X = x$ a cause of $Y = y$.

The rest of this paper is organized as follows. In the next section, I review the HP definition (and the original definition) and its extension to deal with normality, as discussed in (Halpern & Hitchcock 2014). I discuss the five examples in Section 3. In Section 4, I discuss how adding extra variables can deal with the Hopkins-Pearl example and, more generally, can obviate the need to modify the original HP definition. I prove that non-causality is stable in Section 5. I conclude in Section 6 with some discussion of the implications of these results. Details of some proofs are left to the full paper (available at www.cs.cornell.edu/home/halpern/papers/causalmodeling.pdf).

2 Review

In this section, I briefly review the definitions of causal structures, the HP definition(s) of causality, and the extension that takes into account normality given by Halpern and Hitchcock. The exposition is largely taken from (Halpern 2008). The reader is encouraged to consult (Halpern & Pearl 2005), and (Halpern & Hitchcock 2014) for more details and intuition.

2.1 Causal structures

The HP approach assumes that the world is described in terms of random variables and their values. Some random variables may have a causal influence on others. This influence is modeled by a set of *structural equations*. It is conceptually useful to split the random variables into two sets: the *exogenous* variables, whose values are determined by factors outside the model, and the *endogenous* variables, whose values are ultimately determined by the exogenous variables. For example, in a voting scenario, we could have endogenous variables that describe what the voters actually do (i.e., which candidate they vote for), exogenous variables that describe the factors that determine how the voters vote, and a variable describing the outcome (who wins). The structural equations describe how the outcome is determined (e.g., majority rules).

Formally, a *causal model* M is a pair $(\mathcal{S}, \mathcal{F})$, where $\mathcal{S}$ is a *signature*, which explicitly lists the endogenous and exogenous variables and characterizes their possible values, and $\mathcal{F}$ defines a set of *modifiable structural equations*, relating the values of the variables. A signature $\mathcal{S}$ is a tuple $(\mathcal{U}, \mathcal{V}, \mathcal{R})$, where $\mathcal{U}$ is a set of exogenous variables, $\mathcal{V}$ is a set of endogenous variables, and $\mathcal{R}$ associates with every variable $Y \in \mathcal{U} \cup \mathcal{V}$ a nonempty set $\mathcal{R}(Y)$ of possible values for Y (that is, the set of values over which Y *ranges*). For simplicity, I assume here that $\mathcal{V}$ is finite, as is $\mathcal{R}(Y)$ for every endogenous variable $Y \in \mathcal{V}$. $\mathcal{F}$ associates with each endogenous variable $X \in \mathcal{V}$ a function denoted F_X such that $F_X : (\times_{U \in \mathcal{U}} \mathcal{R}(U)) \times (\times_{Y \in \mathcal{V} - \{X\}} \mathcal{R}(Y)) \rightarrow \mathcal{R}(X)$. This mathematical notation just makes precise the fact that F_X determines the value of X, given the values of all the other variables in $\mathcal{U} \cup \mathcal{V}$. If there is one exogenous variable U and three endogenous variables, X, Y, and Z, then F_X defines the values of X in terms of the values of Y, Z, and U. For example, we might have $F_X(u, y, z) = u + y$, which is usually written as $X = U + Y$.[1] Thus, if $Y = 3$ and $U = 2$, then $X = 5$, regardless of how Z is set.

The structural equations define what happens in the presence of external interventions. Setting the value of some variable X to x in a causal model $M = (\mathcal{S}, \mathcal{F})$ results in a new causal model, denoted $M_{X=x}$, which is identical to M, except that the equation for X in $\mathcal{F}$ is replaced by $X = x$.

Following (Halpern & Pearl 2005), I restrict attention here to what are called *recursive* (or *acyclic*) models. This is the special case where there is some total ordering $\prec$ of the endogenous variables (the ones in $\mathcal{V}$) such that if $X \prec Y$, then X is independent of Y, that is, $F_X(\ldots, y, \ldots) = F_X(\ldots, y', \ldots)$ for all $y, y' \in \mathcal{R}(Y)$. Intuitively, if a theory is recursive, there is no feedback. If $X \prec Y$, then the value of X may affect the value of Y, but the value of Y cannot affect the value of X. It should be clear that if M is an acyclic causal model, then given a *context*, that is, a setting $\vec{u}$ for the exogenous variables in $\mathcal{U}$, there is a unique solution for all the equations. We simply solve for the variables in the order given by $\prec$. The value of the variables that come first in the order, that is, the variables X such that there is no variable Y such that $Y \prec X$, depend only on the exogenous variables, so their value is immediately determined by the values of the exogenous variables. The values of variables later in the order can be determined once we have determined the values of all the variables earlier in the order.

2.2 A language for reasoning about causality

To define causality carefully, it is useful to have a language to reason about causality. Given a signature $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$, a *primitive event* is a formula of the form $X = x$, for $X \in \mathcal{V}$ and $x \in \mathcal{R}(X)$. A *causal formula (over $\mathcal{S}$)* is one of the form $[Y_1 \leftarrow y_1, \ldots, Y_k \leftarrow y_k]\varphi$, where φ is a Boolean combination of primitive events, $Y_1, \ldots, Y_k$ are distinct variables in $\mathcal{V}$, and $y_i \in \mathcal{R}(Y_i)$. Such a formula is abbreviated as $[\vec{Y} \leftarrow \vec{y}]\varphi$. The special case where $k = 0$ is abbreviated as

[1] The fact that X is assigned $U + Y$ (i.e., the value of X is the sum of the values of U and Y) does not imply that Y is assigned $X - U$; that is, $F_Y(U, X, Z) = X - U$ does not necessarily hold.

φ. Intuitively, $[Y_1 \leftarrow y_1, \ldots, Y_k \leftarrow y_k]\varphi$ says that φ would hold if Y_i were set to y_i, for $i = 1, \ldots, k$.

A causal formula ψ is true or false in a causal model, given a context. As usual, I write $(M, \vec{u}) \models \psi$ if the causal formula ψ is true in causal model M given context $\vec{u}$. The $\models$ relation is defined inductively. $(M, \vec{u}) \models X = x$ if the variable X has value x in the unique (since we are dealing with acyclic models) solution to the equations in M in context $\vec{u}$ (that is, the unique vector of values for the exogenous variables that simultaneously satisfies all equations in M with the variables in $\mathcal{U}$ set to $\vec{u}$). The truth of conjunctions and negations is defined in the standard way. Finally, $(M, \vec{u}) \models [\vec{Y} \leftarrow \vec{y}]\varphi$ if $(M_{\vec{Y}=\vec{y}}, \vec{u}) \models \varphi$. I write $M \models \varphi$ if $(M, \vec{u}) \models \varphi$ for all contexts $\vec{u}$.

2.3 The definition(s) of causality

The HP definition of causality, like many others, is based on counterfactuals. The idea is that A is a cause of B if, if A hadn't occurred (although it did), then B would not have occurred. But there are many examples showing that this naive definition will not quite work. To take just one example, consider the following story, due to Ned Hall and already discussed in (Halpern & Pearl 2005), from where the following version is taken.

> Suzy and Billy both pick up rocks and throw them at a bottle. Suzy's rock gets there first, shattering the bottle. Since both throws are perfectly accurate, Billy's would have shattered the bottle had it not been preempted by Suzy's throw.

We would like to say that Suzy's throw is a cause of the bottle shattering, and Billy's is not. But if Suzy hadn't thrown, Billy's rock would have hit the bottle and shattered it.

The HP definition of causality is intended to deal with this example, and many others.

Definition 2.1: $\vec{X} = \vec{x}$ is an *actual cause of* φ *in* $(M, \vec{u})$ if the following three conditions hold:

AC1. $(M, \vec{u}) \models (\vec{X} = \vec{x})$ and $(M, \vec{u}) \models \varphi$.

AC2. There is a partition of $\mathcal{V}$ (the set of endogenous variables) into two subsets $\vec{Z}$ and $\vec{W}$[2] with $\vec{X} \subseteq \vec{Z}$ and a setting $\vec{x}'$ and $\vec{w}$ of the variables in $\vec{X}$ and $\vec{W}$, respectively, such that if $(M, \vec{u}) \models Z = z$ for all $Z \in \vec{Z}$ (i.e., z is the value of the random variable Z in the real world), then both of the following conditions hold:

- (a) $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}]\neg\varphi$.
- (b) $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W}' \leftarrow \vec{w}, \vec{Z}' \leftarrow \vec{z}]\varphi$ for all subsets $\vec{W}'$ of $\vec{W}$ and all subsets $\vec{Z}'$ of $\vec{Z}$, where I abuse notation and write $\vec{W}' \leftarrow \vec{w}$ to denote the assignment where the variables in $\vec{W}'$ get the same values as they would in the assignment $\vec{W} \leftarrow \vec{w}$, and similarly for $\vec{Z}' \leftarrow \vec{z}$.

AC3. $\vec{X}$ is minimal; no subset of $\vec{X}$ satisfies conditions AC1 and AC2.

The tuple $(\vec{W}, \vec{w}, \vec{x}')$ is said to be a *witness* to the fact that $\vec{X} = \vec{x}$ is a cause of φ.

AC1 just says that $\vec{X} = \vec{x}$ cannot be considered a cause of φ unless both $\vec{X} = \vec{x}$ and φ actually happen. AC3 is a minimality condition, which ensures that only those elements of the conjunction $\vec{X} = \vec{x}$ that are essential are considered part of a cause; inessential elements are pruned. Without AC3, if dropping a lit cigarette is a cause of a fire then so is dropping the cigarette and sneezing.

AC2 is the core of the definition. We can think of the variables in $\vec{Z}$ as making up the "causal path" from $\vec{X}$ to φ. Intuitively, changing the value of some variable in X results in changing the value(s) of some variable(s) in $\vec{Z}$, which results in the values of some other variable(s) in $\vec{Z}$ being changed, which finally results in the value of φ changing. The remaining endogenous variables, the ones in $\vec{W}$, are off to the side, so to speak, but may still have an indirect effect on what happens. AC2(a) is essentially the standard counterfactual definition of causality, but with a twist. If we want to show that $\vec{X} = \vec{x}$ is a cause of φ, we must show (in part) that if $\vec{X}$ had a different value, then so too would φ. However, this effect of the value of $\vec{X}$ on the value of φ may not hold in the actual context; essentially, it ensures that $\vec{X}$ alone suffices to bring about the change from φ to $\neg\varphi$; setting $\vec{W}$ to $\vec{w}$ merely eliminates possibly spurious side effects that may mask the effect of changing the value of $\vec{X}$. Moreover, although the values of variables on the causal path (i.e., the variables $\vec{Z}$) may be perturbed by the change to $\vec{W}$, this perturbation has no impact on the value of φ. As I said above, if $(M, \vec{u}) \models \vec{Z} = \vec{z}$, then z is the value of the variable Z in the context $\vec{u}$ (i.e., in the actual situation). We capture the fact that the perturbation has no impact on the value of φ by saying that if some variables Z on the causal path were set to their original values in the context $\vec{u}$, φ would still be true, as long as $\vec{X} = \vec{x}$. Roughly speaking, it says that if the variables in $\vec{X}$ are reset to their original value, then φ holds, even under the contingency $\vec{W}' = \vec{w}$ and even if some variables in $\vec{Z}$ are given their original values (i.e., the values in $\vec{z}$). The fact that AC2(b) must hold even if only a subset $\vec{W}'$ of the variables in $\vec{W}$ are set to their values in $\vec{w}$ (so that the variables in $\vec{W} - \vec{W}'$ essentially act as they do in the real world) and only a subset of the variables in $\vec{Z}$ are set to their values in the actual world says that we must have φ even if some things happen as they do in the actual world. See Sections 3.1 and 4 for further discussion of and intuition for AC2(b).

The original HP paper (Halpern & Pearl 2001) used a weaker version of AC2(b). Rather than requiring that $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W}' \leftarrow \vec{w}, \vec{Z}' \leftarrow \vec{z}]\varphi$ for all subsets $\vec{W}'$ of $\vec{W}$, it was required to hold only for $\vec{W}$. That is, the following condition was used instead of AC2(b).

AC2(b′) $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W} \leftarrow \vec{w}, \vec{Z}' \leftarrow \vec{z}]\varphi$ for all

[2] I occasionally use the vector notation ($\vec{Z}$, $\vec{W}$, etc.) to denote a set of variables if the order of the variables matters, which it does when we consider an assignment such as $\vec{W} \leftarrow \vec{w}$.

subsets $\vec{Z}'$ of $\vec{Z}$.

The change from AC2(b′) to AC2(b) may seem rather technical, but it has some nontrivial consequences. One of the contributions of this paper is to examine whether it is necessary; see Section 4 for details.

To deal with other problems in the HP definition, various authors have added the idea of *normality* to the definition. This can be done in a number of ways. I now briefly sketch one way that this can be done, following the approach in (Halpern & Hitchcock 2014).

Take a *world* (in a model M) to be a complete assignment of values to all the endogenous variables in M.[3] We assume a partial preorder $\succeq$ on worlds, that is, a reflexive transitive relation.[4] Intuitively, if $s \succeq s'$, then s is at least as normal, or typical, as s'. We can use normality in the definition of causality in two ways. Say that a world s is a *witness world* for $\vec{X} = \vec{x}$ being a cause of φ in $(M, \vec{u})$ if there is a witness $(\vec{W}, \vec{w}, \vec{x}')$ to $\vec{X} = \vec{x}$ being a cause of φ and $s = s_{\vec{X}=\vec{x}',\vec{W}=\vec{w},\vec{u}}$, where $s_{\vec{X}=\vec{x}',\vec{W}=\vec{w},\vec{u}}$ is the world that results by setting $\vec{X}$ to $\vec{x}'$ and $\vec{W}$ to $\vec{w}$ in context $\vec{u}$. We can then modify AC2(a) so as to require that we consider $\vec{X} = \vec{x}$ to be a cause of φ in $(M, \vec{u})$ only if the witness world s for $\vec{X} = \vec{x}$ being a cause is such that $s \succeq s_{\vec{u}}$, where $s_{\vec{u}}$ is the world determined by context $\vec{u}$. This says that, in determining causality, we consider only possibilities that result from altering atypical features of a world to make them more typical, rather than vice versa. This captures an observation made by Kahneman and Miller (1986) regarding human ascriptions of causality.

A somewhat more refined use of normality is to use it to "grade" causes. Say that s is a *best witness* for $\vec{X} = \vec{x}$ being a cause of φ if s is a witness world for $\vec{X} = \vec{x}$ being a cause of φ and there is no other witness world s' for $\vec{X} = \vec{x}$ being a cause of φ such that $s' \succ s$. (Note that there may be more than one best witness.) We can then grade candidate causes according to the normality of their best witnesses (without requiring that there must be a witness s such that $s \succeq s_{\vec{u}}$). Experimental evidence suggests that people are focusing on the cause with the best witness (according to their subjective ordering on worlds); see, e.g., (Cushman, Knobe, & Sinnott-Armstrong 2008; Hitchcock & Knobe 2009; Knobe & Fraser 2008).

3 The Examples

In this section, I consider examples due to Spohn (2008), Glymour et al. (2010), and Livengood (2013). I go through these examples in turn. I set the scene by considering the rock-throwing example mentioned above.

[3] In (Halpern & Hitchcock 2014), a world is defined as a complete assignment of values to the *exogenous* variables, but this is a typo.

[4] $\succeq$ is not necessarily a partial order; in particular, it does not necessarily satisfy *antisymmetry* (i.e., $s \succeq s'$ and $s' \succeq s$ does not necessarily imply $s = s'$).

3.1 Throwing rocks at bottles

A naive model of the rock-throwing story just has binary random variables *ST*, *BT*, and *BS* (for "Suzy throws", "Billy throws", and "bottle shatters", with the value of *BS* given by the equation $BS = ST \vee BT$: the bottle shatters if one of Suzy and Billy throws. Call this model M_{RT}. Let u be the context that results in $ST = BT = 1$: Suzy and Billy both throw. M_{RT} is described in Figure 1. (Although I have included the exogenous variable here, in later figures exogenous variables are omitted for ease of presentation.)

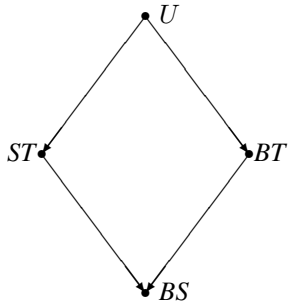


Figure 1: M_{RT}: the naive rock-throwing model.

As already pointed out by Halpern and Pearl (2005), in M_{RT} Suzy and Billy play completely symmetric roles. Not surprisingly, both $ST = 1$ and $BT = 1$ are causes of $BS = 1$ according to the HP definition. Clearly, M_{RT} cannot be used to distinguish a situation where Suzy is a cause from one where Billy is a cause.

In the story as given, people seem to agree that Suzy's throw is a cause and Billy's throw is not, since Suzy's rock hit the bottle and Billy's did not. M_{RT} does not capture this fact. Following Halpern and Pearl (2005), we extend M_{RT} so that it can express the fact that Suzy's rock hit first by adding two more variables:

- *BH* for "Billy's rock hits the (intact) bottle", with values 0 (it doesn't) and 1 (it does); and
- *SH* for "Suzy's rock hits the bottle", again with values 0 and 1.

The equations are such that $SH = ST$ (Suzy's rock hits the bottle if Suzy throws), $BH = BT \wedge \neg SH$ (Billy's rock hits an intact bottle if Billy throws and Suzy's rock does not hit), and $BS = SH \vee BH$ (the bottle shatters if either Suzy's rock or Billy's rock hit it). Now if Suzy and Bill both throw ($ST = 1$ and $BT = 1$), Suzy's rock hits the bottle ($SH = 1$), so that Billy's rock does not hit an intact bottle ($BH = 0$). Call the resulting model M'_{RT}. M'_{RT} is described in Figure 2 (with the exogenous variable omitted).

In this model $BT = 1$ is *not* a cause. For example, if we take $\vec{Z} = \{BT, BH, BS\}$ in AC2 and set $ST = 0$, then while it is the case that $BS = 0$ if $BT = 0$ and $BS = 1$ if $BT = 1$, it is not the case that $BT = 1$ if we set *BH* to its original value of 0. This example shows the necessity of allowing some variables in $\vec{Z}$ to be set to their original values in AC2(b). This captures the intuition that Billy's throw is not a cause because, in the actual world, his rock did not hit the bottle

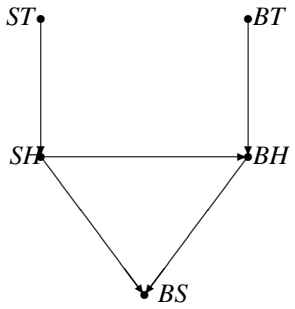


Figure 2: M'_{RT}: the better rock-throwing model.

($BH = 0$). By AC2(b), to establish $BT = 1$ as a cause of $BS = 1$, it would have to force $BS = 1$ even if $BH = 0$, which is not the case.

3.2 Spohn's example

The next example is due to Spohn (2008).

Example 3.1: There are four endogenous binary variables, A, B, C, and S, taking values 1 (on) and 0 (off). Intuitively, A and B are supposed to be alternative causes of C, and S acts as a switch. If $S = 0$, the causal route from A to C is active and that from B to C is dead; and if $S = 1$, the causal route from A to C is dead and the one from B to C is active. There are no causal relations between A, B, and S; their values are determined by the context. The equation for C is $C = (\neg S \wedge A) \vee (S \wedge B)$.

Suppose that the context is such that $A = B = S = 1$, so $C = 1$. The HP definition yields $B = 1$ and $S = 1$ as causes of $C = 1$, as we would hope. But, unfortunately, it also yields $A = 1$ as a cause of $C = 1$. The argument is that in the contingency where S is set to 0, then if $A = 0$, then $C = 0$, while if $A = 1$, then $C = 1$. This does not seem so reasonable. Intuitively, if $S = 1$, then the value of A seems irrelevant to the outcome. Considerations of normality do not help here; all worlds seem to be equally normal.

But now consider a slightly different story. This time, we view B as the switch, rather than S. If $B = 1$, then $C = 1$ if either $A = 1$ or $S = 1$; if $B = 0$, then $C = 1$ only if $A = 1$ and $S = 0$. That is, $C = (B \wedge (A \vee S)) \vee (\neg B \wedge A \wedge \neg S)$. Although this is perhaps not as natural a story as the original, such a switch is surely implementable. In any case, a little playing with propositional logic shows that, in this story, C satisfies exactly the same equation as before: $(\neg S \wedge A) \vee (S \wedge B)$ is equivalent to $(B \wedge (A \vee S)) \vee (\neg B \wedge A \wedge \neg S)$. The key point is that, unlike the first story, in the second story, it seems to me quite reasonable to say that $A = 1$ is a cause of $C = 1$ (as are $S = 1$ and $B = 1$). Having $A = 1$ is necessary for the first "mechanism" to work.

Given that we have different causal intuitions for the stories, we should model them differently. One way to distinguish them is to add two more endogenous random variables, say D and E, that describe the ways that C could be 1. In Spohn's original story, we would have the equation $D = \neg S \wedge A$, $E = S \wedge B$, and $C = D \vee E$. In this model, since $D = 0$ in the actual context, it is not hard to see that $A = 1$ is *not* a cause of $C = 1$, while $B = 1$ and $S = 1$ are, as they should be. Thus, in this model, we correctly capture our intuitions for the story.

To capture the second story, we can add variables D' and E' such that $D' = B \wedge (A \vee S)$, $E' = \neg B \wedge A \wedge S$, and $C = D' \vee E'$. In this model, it is not hard to see that all of $A = 1$, $B = 1$, and $S = 1$ are causes of $C = 1$.

This approach of adding extra variables leads to an obvious question: What is the role of these variables? I view D and E (resp., D' and E') as "structuring" variables, that help an agent "structure" a causal story. Consider Spohn's original story. We can certainly design a circuit where there is a source of power at A and B, a physical switch at S, and a bulb at C that turns on ($C = 1$) if either there is a battery at A ($A = 1$) and the switch is turned left ($S = 0$) or there is battery at B ($B = 1$) and the switch is turned right ($S = 1$). In this physical setup, there is no analogue of D and E. Nevertheless, to the extent that we view the models as a modeler's description of what is going, a modeler could usefully introduce D and E to describe the conditions under which $C = 1$, and to disambiguate this model from one where, conceptually, we might want to think of other ways that C could be 1 (as in the story with D' and E').

Note that we do not want to think of D as being *defined* to take the value 1 if $A = 1$ and $S = 0$. For then we could not intervene to set $D = 0$ if $A = 1$ and $S = 0$. Adding a variable to the model commits us to be able to intervene on it.[5] In the real world, setting D to 0 despite having $A = 1$ and $S = 0$ might correspond to the connection being faulty when the switch is turned left. Indeed, since the equation for C is the same in both stories, it is only at the level of interventions that the difference between the two stories becomes meaningful. ∎

3.3 Glymour et al.'s example

The next example is due to Glymour et al. (2010).

Example 3.2: A ranch has five individuals: $a_1, \ldots, a_5$. They have to vote on two possible outcomes: staying around the campfire ($O = 0$) or going on a round-up ($O = 1$). Let A_i be the random variable denoting a_i's vote, so $A_i = j$ if a_i votes for outcome j. There is a complicated rule for deciding on the outcome. If a_1 and a_2 agree (i.e., if $A_1 = A_2$), then that is the outcome. If $a_2, \ldots, a_5$ agree, and a_1 votes differently, then then outcome is given by a_1's vote (i.e., $O = A_1$). Otherwise, majority rules. In the actual situation, $A_1 = A_2 = 1$ and $A_3 = A_4 = A_5 = 0$, so by the first mechanism, $O = 1$. The question is what were the causes of $O = 1$.

Using the naive causal model with just the variables $A_1, \ldots, A_5, O$, and the obvious equations describing O in terms of $A_1, \ldots, A_5$, it is almost immediate that $A_1 = 1$ is a cause of $O = 1$. Changing A_1 to 0 results in $O = 0$. Somewhat surprisingly, in this naive model, $A_2 = 1$, $A_3 = 0$, $A_4 = 0$, and $A_5 = 0$ are also causes.[6] To see that $A_2 = 1$

[5] I thank Chris Hitchcock for stressing this point.

[6] Glymour et al. point out that $A_1 = 1$, $A_3 = 0$, $A_4 = 0$, and $A_5 = 0$ are causes; they do not mention that $A_2 = 1$ is also a

is a cause, consider the contingency where $A_3 = 1$. Now if $A_2 = 0$, then $O = 0$ (majority rules); if $A_2 = 1$, then $O = 1$, since $A_1 = A_2 = 1$, and $O = 1$ even if A_3 is set back to its original value of 0. To see that $A_3 = 0$ is a cause, consider the contingency where $A_2 = 0$, so that all voters but a_1 vote for 0 (staying at the campsite). If $A_3 = 1$, then $O = 0$ (majority rules). If $A_3 = 0$, then $O = 1$, by the second mechanism (a_1 is the only vote for 0), while if A_2 is set to its original value of 1, then we still have $O = 1$, now by the second mechanism.

But all this talk of mechanisms (which is also implicit in Glymour et al. (2010); in footnote 11, they say that setting A_2 back to its original value of 1 "brings out the original result, but in a different way") suggests that the mechanism should be part of the model. There are several ways of doing this. One is to add three new variables, call them M_1, M_2, and M_3. These variables have values in $\{0, 1, 2\}$, where $M_j = 0$ if mechanism j is active and suggests an outcome 0, $M_j = 1$ if mechanism j is active and suggests an outcome of 1, and $M_j = 2$ if mechanism j is not active. (We actually don't need the value $M_3 = 2$; mechanism 3 is always active, because there is always a majority with 5 voters all of whom must vote.) Note that at most one of the first two mechanisms can be active. We have obvious equations linking the value of M_1, M_2, and M_3 to the values of $A_1, \ldots, A_5$.

Now the value of O just depends on the values of M_1, M_2, and M_3: if $M_1 \neq 2$, then $O = M_1$; if $M_2 \neq 2$, then $O = M_2$, and if $M_1 = M_2 = 2$, then $O = M_3$. It is easy to see that in this model, if $A_1 = A_2 = 1$ and $A_3 = A_4 = A_5 = 0$, then none of $A_2 = 1$, $A_3 = 0$, $A_4 = 0$, and $A_5 = 0$ are causes. $A_1 = 1$ is cause, as we would expect, as is $M_2 = 1$, which seems reasonable: the second mechanism was the one that resulted in the outcome.

Now suppose that we change the description of the voting rule. We take $O = 1$ if one of the following two mechanisms applies:

- $A_1 = 1$ and it is not the case that both $A_2 = 0$ and exactly one of A_3, A_4, and A_5 is 1.
- $A_1 = 0$, $A_2 = 1$, and exactly two of A_3, A_4, and A_5 are 1.

It is not hard to check that, although the description is different, O satisfies the same equation in both stories. But now it does not seem so unreasonable that $A_2 = 1$, $A_3 = 0$, $A_4 = 0$, and $A_5 = 0$ are causes of $O = 1$. And indeed, if we construct a model in terms of these two mechanisms (i.e., add variables M_1' and M_2' that correspond to these two mechanisms), then it is not hard to see that $A_1 = 1$, $A_2 = 1$, $A_3 = 0$, $A_4 = 0$, and $A_5 = 0$ are all causes.

Here the role of the structuring variables M_1, M_2, and M_3 (resp. M_1' and M_2') as descriptors of the mechanism being invoked seems particularly clear. For example, setting $M_1 = 2$ says that the first mechanism will not be applied, even if $A_1 = A_2$; setting $M_1 = 1$ says that we act as if both a_1 and a_2 voted in favor, even if that is not the case. ∎

cause.

3.4 Livengood's voting examples

As Livengood (2013) points out, voting can lead to some apparently unreasonable causal outcomes (at least, if we model things naively). He first considers Jack and Jill, who live in an overwhelmingly Republican district. As expected, the Republican candidate wins with an overwhelming majority. Jill would normally have voted Democrat, but did not vote because she was disgusted by the process. Jack would normally have voted Republican, but did not vote because he (correctly) assumed that his vote would not affect the outcome. In the naive model, both Jack and Jill are causes of the Republican victory. For if enough of the people who voted Republican had switched to voting Democrat, then if Jack (or Jill) had voted Democrat, the Democrat would have won, while he would not have won had they abstained. Notice that, in this argument, Jack and Jill are treated the same way; their preferences make no difference.

We can easily construct a model that takes these preferences into account. One way to do so is to assume that their preferences are so strong that we may as well take them for granted. Thus, the preferences become exogenous; the only endogenous variables are whether or not they vote. In this case, Jack's not voting is not a cause of the outcome, but Jill's not voting is.

More generally, with this approach, a voter whose preference is made exogenous and is a strong supporter of the victor does not count as a cause of victory. This does not seem so unreasonable. After all, in an analysis of a close political victory in Congress, when an analyst talks about the cause(s) of victory, she points to the swing voters who voted one way or the other, not the voters that were taken to be staunch supporters of one particular side.

That said, making a variable exogenous seems like a somewhat draconian solution to the problem. It also does not allow us to take into account smaller gradations in depth of feeling. At what point should a preference switch from being endogenous to exogenous? We can achieve the same effect in an arguably more natural way by using normality considerations. In the case of Jack and Jill, we can take voting for a Democrat to be highly abnormal for Jack, and voting for a Republican to be highly abnormal for Jill. To show that either Jack (resp., Jill) is a cause of the victory, we need to consider a contingency where Jack (resp., Jill) votes for the Democratic candidate. This would be a change to a highly abnormal world in the case of Jack, but to a more normal world in the case of Jill. Thus, if we use normality as a criterion for determining causality, Jill would count as a cause, but Jack would not. If we use normality as a way of grading causes, Jack and Jill would still both count as causes for the victory, but Jill would be a much better cause. More generally, the more normal it would be for someone to vote Democratic, the better a cause that voter would be. The use of normality here allows for a more nuanced gradation of cause than the rather blunt approach of either making a variable exogenous or endogenous.

Now, following Livengood (2013), consider a vote where everyone can either vote for one of three candidates. Suppose that the actual vote is 17–2–0 (i.e., 17 vote for candidate A, 2 for candidate B, and none for candidate C). Then

not only is every vote for candidate A a cause of A winning, every vote for B is also a cause of A winning. To see this, consider a contingency where 8 of the voters for A switch to C. Then if one of the voters for B votes for C, the result is a tie; if that voter switches back to B, then A wins (even if some subset of the voters who switch from A to C switch back to A).

Is this reasonable? What makes it seem particularly unreasonable is that if it had just been a contest between A and B, with the vote 17–2, then the voters for B would not have been causes of A winning. Why should adding a third option make a difference?

In some cases it does seem reasonable that adding a third option makes a difference. For example, we speak of Nader costing Gore a victory over Bush in the 2000 election. But, as Livengood (2013) points out, we don't speak of Gore costing Nader a victory, although in a naive HP model of the situation, all the voters for Gore are causes of Nader not winning as much as the voters for Nader are causes of Gore not winning. The discussion above points a way out of this dilemma. If a sufficiently large proportion of Bush and Gore voters are taken to be such strong supporters that they will never change their minds, and we make their votes exogenous, then it is still the case that Nader caused Gore to lose, but not the case that Gore caused Nader to lose. Similar considerations apply in the case of the 17–2 vote. (Again, we can use normality considerations to give arguably more natural models of these examples.)[7]

4 Do we need AC2(b)?

In this section, I consider the extent to which we can use AC2(b$'$) rather than AC2(b), and whether this is a good thing.

4.1 The Hopkins-Pearl example

I start by examining the Hopkins-Pearl example that was intended to show that AC2(b$'$) was inappropriate. The following description is taken from (Halpern & Pearl 2005).

Example 4.1: Suppose that a prisoner dies either if A loads B's gun and B shoots, or if C loads and shoots his gun. Taking D to represent the prisoner's death and making the obvious assumptions about the meaning of the variables, we have that $D = (A \wedge B) \vee C$. Suppose that in the actual context u, A loads B's gun, B does not shoot, but C does load and shoot his gun, so that the prisoner dies. That is, $A = 1$, $B = 0$, and $C = 1$. Clearly $C = 1$ is a cause of $D = 1$. We would not want to say that $A = 1$ is a cause of $D = 1$, given that B did not shoot (i.e., given that $B = 0$). However, suppose that we take the obvious model with the random variables A, B, C, D. With AC2(b$'$), $A = 1$ is a cause of $D = 1$. For we can take $\vec{W} = \{B, C\}$ and consider the contingency where $B = 1$ and $C = 0$. It is easy to check that AC2(a) and AC2(b$'$) hold for this contingency, so under the original HP definition, $A = 1$ is a cause of $D = 1$. However, AC2(b) fails in this case, since $(M, u) \models [A \leftarrow 1, C \leftarrow 0](D = 0)$. The key point is that AC2(b) says that for $A = 1$ to be a cause of $D = 1$, it must be the case that $D = 0$ if only some of the values in $\vec{W}$ are set to $\vec{w}$. That means that the other variables get the same value as they do in the actual context; in this case, by setting only A to 1 and leaving B unset, B takes on its original value of 0, in which case $D = 0$. AC2(b$'$) does not consider this case.

Nevertheless, as pointed out by Halpern and Hitchcock (2014), we can use AC2(b$'$) if we have the "right" model. Suppose that we add a new variable E such that $E = A \wedge B$, so that $E = 1$ iff $A = B = 1$, and set $D = E \vee C$. Thus, we have captured the intuition that there are two ways that the prisoner dies. Either C shoots, or A loads and B fires (which is captured by E). It is easy to see that (using either AC2(b) or AC2(b$'$)) $B = 0$ is *not* a cause of $D = 1$. ∎

As I now show, the ideas of this example generalize. But before doing that, I define the notion of a conservative extension.

4.2 Conservative extensions

In the rock-throwing example, adding the extra variables converted $BT = 1$ from being a cause to not being a cause of $BS = 1$. Similarly, adding extra variables affected causality in all the other examples above. Of course, without any constraints, it is easy to add variables to get any desired result. For example, consider the rock-throwing model M'_{RT}. Suppose that we add a variable BH_1 with equations that set $BH_1 = BT$ and $BS = SH \vee BT \vee BH_1$. This results in a new "causal path" from BT to BS going through BH_1, independent of all other paths. Not surprisingly, in this model, $BT = 1$ is indeed a cause of $BS = 1$.

But this seems like cheating. What we want is a model where the new variable(s) are added in a way that extends what we know about the old variables. Intuitively, suppose that we had a better magnifying glass and could look more carefully at the model. We might discover new variables that were previously hidden. But we want it to be the case that any setting of the old variables results in the same observations. This is made precise in the following definition.

Definition 4.2: A causal model $M' = ((\mathcal{U}', \mathcal{V}', \mathcal{R}'), \mathcal{F}')$ is a *conservative extension* of $M = ((\mathcal{U}, \mathcal{V}, \mathcal{R}), \mathcal{F})$ if $\mathcal{U} = \mathcal{U}'$, $\mathcal{V} \subseteq \mathcal{V}'$ if, for all contexts $\vec{u}$, all variables $X \in \mathcal{V}$, and all settings $\vec{w}$ of the variables in $\vec{W} = \mathcal{V} - \{X\}$, we have $(M, \vec{u}) \models [\vec{W} \leftarrow \vec{w}](X = x)$ iff $(M', \vec{u}) \models [\vec{W} \leftarrow \vec{w}](X = x)$. That is, no matter how we set the variables other than X, X has the same value in context $\vec{u}$ in both M and M'. ∎

The following lemma gives further insight into the notion of conservative extension. It says that if M' is a conservative extension of M, then the same causal formulas involving only the variables in $\mathcal{V}$ hold in M and M'.

[7] As a separate matter, most people would agree that Nader entering the race was a cause of Gore not winning, while Gore entering the race was not a cause of Nader not winning. Here the analysis is different. If Nader hadn't entered, it seems reasonable to assume that there would have been no other strong third-party candidate, so just about all of Nader's votes would have gone to Bush or Gore, with the majority going to Gore. On the other hand, if Gore hadn't entered, there would have been another Democrat in the race replacing him, and most of Gore's votes would have gone to the new Democrat in the race, rather than Nader.

Lemma 4.3: *Suppose that M' is a conservative extension of $M = ((\mathcal{U}, \mathcal{V}, \mathcal{R}), \mathcal{F})$. Then for all causal formulas φ that mention only variables in $\mathcal{V}$ and all context u, we have $(M, u) \models \varphi$ iff $(M', u) \models \varphi$.*

Proof: See the full paper. ∎

4.3 Avoiding AC2(b)

I now show that we can always use AC2(b′) instead of AC2(b), if we add extra variables.

Theorem 4.4: *If $X = x$ is not a cause of $Y = y$ in $(M, \vec{u})$ using AC2(b), but is a cause using AC2(b′), then there is a model M' that is a conservative extension of M such that $X = x$ is not a cause of $Y = y$ using AC2(b′).*

Proof: Suppose that $(\vec{W}, \vec{w}, x')$ is a witness to $X = x$ being a cause of $Y = y$ in $(M, \vec{u})$ using AC2(b′). Let $(M, \vec{u}) \models \vec{W} = \vec{w}$. We must have $\vec{w} \neq \vec{w}^*$, for otherwise it is easy to see that $X = x$ would be a cause of $Y = y$ in $(M, \vec{u})$ using AC(2b) with witness $(\vec{W}, \vec{w}, x')$.

If M' is a conservative extension of M with additional variables $\mathcal{V}'$, and $X = x$ is a cause of $Y = y$ in $(M, \vec{u})$ with witness $(\vec{W}, \vec{w}, x')$, we say that $(\vec{W}', \vec{w}', x')$ *extends* $(\vec{W}, \vec{w}, x')$, if $\vec{W} \subseteq \vec{W}' \subseteq \vec{W} \cup \mathcal{V}'$ and $\vec{w}'$ agrees with $\vec{w}$ on the variables in $\vec{W}$.

I now construct a conservative extension M' of M in which $X = x$ is not a cause of $Y = y$ using AC2(b′) with a witness extending $(\vec{W}', \vec{w}, x')$. Of course, this just kills one witness. I then show that we can construct further extensions to kill all other witnesses to $X = x$ being a cause of $Y = y$ using AC2(b′).

Let M' be obtained from M by adding one new variable N. All the variables have the same equations in M and M' except for Y and (of course) N. The equations for N are easy to explain: if $X = x$ and $\vec{W} = \vec{w}$, then $N = 1$; otherwise, $N = 0$. The equations for Y are the same in M and M' (and do not depend on the value of N) except for two special cases. To define these cases, for each variable $Z \in \mathcal{V} - \vec{W}$, if $x'' \in \{x, x'\}$ define $z_{x'',\vec{w}}$ as the value such that $(M, \vec{u}) \models [X \leftarrow x'', \vec{W} \leftarrow \vec{w}](Z = z_{x'',\vec{w}})$. That is, $z_{x'',\vec{w}'}$ is the value taken by Z if X is set to x'' and $\vec{W}$ is set to $\vec{w}$. Let $\vec{V}'$ consist of all variables in $\mathcal{V}$ other than Y, let $\vec{v}'$ be a setting of the variables in $\vec{V}'$, and let $\vec{Z}'$ consist of all variables in $\vec{V}' - \vec{W}$ other than X. Then we want the equations for Y in M' to be such that for all $j \in \{0, 1\}$, we have

$$(M, \vec{u}) \models [\vec{V}' \leftarrow \vec{v}'](Y = y'') \text{ iff}$$
$$(M', \vec{u}) \models [\vec{V}' \leftarrow \vec{v}'; N \leftarrow j](Y = y'')$$

unless the assignment $\vec{V}' \leftarrow \vec{v}'$ results in either (a) $X = x$, $\vec{W} = \vec{w}$, $Z = z_{x,\vec{w}}$ for all $Z \in \vec{Z}'$, and $N = 0$ or (b) $X = x'$, $\vec{W} = \vec{w}$, $Z = z_{x',\vec{w}}$ for all $Z \in \vec{Z}'$, and $N = 1$. (Note that in both of these cases, the value of N is "abnormal". If $X = x$, $\vec{W} = \vec{w}$ and $Z = z_{x,\vec{w}}$ for all $z \in \vec{Z}'$, then N should be 1; if we set X to x' and change the values of the variables in $\vec{Z}'$ accordingly, then N should be 0.) If (a) holds, $Y = y'$ in M'; if (b) holds, $Y = y$.

Lemma 4.5:

(a) It is not the case that $X = x$ is a cause of $Y = y$ using AC2(b′) in M' with a witness that extends $(\vec{W}, \vec{w}, x')$.

(b) M' is a conservative extension of M.

(c) if $X = x$ is a cause of $Y = y$ in $(M', \vec{u})$ using AC2(b) (resp. AC2(b′)) with a witness extending $(\vec{W}', \vec{w}', x'')$ then $X = x$ is a cause of $Y = y$ in $(M, \vec{u})$ using AC2(b) (resp. AC2(b′)) with witness $(\vec{W}', \vec{w}', x'')$.

The proof of Lemma 4.5 and the remainder of the proof of Theorem 4.4 can be found in the full paper. ∎

4.4 Discussion

Theorem 4.4 suggests that, by adding extra variables appropriately, we can go back to the definition of causality using AC2(b′) rather than AC2(b). This has some technical advantages. For example, with AC2(b′), causes are always single conjuncts (Eiter & Lukasiewicz 2002; Hopkins 2001). As shown in (Halpern 2008), this is not in general the case with AC2(b); it may be that $X_1 = x_1 \wedge X_2 = x_2$ is a cause of $Y = y$ with neither $X_1 = x_1$ nor $X_2 = x_2$ being causes. It also seems that testing for causality is harder using AC2(b). Eiter and Lukasiewicz (2002) show that, using AC2(b′), testing for causality is NP-complete for binary models (where all random variables are binary) and Σ_2-complete in general; with AC2(b), it seems to be Σ_2-complete in the binary case and Π_3-complete in the general case (Aleksandrowicz *et al.* 2013).

On the other hand, adding extra variables may not always be a natural thing to do. For example, in Beer et al.'s (2012) analysis of software errors, the variables chosen for the analysis are determined by the program specification. Moreover, Beer et al. give examples where AC2(b) is needed to get the intuitively correct answer. Unless we are given a principled way of adding extra variables so as to be able to always use AC2(b′), it is not clear how to automate an analysis. There does not always seem to be a "natural" way of adding extra variables so that AC2(b′) suffices (even assuming that we can agree on what "natural" means!).

Adding extra variables also has an impact on complexity. Note that, in the worst case, we may have to add an extra variable for each pair (W, w) such that there is a witness (W, w, x') for $X = x$ being a cause of $Y = y$. In all the standard examples, there are very few witnesses (typically 1–2), but I have been unable to prove a nontrivial bound on the number of witnesses.

More experience is needed to determine which of AC2(b) and AC2(b′) is most appropriate. Fortunately, in many cases, the causality judgment is independent of which we use.

5 The Stability on Non-Causality

The examples in Section 3 raises a potential concern. Consider the rock-throwing example again. Adding extra variables changed $BT = 1$ from being a cause of $BS = 1$ to

not being a cause. Could adding even more variables convert $BT = 1$ back to being a cause? Could it then alternate further?

As I now show, this cannot happen. Although, as we saw with the rock-throwing example, adding variables can convert a single-variable cause to a non-cause, it cannot convert a single-variable non-cause to a cause. Non-causality is stable.

Theorem 5.1: *If $X = x$ is not a cause of φ in (M, u), and M' is a conservative extension of M, then $X = x$ is not a cause of φ in (M', u).*

Proof: Suppose, by way of contradiction, that $X = x$ is not a cause of φ in (M, u), M' is a conservative extension of M, and $\vec{X} = \vec{x}$ is a cause of φ in (M', u) with witness $(\vec{W}, \vec{w}, \vec{x}')$. Let $\vec{W}_1$ be the intersection of $\vec{W}$ with $\mathcal{V}$, the set of endogenous variables in M, let $\vec{Z}_1 = \mathcal{V} = \vec{W}$, and let $\vec{w}_1$ be the restriction of $\vec{w}$ to the variables in $\vec{W}_1$. Since $X = x$ is not a cause of φ in (M, u), it is certainly not a cause with witness $(\vec{W}_1, \vec{w}_1, x')$, so either (a) $(M, u) \models X \neq x \vee \neg\varphi$; (b) $(M, u) \models [X \leftarrow x', \vec{W}_1 \leftarrow \vec{w}_1]\varphi$, or (c) there must be subsets $\vec{W}_1'$ of $\vec{W}_1$ and $\vec{Z}_1'$ of $\vec{Z}_1$ such that if $(M, u) \models \vec{Z}_1' = \vec{z}_1$ (i.e., $\vec{z}_1$ gives the actual values of the variables in $\vec{Z}_1'$), then $(M, u) \not\models [X \leftarrow x, \vec{W}_1' \leftarrow \vec{w}_1, \vec{Z}_1' \leftarrow \vec{z}_1]\neg\varphi$. Since M' is a conservative extension of M, by Lemma 4.3, if (a) holds, then $(M', u) \models X \neq x \vee \neg\varphi$; if (b) holds, $(M', u) \models [X \leftarrow x', \vec{W}_1 \leftarrow \vec{w}_1]\varphi$; and if (c) holds, then $(M', u) \not\models [X \leftarrow x, \vec{W}_1' \leftarrow \vec{w}_1, \vec{Z}_1' \leftarrow \vec{z}_1]\neg\varphi$. Moreover, $(M', u) \models \vec{Z}_1' = \vec{z}_1$. Thus, it follows that $X = x$ is not a cause of φ with witness $(\vec{W}, \vec{w}, x')$ in (M', u). For if (a) holds, then AC1 fails; if (b) holds, then AC2(a) fails, and if (c) holds, then AC2(b) fails. ∎

Notice that this argument does not quite work if the cause involves multiple variables. If $\vec{X} = \vec{x}$ is a cause of φ in (M', u) with witness $(\vec{W}, \vec{w}, \vec{x}')$, it is possible that it is not a cause in (M, u) because AC3 fails. While I believe that stability of non-causality holds even for causes involving multiple variables, proving it remains an open problem.

It is interesting that, in the single-variable case, non-causality is stable, while causality is not. Intuitively, this says that if $X = x$ is not a cause of $Y = y$, this is somehow encoded in the variables. For example, the fact that Billy's throw is not a cause of the bottle shattering in M'_{RT} is due to the variables *BH* and *SH* encoding the fact that Suzy's rock hits and Billy's does not. This information is maintained in any extension of $M'_{|RT}$, so Billy's throw cannot become a cause again.

6 Conclusions

This paper has demonstrated the HP definition of causality is remarkably resilient, but it emphasizes how sensitive knowledge representation can be to the choice of models. The focus has been on showing that the choice of variables is a powerful modeling tool. But it is one that can be abused. One lesson that comes out clearly is the need to have variables that describe the mechanism of causality, particularly if there is more than one mechanism. However, this is hardly a general recipe. Rather, it is a heuristic for constructing a "good" model. As Hitchcock and I pointed out (Halpern & Hitchcock 2010), constructing a good model is still more of an art than a science.

References

Aleksandrowicz, G.; Chockler, H.; Halpern, J. Y.; and Ivrii, A. 2013. Complexity of causality: updated results. unpublished manuscript.

Beer, I.; Ben-David, S.; Chockler, H.; Orni, A.; and Trefler, R. J. 2012. Explaining counterexamples using causality. *Formal Methods in System Design* 40(1):20–40.

Blanchard, T., and Schaffer, J. 2013. Cause without default. unpublished manuscript.

Cushman, F.; Knobe, J.; and Sinnott-Armstrong, W. 2008. Moral appraisals affect doing/allowing judgments. *Cognition* 108(1):281–289.

Eiter, T., and Lukasiewicz, T. 2002. Complexity results for structure-based causality. *Artificial Intelligence* 142(1):53–89.

Gerstenberg, T., and Lagnado, D. 2010. Spreading the blame: the allocation of responsibility amongst multiple agents. *Cognition* 115:166–171.

Glymour, C.; Danks, D.; Glymour, B.; Eberhardt, F.; Ramsey, J.; Scheines, R.; Spirtes, P.; Teng, C. M.; and Zhang, J. 2010. Actual causation: a stone soup essay. *Synthese* 175:169–192.

Hall, N. 2007. Structural equations and causation. *Philosophical Studies* 132:109–136.

Halpern, J. Y., and Hitchcock, C. 2010. Actual causation and the art of modeling. In Dechter, R.; Geffner, H.; and Halpern, J., eds., *Causality, Probability, and Heuristics: A Tribute to Judea Pearl*. London: College Publications. 383–406.

Halpern, J. Y., and Hitchcock, C. 2014. Graded causation and defaults. *The British Journal for the Philosophy of Science*. Available at http://www.cs.cornell.edu/home/halpern/papers/normality.pdf.

Halpern, J. Y., and Pearl, J. 2001. Causes and explanations: A structural-model approach. Part I: Causes. In *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI 2001)*, 194–202.

Halpern, J. Y., and Pearl, J. 2005. Causes and explanations: A structural-model approach. Part I: Causes. *British Journal for Philosophy of Science* 56(4):843–887.

Halpern, J. Y. 2008. Defaults and normality in causal structures. In *Principles of Knowledge Representation and Reasoning: Proc. Eleventh International Conference (KR '08)*. 198–208.

Hiddleston, E. 2005. Causal powers. *British Journal for Philosophy of Science* 56:27–59.

Hitchcock, C., and Knobe, J. 2009. Cause and norm. *Journal of Philosophy* 106:587–612.

Hitchcock, C. 2001. The intransitivity of causation revealed in equations and graphs. *Journal of Philosophy* XCVIII(6):273–299.

Hitchcock, C. 2007. Prevention, preemption, and the principle of sufficient reason. *Philosophical Review* 116:495–532.

Hopkins, M., and Pearl, J. 2003. Clarifying the usage of structural models for commonsense causal reasoning. In *Proc. AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*.

Hopkins, M. 2001. A proof of the conjunctive cause conjecture. Unpublished manuscript.

Kahneman, D., and Miller, D. T. 1986. Norm theory: comparing reality to its alternatives. *Psychological Review* 94(2):136–153.

Knobe, J., and Fraser, B. 2008. Causal judgment and moral judgment: two experiments. In Sinnott-Armstrong, W., ed., *Moral Psychology, Volume 2: The Cognitive Science of Morality*. Cambridge, MA: MIT Press. 441–447.

Lagnado, D.; Gerstenberg, T.; and Zultan, R. 2013. Causal responsibility and counterfactuals. *Cognitive Science* 37:1036–1073.

Livengood, J. 2013. Actual causation in simple voting scenarios. *Nous* 47(2):316–345.

Spohn, W. 2008. Personal email.

Weslake, B. 2014. A partial theory of actual causation. *The British Journal for the Philosophy of Science*. To appear.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

∃GUARANTEENASH for Boolean Games is NEXP-Hard

Egor Ianovski and **Luke Ong**
Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford, UK

Abstract

Boolean games are an expressive and natural formalism through which to investigate problems of strategic interaction in multiagent systems. Although they have been widely studied, almost all previous work on Nash equilibria in Boolean games has focused on the restricted setting of pure strategies. This is a shortcoming as finite games are guaranteed to have at least one equilibrium in mixed strategies, but many simple games fail to have pure strategy equilibria at all. We address this by showing that a natural decision problem about mixed equilibria: determining whether a Boolean game has a mixed strategy equilibrium that guarantees every player a given payoff, is NEXP-hard. Accordingly, the ϵ variety of the problem is NEXP-complete. The proof can be adapted to show coNEXP-hardness of a similar question: whether all Nash equilibria of a Boolean game guarantee every player at least the given payoff.

Introduction

A multiagent environment makes strategic considerations inevitable. Any attempt to explain the behaviour of a system consisting of self-interested agents cannot ignore the fact that agents' behaviour may be influenced or completely determined by the behaviour of other agents in the system. As the field of game theory concerns itself with precisely these issues, its concepts find fertile ground in the study of multiagent systems.

A shortcoming of game theoretical techniques is that games, being combinatorial objects, are liable to get very large very quickly. Any computational application of game theory would need alternative representations to the normal and extensive forms prominent in the economics literature. One such representation, based on propositional logic, is the Boolean game.

Boolean games were initially introduced as two player games which have an algebra isomorphic to the Lindenbaum algebra for propositional logic (Harrenstein et al. 2001). Since then Boolean games have garnered interest from the multiagent community as a simple yet expressive framework to model strategic interaction. This has led to the study of complexity issues involved in reasoning about these games. While many questions have been answered, the issue of mixed strategies remained open.

In this paper we address this lacuna and present the first complexity result about mixed equilibria in the Boolean games literature: the NEXP-hardness of ∃GUARANTEENASH, which asks whether a Boolean game has an equilibrium where each player attains at least $\boldsymbol{v}[i]$ utility, for some input vector $\boldsymbol{v}$.

Related Work

Complexity results for Boolean games were first studied in the two player case by Dunne and van der Hoek (2004) and the n-player case by Bonzon et al. (2006), where among other results the authors showed that determining the existence of a pure equilibrium is Σ_2^p-complete in the general case, and can be easier should some restrictions be placed on the goal formulae of the players. A later paper considered to what extent the interdependency of players contributed to complexity (Bonzon, Lagasquie-Schiex, and Lang 2009). Further enquiry into tractable fragments of Boolean games was carried out by Dunne and Wooldridge (2012).

Since their inception Boolean games have attracted the interest of the multiagent community, which led to increased interest about their cooperative side. The standard game theoretic notion of core was studied in a Boolean setting (Dunne et al. 2008; Bonzon, Lagasquie-Schiex, and Lang 2012), and a number of papers considered means of lining up player incentives with a central planner (Endriss et al. 2011; Kraus and Wooldridge 2012; Levit et al. 2013). in two recent papers (2013a; 2013b), Ågotnes et al. considered ways of combining Boolean games with the spiritually similar games in epistemic logic (Ågotnes et al. 2011; Ågotnes and van Ditmarsch 2011).

All the work mentioned has been in the setting of pure strategies only. Mixed strategies do arise in a cardinal extension to Boolean games dubbed weighted Boolean formula games (Mavronicolas, Monien, and Wagner 2007) or satisfiability games (Bilò 2007), but there the authors only consider a very restricted fragment of these games, and its relation to a subclass of the congestion games of Rosenthal (1973).

A very similar framework to Boolean games is that of Boolean circuit games (Schoenebeck and Vadhan 2012), building on earlier work by Feigenbaum, Koller, and Shor

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

(1995). There players are equipped with a Boolean circuit with k input gates and m output gates. The input gates are partitioned among the players, and a player's strategy is an assignment of values to the gates under his control. The output gates encode a binary representation of the player's utility on a given input. In this setting we do find a treatment of mixed strategies, including the complexity of ∃GUARANTEENASH.

Note that a Boolean game can be seen as a very specific type of Boolean circuit game: the players' circuits are restricted to NC^1, and the number of output gates to one. Thus easiness results for Boolean circuit games directly transfer to Boolean games, and hardness results transfer in the other direction. In particular, this means that the NEXP-completeness of ∃GUARANTEENASH for Boolean circuit games proved by Schoenebeck and Vadhan does not imply the result of this paper.

Preliminaries

While there are many breeds of games in the literature, we here restrict ourselves to what is perhaps the most widely studied class:

Definition 1. A *finite strategic game* consists of n players, each equipped with a finite set of pure strategies, S_i, and a utility function $u_i : S_1 \times \cdots \times S_n \to \mathbb{R}$.

An n-tuple of strategies is called a *strategy profile*: thus a utility function maps strategy profiles to the reals.

Example 1. In a game of *matching pennies* two players are given a coin each and may choose to display that coin heads or tails up. Player Two seeks to match the move of Player One, while player one seeks to avoid that. Hence we have $u_2(HH) = u_2(TT) = 1$, $u_1(HT) = u_1(TH) = 1$, and 0 otherwise.

Note that to represent a finite strategic game explicitly (the *normal form* of the game) we would need to list the players' utility on every possible profile. This would require on the order of $n|S_i|^n$ entries, taking S_i to mean the size of the "typical" strategy set. Such a representation is both exponential in the number of players and linear in the number of strategies - which in itself may be very large, e.g. in chess.

Ideally we would wish to avoid such a representation. If a game has some internal structure, it would be natural to ask if the game can be described in a more succinct way. In the case where the game can be interpreted as players holding propositional preferences over Boolean variables the Boolean game offers precisely that.

Definition 2. A *Boolean game* is a representation of a finite strategic game given by n disjoint sets of propositional variables, Φ_i, and n formulae of propositional logic, γ_i.

The intended interpretation is that player i controls the variables in Φ_i in an attempt to satisfy γ_i, which may depend on variables not in player i's control. Player i plays a truth assignment to Φ_i, i.e. his set of pure strategies is 2^{Φ_i}. A profile of strategies, ν, is thus a model of proposition logic and i's utility function is $\nu \mapsto 1$ if $\nu \vDash \gamma_i$, and $\nu \mapsto 0$ otherwise.

Example 2. Matching pennies can be given a Boolean representation by setting $\Phi_1 = \{p\}$, $\Phi_2 = \{q\}$, $\gamma_1 = \neg(p \leftrightarrow q)$ and $\gamma_2 = p \leftrightarrow q$.

The size of a Boolean game is thus on the order of $n(\max_i(|\Phi_i|) + \max_i(|\gamma_i|))$. In the best case γ_i is small and the resulting representation is linear in the number of players and logarithmic in the number of strategies, giving greater succinctness on both fronts.

Having defined the game representation, we now turn to reasoning about such games. The most common solution concept is the Nash equilibrium, which we define below.

Definition 3. Given a strategy profile $\boldsymbol{s}$, we use $\boldsymbol{s}_{-i}(\sigma_i')$ to mean the profile obtained by replacing the strategy of i in $\boldsymbol{s}$ with σ_i'. A *best response* for i to $\boldsymbol{s}$ is some σ_i' that maximises $u_i(\boldsymbol{s}_{-i}(\sigma_i'))$.

A strategy profile $\boldsymbol{s} = (\sigma_1, \ldots, \sigma_n)$ where every σ_i is a best response to $\boldsymbol{s}$ is a *Nash equilibrium*.

Example 3. In a game of matching pennies, T is a best response for Player One to HH, and H is a best response for Player Two, so the profile HH is not in equilibrium. In fact, the game has no equilibria in pure strategies.

The fact that games as simple as matching pennies may fail to have a pure strategy equilibrium casts doubt on its suitability as a solution concept. Fortunately, a natural extension of the framework rectifies the matter.

Definition 4. A *mixed strategy* for player i in a finite strategic game is a probability distribution over S_i.

The utility player i obtains from a profile of mixed strategies S is $\sum p(S')u_i(S')$, where $p(S')$ is the probability assigned to the pure profile S' by the mixed strategies in S.

If the idea of playing a probability distribution seems strange, the reader may instead opt to view a mixed strategy as a randomised decision procedure that leads to an outcome in the set of pure strategies, S_i. In a game of rock-paper-scissors, a mixed strategy could consist of generating a number in $[0, 1]$ then playing rock if it falls between 0 and 0.4, paper if it falls between 0.4 and 0.9 and scissors if it is above 0.9. The payoff of a mixed strategy is the expected payoff - the sum of the possible outcomes weighted by their probability.

It is in this context that Nash proved his seminal result:

Theorem 1 (Nash, 1951). *Every finite strategic game has an equilibrium in mixed strategies.*

Example 4. The unique equilibrium of matching pennies involves both players randomising over their sets of strategies by assigning a weight of 0.5 to both H and T. In this equilibrium both players attain a utility of 0.5.

In the above example the reader will observe that 0.5 is also the utility Player One would attain by deviating to the pure strategy H or the pure strategy T. This is always the case - as the payoff of a mixed strategy is a weighted sum of pure strategies, it cannot yield more utility than any of its components individually. What this means for us is that if it is possible for a player to deviate from a profile, then it is possible to deviate with a pure strategy. If we want to verify that a mixed profile is in equilibrium, we need only

check that no pure strategy for a player will lead to a better outcome.

Since every game has an equilibrium, the algorithmic question of asking whether an equilibrium exists is not relevant. This motivates decision problems based on qualified notions of equilibria, such as the one that concerns us in this paper:

∃GUARANTEENASH: Given a Boolean game G and a rational-valued vector $\boldsymbol{v} \in [0,1]^n$, does G have an equilibrium $\boldsymbol{s}$ such that $u_i(\boldsymbol{s}) \geq \boldsymbol{v}[i]$ for each player i?

For representation purposes, as the entries of $\boldsymbol{v}$ are rationals we will assume they are given as a pair of integers representing the numerator and denominator.

It is natural to also consider a problem closely related to the dual:

∀GUARANTEENASH: Given a Boolean game G and a rational-valued vector $\boldsymbol{v} \in [0,1]^n$, does every equilibrium of G, $\boldsymbol{s}$, satisfy $u_i(\boldsymbol{s}) \geq \boldsymbol{v}[i]$ for each player i?

Main Result

Our reduction will be from the following NEXP-complete problem:

NEXPTM: Given a non-deterministic Turing machine M, an integer in binary K and a string w, does M accept w in at most K steps?

Proposition 1. NEXPTM *is NEXP-complete.*

Proof. For membership in NEXP, we need only simulate the computation of M on w for K steps. Each step can be simulated in non-deterministic polynomial time, and the number of steps is exponential in $|K|$.

For hardness, let N be a non-deterministic Turing machine with an exponential time clock f. Let M be a Turing machine with an identical transition relation to N, but with no internal clock. Clearly, N accepts w if and only if M accepts w in at most $f(w)$ steps. That is, $(M, f(w), w)$ is a positive instance of NEXPTM. Moreover, the triple $(M, f(w), w)$ is polynomial in the size of N and w: $|M| \leq |N|$, $|w| = |w|$ and as $f(w) \in O(2^{p(|w|)})$, when written in binary it is of size $O(p(|w|))$. This gives us the desired reduction. □

We can now prove the hardness of ∃GUARANTEENASH. For questions of NEXP-membership, see the discussion below.

Theorem 2. ∃GUARANTEENASH *for Boolean games is NEXP-hard.*

Proof. We will give a reduction from NEXPTM. Given a triple (M, K, w) we shall construct, in polynomial time, a Boolean game G and a utility vector $\boldsymbol{v}$, such that G has an equilibrium where player i's utility is at least $\boldsymbol{v}[i]$ if and only if M accepts w in K steps or less.

For convenience, we augment M with a "do nothing" transition: if M is at an accepting state, then we allow it to move to the next computation step without moving the head, changing state, or writing anything to the tape. It is clear that augmenting M in such a fashion does not change the language accepted by M, but it ensures that the machine state is defined at all computation steps; we do not need to worry about the case where the machine accepts in under K steps, as if it does, it will still accept at step K.

Let $k = |K|$, and q be the number of states of M.

For intuition, a computation history of M on w could be seen as a $K \times K$ table, or for simplicity $2^k \times 2^k$, padding as needed. The ith row, jth column contains the contents of the jth tape cell at the ith computation step, whether or not the head is over that cell at that step and, if so, which state the machine is in. As the size of the table is 2^k by 2^k, we can index an entry by using two k-bit integers.

A way to visualise the proof is that in our game G, which consists of six players, Player One is equipped with variables that allow him to describe a single entry of this table. Player Four plays a partial matching pennies game against Player One, thereby forcing Player One to play a mixed strategy randomising over all entries of the table, and thus specifying an entire computation history with his mixed strategy. Player Two then verifies that the mixed strategy provided by Player One contains a consistent description of the head location at each computation step, and Player Three checks that every two consecutive steps are linked by exactly one transition rule. Players Five and Six play matching pennies with Players Two and Three to force them to randomise across all table entries.

To this end, let:

$$\Phi_1 = \{Zero_1, One_1, Head_1, Left_1, Right_1\} \cup \{Time_1^i\}_{1\leq i\leq k} \cup \{Tape_1^i\}_{1\leq i\leq k} \cup \{State_1^i\}_{1\leq i\leq q}.$$

The intended meaning of $Time_1^i$ (respectively $Tape_1^i$) is the value of the ith most significant bit of the integer denoting the index of the computation step (respectively tape cell) in question, given the standard convention of interpreting "true" as 1 and "false" as 0. A truth assignment by Player One can therefore be read as: at the computation step specified by $Time_1^1, \ldots, Time_1^k$ the tape cell specified by $Tape_1^1, \ldots, Tape_1^k$ contains 0 if $Zero_1$, 1 if One_1 and is blank if neither. The machine head is hovering over the cell in question if $Head_1$, and is located to the left or right of that cell respectively if $Left_1$ or $Right_1$. If the head is over the cell in question, the machine is in state i if $State_1^i$ (if the head is not over the cell, $State_1^i$ is a junk variable that has no meaning).

Player One's goal formula is a conjunction of four subformulae:

$$\gamma_1 = Init \wedge Final \wedge Cons_1 \wedge \neg\gamma_4.$$

Intuitively, $Init$ means that if the player plays the first computation step, his description of the machine must agree with the initial configuration of M on w. $Final$ means that if the player plays the last (Kth) computation step, the machine must be in an accepting state. $Cons_1$ states the description of the machine must be internally consistent. The final conjunct is to force the player to randomise across all

computation steps and tape cells, to which we will return later.

Init requires that at time zero the configuration of the machine is faithfully represented by Player One's play. This takes the form of an implication where the antecedent states that we are at step zero:

$$Init = (\bigwedge_{1 \leq i \leq k} \neg Time_1^i) \rightarrow Consequent.$$

The consequent itself is a conjunction of three further subformulae, for the head, the state and the tape.

$$Consequent = InitHead \wedge InitState \wedge InitTape.$$

The head requirement states that the head is at the leftmost cell. That is, at cell zero $Head_1$ is true, and at every other cell $Left_1$ is true:

$$InitHead = \Big((\bigwedge_{1 \leq i \leq k} \neg Tape_1^i) \rightarrow Head_1\Big) \wedge \Big(\neg(\bigwedge_{1 \leq i \leq k} \neg Tape_1^i) \rightarrow Left_1\Big).$$

The state requirement is simply M's initial state:

$$InitState = State_1^{initial}.$$

The tape requirement is a conjunction of $|w| + 1$ implications. The first $|w|$ implications state that if the tape cell chosen is within the first $|w|$ cells, then its contents must agree with w. If we use i as shorthand for the conjunction of tape variables expressing i, and $w[i]$ for $Zero_1$ or One_1 depending on the ith bit of w, this has the following form:

$$InitTape = \bigwedge_{0 \leq i < |w|} (i \rightarrow w[i]) \wedge BlankCells.$$

Note that this formula is linear in $|w|$, so the construction so far was polynomial.

The last formula in *InitTape* states that all other cells are blank.

$$BlankCells = \neg(\bigvee_{0 \leq i < |w|} i) \rightarrow (\neg Zero_1 \wedge \neg One_1).$$

Final states that at computation step K, the machine accepts. If we use K as shorthand for the appropriate conjunction of time variables, we get the following implication:

$$Final = K \rightarrow State_1^{accepting}.$$

$Cons_1$ requires that the player's description of a given computation step and cell is internally consistent. This means the cell cannot have both 0 and 1 on it, the head must be either over the cell or to one direction and the machine must be in exactly one state. It is worth noting that this says nothing about whether Player One's descriptions of different steps and cells are consistent with each other: this is the task of Players Two and Three.

For $Cons_1$, we introduce a generalised XOR symbol, which we denote $\boldsymbol{OneOf}$, with the interpretation that $\boldsymbol{OneOf}(\varphi_1, \ldots, \varphi_n)$ is true if and only if exactly one φ_i is. Such a symbol could be replaced by a propositional logic formula polynomial in the size of $\varphi_1, \ldots, \varphi_n$ - simply take the disjunction of all n admissible possibilities. This gives us the desired formula:

$$\begin{aligned} Cons_1 = {} & \neg(Zero_1 \wedge One_1) \\ & \wedge \boldsymbol{OneOf}(Head_1, Left_1, Right_1) \\ & \wedge \boldsymbol{OneOf}(\overline{State_1^i}). \end{aligned}$$

By $\overline{State_1^i}$ we mean $State_1^1, \ldots, State_1^q$.

To finish the description of γ_1, we turn to Player Four. Player Four is playing a partial matching pennies game with Player One over the time and tape variables. We thus equip her with the following:

$$\Phi_4 = \{Time_4^i\}_{1 \leq i \leq k} \cup \{Tape_4^i\}_{1 \leq i \leq k}.$$

The objective is to guess the same computation step and cell index as player one:

$$\gamma_4 = \Big(\bigwedge_{1 \leq i \leq k} (Time_1^i \leftrightarrow Time_4^i)\Big) \wedge \Big(\bigwedge_{1 \leq i \leq k} (Tape_1^i \leftrightarrow Tape_4^i)\Big).$$

Since the payoffs of Player One and Four depend only on each other, we can already describe how they will behave in any equilibrium. Player One has the power to satisfy *Init*, *Final* and $Cons_1$ unilaterally, so all that remains is $\neg\gamma_4$. To maximise the chances of this happening, Player One's best bet is to play every step/cell with probability $1/2^{2k}$ - this will ensure that Player Four will guess the same step with probability $1/2^{2k}$, whereas if Player One were to play any (i, j) with a greater probability than $1/2^{2k}$, then Player Four could play (i, j) with probability 1 and thus guess Player One's choice more often.

As such in any equilibrium play Player One will describe entry (i, j) of the computation table with probability $1/2^{2k}$.

Player Two's purpose is to verify the consistency of Player One's description of the head. This involves verifying that at a given computation step the $Head_1$ variable is true in exactly one cell, $Left_1$ is true in every cell to the right and $Right_1$ is true in every cell to the left.

Now of course, she cannot verify the entire row in one go - there are 2^k cells to check. Instead a pure strategy for her will check only a pair of consecutive cells. Player Five will then force her to randomise over *all* pairs of consecutive cells - and if Player One's description is inconsistent, there must be a witnessing pair.

Player Two controls the following variables:

$$\begin{aligned} \Phi_2 = {} & \\ & \{Head_2, sHead_2, Left_2, sLeft_2, Right_2, sRight_2\} \\ & \cup \{Tape_2^i\}_{1 \leq i \leq k} \cup \{sTape_2^i\}_{1 \leq i \leq k} \cup \{Time_2^i\}_{1 \leq i \leq k}. \end{aligned}$$

The lowercase "s" can be read as "successor". The intended meaning of these variables is that $Tape_2^1, \ldots, Tape_2^k$ name a cell and $sTape_2^1, \ldots, sTape_2^k$ the cell directly to the right of it. The other variables state the location of the head

in relation to these two cells at the computation step specified by the time variables.

Player Two's goal formula is a conjunction of four subformulae:

$$\gamma_2 = \mathit{MatchOne}_2 \wedge \mathit{Cons}_2 \wedge \boldsymbol{Succ}_2 \wedge \neg\gamma_5.$$

Intuitively, $\mathit{MatchOne}_2$ states that Player Two ought to play the same head configuration as dictated by Player One. Cons_2 requires that this configuration be internally consistent. $\boldsymbol{Succ}_2$ is to ensure that the two cells chosen are indeed consecutive.

Before we state $\mathit{MatchOne}_2$ we ought to first ask what we mean by saying that players one and two play the same head configuration. As in any given (pure) strategy profile, either player will be describing a single computation step and at most two cells; if it turns out that they are speaking about different step/cell configurations we should not be concerned about whatever claims they make - we have no way to prove they are not in agreement, so we may as well assume that they are. Only in the instance where they happen to refer to the same step/cell should we expect accord. Since Player Two is referring to two cells in any play, we require that if either of the cells she references coincides with that referenced by Player One, they must agree.

The desired formula is thus of the following form:

$$\begin{aligned}\mathit{MatchOne}_2 = \\ &\mathit{AgreeTime} \rightarrow \Big((\mathit{AgreeCell} \rightarrow \mathit{AgreeHead}) \\ &\wedge (\mathit{sAgreeCell} \rightarrow \mathit{sAgreeHead})\Big).\end{aligned}$$

The subformulae are as follows:

$$\begin{aligned}\mathit{AgreeTime} &= \bigwedge_{1\le i\le k} (\mathit{Time}_1^i \leftrightarrow \mathit{Time}_2^i).\\ \mathit{AgreeCell} &= \bigwedge_{1\le i\le k} (\mathit{Tape}_1^i \leftrightarrow \mathit{Tape}_2^i).\\ \mathit{sAgreeCell} &= \bigwedge_{1\le i\le k} (\mathit{Tape}_1^i \leftrightarrow \mathit{sTape}_2^i).\\ \mathit{AgreeHead} &= (\mathit{Head}_1 \leftrightarrow \mathit{Head}_2)\\ &\quad \wedge (\mathit{Left}_1 \leftrightarrow \mathit{Left}_2)\\ &\quad \wedge (\mathit{Right}_1 \leftrightarrow \mathit{Right}_2).\\ \mathit{sAgreeHead} &= (\mathit{Head}_1 \leftrightarrow \mathit{sHead}_2)\\ &\quad \wedge (\mathit{Left}_1 \leftrightarrow \mathit{sLeft}_2)\\ &\quad \wedge (\mathit{Right}_1 \leftrightarrow \mathit{sRight}_2).\end{aligned}$$

Internal consistency amounts simply to the conjunction of the valid combinations of claims about the head:

$$\begin{aligned}\mathit{Cons}_2 = &(\mathit{Right}_2 \wedge \mathit{sRight}_2) \vee (\mathit{Right}_2 \wedge \mathit{sHead}_2)\\ &\vee (\mathit{Head}_2 \wedge \mathit{sLeft}_2) \vee (\mathit{Left}_2 \wedge \mathit{sLeft}_2)\end{aligned}$$

Succ_2 states that the two tape locations are, in fact, consecutive. We will prove a lemma to show that this is concisely expressible in propositional logic.

Lemma 1. *Let $\boldsymbol{Succ}(p_1,\ldots,p_n;q_1,\ldots,q_n)$ be a formula that is true if and only if the binary integer encoded by $q_1,\ldots,q_n$ is the successor of the binary integer encoded by $p_1,\ldots,p_n$. As a convention, 2^n-1 has no successor.*

$\boldsymbol{Succ}(p_1,\ldots,p_n;q_1,\ldots,q_n)$ can be replaced by a propositional formula of size polynomial in $p_1,\ldots,p_n$ and $q_1,\ldots,q_n$.

Proof. We take advantage of the fact that to increment a binary integer we only need to modify the rightmost consecutive block of 1s, and there are only n such possible blocks.

Since we have a boundary condition to consider, we require that the first integer is not 2^n-1:

$$\boldsymbol{Succ}(p_1,\ldots,p_n;q_1,\ldots,q_n) = \neg(\bigwedge_{1\le i\le n} p_i) \wedge \mathit{Succ}'.$$

Succ' is then:

$$\begin{aligned}&\Big(\neg p_1 \rightarrow \big(q_1 \wedge \bigwedge_{i=2}^{n}(p_i \leftrightarrow q_i)\big)\Big)\\ &\wedge\Big((p_1 \wedge \neg p_2) \rightarrow \big(\neg q_1 \wedge q_2 \wedge \bigwedge_{i=3}^{n}(p_i \leftrightarrow q_i)\big)\Big)\\ &\wedge\Big((p_1 \wedge p_2 \wedge \neg p_3) \rightarrow \big(\neg q_1 \wedge \neg q_2 \wedge q_3 \wedge \bigwedge_{i=4}^{n}(p_i \leftrightarrow q_i)\big)\Big)\\ &\ \vdots\\ &\wedge\Big((\neg p_n \wedge \bigwedge_{i=1}^{n-1} p_i) \rightarrow \big((\bigwedge_{i=1}^{n-1} \neg q_i) \wedge q_{i+1}\big)\Big).\end{aligned}$$

This is quadratic in the number of variables, giving us the desired result. □

Succ_2 can then be stated simply:

$$\mathit{Succ}_2 = \boldsymbol{Succ}(\overline{\mathit{Tape}_2^i}; \overline{\mathit{sTape}_2^i}).$$

Finally, Player Five is trying to guess Player Two's choice of cell and computation step.

$$\begin{aligned}&\Phi_5 = \{\mathit{Time}_5^i\}_{1\le i\le k} \cup \{\mathit{Tape}_5^i\}_{1\le i\le k}.\\ &\gamma_5 = \bigwedge_{i=1}^{k}(\mathit{Tape}_2^i \leftrightarrow \mathit{Tape}_5^i) \wedge \bigwedge_{i=1}^{k}(\mathit{Time}_2^i \leftrightarrow \mathit{Time}_5^i).\end{aligned}$$

Player Three's purpose is to verify that the tape contents in successive computation steps respect the transition rules of M. Like Player Two, he does this for a small, local part of the table, and is forced to randomise over the whole table by Player Six. To do this he specifies a total of six cells and two computation steps: consecutive triples in consecutive steps. Then he verifies that the tape contents, head position

and machine state are in agreement with some rule of M.

$$\begin{aligned}\Phi_3 = \{&pHead_3, Head_3, sHead_3, npHead_3, nHead_3,\\ &nsHead_3, pZero_3, Zero_3, sZero_3, npZero_3,\\ &nZero_3, nsZero_3, pOne_3, One_3, sOne_3,\\ &npOne_3, nOne_3, nsOne_3\}\\ \cup&\{pState_3^i\}_{1\le i\le q} \cup \{State_3^i\}_{1\le i\le q}\\ \cup&\{sState_3^i\}_{1\le i\le q} \cup \{npState_3^i\}_{1\le i\le q}\\ \cup&\{nState_3^i\}_{1\le i\le q} \cup \{nsState_3^i\}_{1\le i\le q}\\ \cup&\{pTape_3^i\}_{1\le i\le k} \cup \{Tape_3^i\}_{1\le i\le k}\\ \cup&\{sTape_3^i\}_{1\le i\le k} \cup npTape_3^i\}_{1\le i\le k}\\ \cup&\{nTape_3^i\}_{1\le i\le k} \cup \{nsTape_3^i\}_{1\le i\le k}\\ \cup&\{Time_3^i\}_{1\le i\le k} \cup \{nTime_3^i\}_{1\le i\le k}.\end{aligned}$$

The "p" can be read as "predecessor", referring to the cell to the left, and "n" as "next computation step". The intended meaning is simply the state and tape contents in each of the six cells, as well as whether the head is over that cell.

Player Three's goal formula is a conjunction of five subformulae:

$$\gamma_3 = MatchOne_3 \wedge Triple \wedge Succ_3 \wedge Rules \wedge \neg\gamma_6.$$

$MatchOne_3$ states that if any of the step/cell pairs named by Player Three coincide with the one named by Player One, Player Three must agree with Player One. $Triple$ requires that the three cells named in either computation step should be a consecutive triple, and the triple at either step must be the same. $Succ_3$ requires the two computation steps named to be consecutive. $Rules$ is to verify that the configuration thus described is consistent with a rule of M.

$MatchOne_3$ is a conjunction of a total of six statements, depending on which step/cell pair coincides with that played by Player One. We will only give one such statement below, in the case that Player One named the same step as $Time_3^1, \ldots, Time_3^k$ and the same cell as $pTape_3^1, \ldots, pTape_3^k$. The other five statements are obtained in the obvious manner.

$$\begin{aligned}&\Big(\bigwedge_{i=1}^{k}(Time_1^i \leftrightarrow Time_3^i) \wedge \bigwedge_{i=1}^{k}(Tape_1^i \leftrightarrow pTape_3^i)\Big) \rightarrow\\ &\Big((Zero_1 \leftrightarrow pZero_3) \wedge (One_1 \leftrightarrow pOne_3)\\ &\quad\wedge (Head_1 \leftrightarrow pHead_3) \wedge \bigwedge_{i=1}^{q}(State_1^i \leftrightarrow pState_3^i)\Big).\end{aligned}$$

$Triple$ states that the tape cells selected are consecutive triples, and that the same triple is chosen in both steps. It is worth noting that given our previous definition of successor, if Player Three is to satisfy this conjunct then the middle cell cannot be 0 or $2^k - 1$.

$$\begin{aligned}Triple = &\boldsymbol{Succ}(\overline{pTape_3^i}; \overline{Tape_3^i})\\ &\wedge \boldsymbol{Succ}(\overline{Tape_3^i}; \overline{sTape_3^i})\\ &\wedge \boldsymbol{Succ}(\overline{npTape_3^i}; \overline{nTape_3^i})\\ &\wedge \boldsymbol{Succ}(\overline{nTape_3^i}; \overline{nsTape_3^i})\\ &\wedge \bigwedge_{1\le i\le k}(Tape_3^i \leftrightarrow nTape_3^i).\end{aligned}$$

$Succ_3$ requires that the computation steps be consecutive:

$$Succ_3 = \boldsymbol{Succ}(\overline{Time_3^i}; \overline{nTime_3^i}).$$

$Rules$ is a conjunction of four formulae: three of the formulae are conjunctions containing an implication for each $(r, s) \in Q \times \{0, 1, \perp\}$, representing the machine's behaviour if it reads s in state r and the head is over the left, centre or right cell respectively. The fourth term is $NoHead$, to handle the case where the head is not over any cell in the triple:

$$Rules = Left \wedge Centre \wedge Right \wedge NoHead.$$

We will examine $Left$ and $NoHead$, understanding that $Centre$ and $Right$ are handled in similar fashion.

$$\begin{aligned}Left = \Big(\bigwedge_{(r,s)\in Q\times\{0,1,\perp\}} &\big((pState_3^r \wedge s) \rightarrow\\ &\boldsymbol{OneOf}(\overline{Rule[(r,s) \rightarrow (r', s', D)]})\big)\Big).\end{aligned}$$

The s in the antecedent is meant to be replaced by $pZero_3$, $pOne_3$, or $\neg(pZero_3 \vee pOne_3)$ as appropriate. The intuition of the $Rules$ term is that should the machine read s in state r it should pick exactly one of the rules available to it, and if the head is not present then the tape contents should not change.

The subformula to deal with a specific rule can be broken up as follows:

$$Rule[(r, s) \rightarrow (r', s', D)] = L \wedge B.$$

L describes the behaviour of the machine if the left cell is not the leftmost cell on the tape, B deals with the boundary case where it is.

We will give an example of how $Rule[(q_3, 0) \rightarrow (q_4, 1, \mathit{left})]$ would be handled. That is, the rule that says that if the machine reads a 0 in state q_3 it may write a 1, transition to state q_4 and move the head left. All rules except "do nothing" can be handled similarly, and "do nothing" would merely assert that if the machine reads an accepting state, then nothing changes.

The L part triggers if the head is over the leftmost cell in the triple, and the leftmost cell is not cell 0. It then ensures that in the next computation step the leftmost cell contains 1 and the other cells are unchanged. Since the head leaves the

monitored triples we need no terms to account for it.

$$L = \Big(\neg(\bigwedge_{1\leq i\leq k} \neg pTape_3^i) \wedge pHead_3\Big) \rightarrow \Big(npOne_3 \wedge (Zero_3 \leftrightarrow nZero_3) \wedge (sZero_3 \leftrightarrow nsZero_3) \wedge (One_3 \leftrightarrow nOne_3) \wedge (sOne_3 \leftrightarrow nsOne_3)\Big).$$

In the boundary case the head is over the leftmost cell of the tape, so when it attempts to move left it instead stands still.

$$B = \Big((\bigwedge_{1\leq i\leq k} \neg pTape_3^i) \wedge pHead_3\Big) \rightarrow \Big(npState_3^4 \wedge npOne_3 \wedge npHead_3 (Zero_3 \leftrightarrow nZero_3) \wedge (sZero_3 \leftrightarrow nsZero_3) \wedge (One_3 \leftrightarrow nOne_3) \wedge (sOne_3 \leftrightarrow nsOne_3)\Big).$$

Finally, the $NoHead$ term asserts in the absence of a head the tape contents do not change.

$$NoHead = (\neg pHead_3 \wedge \neg Head_3 \wedge \neg sHead_3) \rightarrow \Big((pOne_3 \leftrightarrow npOne_3) \wedge (nZero_3 \leftrightarrow npZero_3) \wedge (One_3 \leftrightarrow nOne_3) \wedge (Zero_3 \leftrightarrow nZero_3) \wedge (sOne_3 \leftrightarrow nsOne_3) \wedge (sZero_3 \leftrightarrow nsZero_3)\Big).$$

This brings us to the last player, who is trying to guess the first step and central cell chosen by Player Three:

$$\Phi_6 = \{Time_6^i\}_{1\leq i\leq k} \cup \{Tape_6^i\}_{1\leq i\leq k}.$$
$$\gamma_6 = \bigwedge_{i=1}^{k}(Tape_3^i \leftrightarrow Tape_6^i) \wedge \bigwedge_{i=1}^{k}(Time_3^i \leftrightarrow Time_6^i).$$

The construction so far has been polynomial. We now claim that M having an accepting run on w in at most K steps is equivalent to the constructed game having a Nash equilibrium where Players One, Two and Three have the following guaranteed payoffs:

$$\boldsymbol{v}[1] = \frac{2^{2k}-1}{2^{2k}}.$$
$$\boldsymbol{v}[2] = \frac{2^k(2^k-1)-1}{2^k(2^k-1)}.$$
$$\boldsymbol{v}[3] = \frac{(2^k-2)(2^k-1)-1}{(2^k-2)(2^k-1)}.$$

Strictly speaking, only the payoffs of Two and Three are necessary - Player One will be able to attain $\boldsymbol{v}[1]$ utility even if M does not accept w. However, we include One's utility above as it will help us to reason about the profile being in equilibrium.

First, suppose M has an accepting run on w in at most K steps. Consider the profile where Player One randomises over all step/cell combinations with equal weight, and at each step/cell combination plays his variables in accordance to the accepting run. Player Four also randomises over all step/cell combinations with equal weight. Player Two randomises over all computation steps and the first $2^k - 1$ cells. Her other variables she plays in accordance to the run. Player Five likewise randomises over all steps and the first $2^k - 1$ cells. Player Three randomises over the first $2^k - 1$ steps and the $2^k - 2$ cells between the first and last. His other variables he plays in accordance to the run. Player Six randomises over the same $2^k - 1$ steps and the $2^k - 2$ cells.

In such a profile, Players One, Two and Three will satisfy their goals unless their step/cell combination is guessed by their opponent. Given our setup, this will happen with probabilities $1/2^{2k}$, $1/2^k(2^k-1)$ and $1/(2^k-2)(2^k-1)$ respectively, giving us the payoffs $\boldsymbol{v}[1]$, $\boldsymbol{v}[2]$ and $\boldsymbol{v}[3]$. It remains to see that this profile is in equilibrium. Recall that this is equivalent to checking that no player has a pure strategy that will grant them more utility than what they are earning already.

Let us first consider Players Four through Six. Any pure strategy by Player Four is a step/cell pair, and hence, given the play of Player One, has a $1/2^{2k}$ chance of satisfying γ_4. Player Four is thus indifferent between the current situation and any deviation. For Player Five any pure strategy using the first $2^k - 1$ cells will have a $1/2^k(2^k - 1)$ chance of satisfying γ_5, and any other pure strategy 0. Player Five thus likewise has no incentive to deviate. In the same fashion, any pure strategy for Player Six will satisfy γ_6 with probability $1/(2^k - 2)(2^k - 1)$ or 0, so she is also indifferent.

In the case of Player One, observe that no matter what pure strategy he picks, there is a $1/2^{2k}$ chance of Player Four guessing the cell/step component and thus making γ_1 false. It follows that any such strategy will yield at most a $\boldsymbol{v}[1]$ chance of satisfying γ_1. For Player Two, if she picks a pure strategy using the first $2^k - 1$ cells there will likewise be a $1/2^k(2^k-1)$ chance of her step/cell combination being guessed. If she picks a pure strategy using the last cell, she will be unable to satisfy the $Succ_2$ component of γ_2, yielding a utility of 0. For Player Three, any pure strategy using the $2^k - 1$ steps and the $2^k - 2$ cells randomised over by six will have a $1/(2^k-2)(2^k-1)$ chance of being guessed, and any other choice of pure strategy will violate either $Triple$ or $Succ_3$. This establishes that the described profile is in equilibrium.

Next, suppose that no accepting run exists. We claim that in any equilibrium Player One will still obtain a utility of $\boldsymbol{v}[1]$, but either Player Two or three will be unable to secure a payoff of $\boldsymbol{v}[2]$, $\boldsymbol{v}[3]$. For the first part, note that for any choice of step/cell by Player One, the remaining variables can be set to satisfy $Init$, $Final$ and $Cons_1$ unilaterally. It is sufficient to simply respect the initial configuration of the machine at step zero, play an accepting state at step K, and any internally consistent description elsewhere. Any strategy that does not satisfy $Init$, $Final$ and $Cons_1$ is thus dominated and can be excluded from consideration. All that remains is the choice of cell/step and it is easy to see that the only equilibrium play would involve giving every pair equal weight.

Player One's play will thus describe a sequence of 2^k con-

figurations of M, with the initial configuration at step zero and an accepting state at step K. However, as M has no accepting run on w in K steps, this sequence cannot represent a valid computation and a violation must occur somewhere.

If this violation involves the assertion of the presence of more than one head or the $Left_1$, $Right_1$ variables incorrectly specifying the location of the head, we claim that Player Two cannot obtain a utility of $\boldsymbol{v}[2]$.

Observe that in this case there must exist two consecutive cells at some time step where Player One plays one of the following combinations:

Cell i	Cell $i+1$
$Left_1$	$Right_1$
$Left_1$	$Head_1$
$Head_1$	$Head_1$
$Right_1$	$Left_1$
$Head_1$	$Right_1$

In this case, should Player Two play a strategy involving cell i, since she is committed to playing a legal head assignment she will have to disagree with Player One on either cell i or cell $i+1$. This means she will suffer a $1/2^{2k}$ chance of having $MatchOne$ falsified if Player One plays the cell in question. As there is still at least a $1/2^k(2^k-1)$ chance of having the cell/step combination guessed by Player Five, this means the maximum utility Player Two can obtain in this case is $\boldsymbol{v}[2] - 1/2^{2k} + 1/2^{2k}2^k(2^k-1)$. (The last term is to avoid double counting the case where both Player One and Player Five name the same cell/step combination.)

Of course, Player Two may opt in this case not to play any strategies involving cell i. This will however mean that she is randomising over at most (2^k-2) cells, and Player Five will randomise accordingly, meaning the highest utility she can obtain is $\frac{2^k(2^k-2)-1}{2^k(2^k-2)}$.

Suppose now that Player One does not make such a violation. The remaining possibilities for an incorrect run are:

1. The head makes an illegal transition.
2. The tape contents undergo an illegal change.
3. The state undergoes an illegal change.

Let us deal with case 1. Suppose between step t and $t+1$ the head, which is at cell i at t, performs an illegal transition. This could mean moving more than one cell in a direction, moving off the edge of the tape, staying still in a non-accepting state or moving one cell left or right without a justifying transition rule. Observe that neither of these possibilities is consistent with the $Rules$ requirement. As such, should Player Three pick step t and cell i, he will have to disagree with Player One on the movement of the head, thereby running a risk of falsifying his formula should Player One play t and i. This will prevent Player Three from obtaining v_3 utility for the same reasoning as with Player Two.

In case 2, there would exist steps t and $t+1$, and a cell i the contents of which would change without a justifying rule. This, too, would violate $Rules$. For case 3, we note that by the machine state we mean the state variable that occurs in the same cell as the head: the value of the other state variables is of no account. As such, $Rules$ again would be violated as it requires the correct state to be propagated to cell hosting the head. This completes the proof. □

We can adapt this proof to show that ∀GUARANTEENASH is coNEXP-hard. Note that this does not follow immediately: ∀GUARANTEENASH is not simply the complement of ∃GUARANTEENASH. Letting $\boldsymbol{s}$ range over equilibrium profiles, ∀GUARANTEENASH is the question whether:

$$\forall \boldsymbol{s} \,.\, \forall i \,.\, u_i(\boldsymbol{s}) \geq \boldsymbol{v}[i]$$

the complement of ∀GUARANTEENASH is then:

$$\exists \boldsymbol{s} \,.\, \exists i \,.\, u_i(\boldsymbol{s}) < \boldsymbol{v}[i].$$

To show that ∀GUARANTEENASH is coNEXP-hard we need only show that the latter problem is NEXP-hard.

Corollary 1. ∀GUARANTEENASH *is coNEXP-hard.*

Proof. We argue that the proof of Theorem 2 can be adapted to show this. Note that the utilities of Players One, Four, Five and Six did not play a rôle in the proof. Those of Four, Five and Six were omitted entirely, whereas Player One has been seen to achieve $\boldsymbol{v}[1]$ utility in every equilibrium. What remains are Two and Three, and we will argue that those players could be collapsed into a single player.

Introduce a new player into the game constructed in the proof of Theorem 2, Player Seven, with $\gamma_7 = \gamma_2 \wedge \gamma_3$ and $\Phi_7 = \emptyset$. We argue that the Turing machine M accepts w in at most K steps if and only if there exists an $\boldsymbol{s}$ for which:

$$u_7(\boldsymbol{s}) \geq 1 - \frac{(2^k-2)(2^k-1) + 2^k(2^k-1) - 1}{2^k(2^k-1)(2^k-2)(2^k-1)}.$$

This can be seen by replicating the argument in the proof: in the presence of an accepting run, the only way Player Seven can lose utility is if Player Five or Six guesses the same cell/step, which happens with probabilities $\frac{1}{2^k(2^k-1)}$ and $\frac{1}{(2^k-2)(2^k-1)}$ respectively. Adding a term for double counting and simplifying yields the quantity above.

If we call that quantity $\boldsymbol{v}[7]$, we have established that $u_7(\boldsymbol{s}) \geq \boldsymbol{v}[7]$ if and only if M accepts w in at most K steps. If we set the other entries of $\boldsymbol{v}$ to 1 to make sure they do not interfere, we demonstrate that the following question is NEXP-hard:

$$\exists \boldsymbol{s} \,.\, \exists i \,.\, u_i(\boldsymbol{s}) \geq \boldsymbol{v}[i].$$

For the next step, add Player Eight with $\gamma_8 = \neg\gamma_7$ and $\Phi_8 = \emptyset$. As $u_8 = 1 - u_7$ the following question is NEXP-hard as well, letting $\boldsymbol{v}[8] = 1 - \boldsymbol{v}[7]$ and the other entries of $\boldsymbol{v}$ be 0:

$$\exists \boldsymbol{s} \,.\, \exists i \,.\, u_i(\boldsymbol{s}) \leq \boldsymbol{v}[i].$$

It remains to show that the inequality can be made strict.

First, observe that we can increase Player Seven's score, and hence decrease Player Eight's, by an arbitrarily small ϵ of a certain form: let $\gamma_7' = \gamma_7 \vee Pennies$ where $Pennies$ is a matching pennies game over a new set of variables Φ_7' against some new player. This will give Player Seven $1/2^{|\Phi_7'|}$ additional utility, minus a double counting term.

All that remains is to show that we can identify a "sufficiently small" ϵ. By this we mean an ϵ satisfying the following:

$$\exists \boldsymbol{s} \,.\, u_8(\boldsymbol{s}) - \epsilon < \boldsymbol{v}[8] \iff \exists \boldsymbol{s} \,.\, u_8(\boldsymbol{s}) \leq \boldsymbol{v}[8].$$

To see that this is possible, recall that if M does not accept w in K steps, then Player One necessarily specifies an incorrect computation history of the machine. As we have seen in the proof of Theorem 2, such a violation decreases the maximum attainable score of Player Two or Three by a fixed amount. It is thus possible to calculate the maximum attainable utility of Player Seven in the presence of such a violation, which will give us the bounds within which ϵ may reside.

This completes the proof. □

Discussion

The preceding proof raises two related questions. To begin with, one may ask whether six players are necessary. The answer is no, as we have hinted in the proof of the corollary. The reader may convince themselves that one may reduce the number to three in a straightforward fashion by collapsing Players Two and Three, and Four, Five and Six onto each other, in a similar fashion to the proof of the corollary. We used six players to simplify the exposition of the proof. Whether it is further possible to reduce the number to two is a different matter - and an interesting one, as we will see shortly

Second: whether there is a membership result to go with the hardness. Strictly speaking, there is not. As there exist games where every equilibrium requires irrational weights on the strategies chosen (Nash 1951; Bilò and Mavronicolas 2012) we cannot rely on the intuitive approach of guessing a strategy profile and checking whether it is in equilibrium.

One way this problem is addressed in the literature is to restrict attention to two player games, where a rational equilibrium *is* guaranteed to exist (Cottle and Dantzig 1968), which brings us back to the first question. The second way is to consider the notion of an ϵ-equilibrium: a profile of strategies where no player can gain more than ϵ utility by deviating. This problem, ϵ-$\exists$GUARANTEENASH, clearly does belong to NEXP, and the reader can convince themselves that by inserting a sufficiently small ϵ into the proof above we can establish that it is NEXP-complete.

Conclusion

We have shown that the problem of determining whether a Boolean game has a Nash equilibrium which guarantees each player a certain payoff is NEXP-hard. This is the first complexity result about mixed equilibria in the Boolean games framework, and demonstrates that in this instance Boolean games are as difficult as the more general class of Boolean circuit games.

The complexity of many other natural problems remains open, most significantly that of NASH: the task of computing a mixed equilibrium. However, given the difficulty in obtaining this result for normal form games (Daskalakis, Goldberg, and Papadimitriou 2006) one could posit that it is unlikely that this can be achieved with the current tools of complexity theory. It would be interesting to see whether there is an exponential time analogue of PPAD that could lead to a solution to this problem.

Acknowledgements

Egor Ianovski is supported by a scholarship, and Luke Ong is partially supported by a grant, from the Oxford-Man Institute of Quantitative Finance.

References

Ågotnes, T.; van Benthem, J.; van Ditmarsch, H. P.; and Minica, S. 2011. Question-answer games. *Journal of Applied Non-Classical Logics* 21(3-4):265–288.

Ågotnes, T.; Harrenstein, P.; van der Hoek, W.; and Wooldridge, M. 2013a. Boolean games with epistemic goals. In *LORI*, 1–14.

Ågotnes, T.; Harrenstein, P.; van der Hoek, W.; and Wooldridge, M. 2013b. Verifiable equilibria in boolean games. In *IJCAI*.

Bilò, V., and Mavronicolas, M. 2012. The complexity of decision problems about Nash equilibria in win-lose games. In Serna, M., ed., *Algorithmic Game Theory*, Lecture Notes in Computer Science. Springer Berlin Heidelberg. 37–48.

Bilò, V. 2007. On satisfiability games and the power of congestion games. In Kao, M.-Y., and Li, X.-Y., eds., *Algorithmic Aspects in Information and Management*, volume 4508 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 231–240.

Bonzon, E.; Lagasquie-Schiex, M.-C.; Lang, J.; and Zanuttini, B. 2006. Boolean games revisited. In *ECAI*, 265–269.

Bonzon, E.; Lagasquie-Schiex, M.-C.; and Lang, J. 2009. Dependencies between players in Boolean games. *Int. J. Approx. Reasoning* 50(6):899–914.

Bonzon, E.; Lagasquie-Schiex, M.-C.; and Lang, J. 2012. Effectivity functions and efficient coalitions in Boolean games. *Synthese* 187(1):73–103.

Cottle, R. W., and Dantzig, G. B. 1968. Complementary pivot theory of mathematical programming. *Linear Algebra and Its Applications* 1:103–125.

Daskalakis, C.; Goldberg, P. W.; and Papadimitriou, C. H. 2006. The complexity of computing a Nash equilibrium. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, 71–78. New York, NY, USA: ACM.

Dunne, P. E., and van der Hoek, W. 2004. Representation and complexity in Boolean games. In Alferes, J. J., and Leite, J. a., eds., *Logics in Artificial Intelligence*, volume 3229 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 347–359.

Dunne, P. E., and Wooldridge, M. 2012. Towards tractable Boolean games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '12, 939–946. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Dunne, P. E.; van der Hoek, W.; Kraus, S.; and Wooldridge, M. 2008. Cooperative Boolean games. In *AAMAS'08*, 1015–1022.

Endriss, U.; Kraus, S.; Lang, J.; and Wooldridge, M. 2011. Designing incentives for Boolean games. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, 79–86.

Feigenbaum, J.; Koller, D.; and Shor, P. 1995. A game-theoretic classification of interactive complexity classes (extended abstract). In *Proceedings of the tenth annual IEEE conference on computational complexity*, 227–237.

Harrenstein, P.; van der Hoek, W.; Meyer, J.-J.; and Witteveen, C. 2001. Boolean games. In *Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge*, TARK '01, 287–298. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kraus, S., and Wooldridge, M. 2012. Delegating decisions in strategic settings. In *ECAI 2012*, 468–473.

Levit, V.; Grinshpoun, T.; Meisels, A.; and Bazzan, A. L. C. 2013. Taxation search in boolean games. In *AAMAS*, 183–190.

Mavronicolas, M.; Monien, B.; and Wagner, K. W. 2007. Weighted Boolean formula games. In Deng, X., and Graham, F., eds., *Internet and Network Economics*, volume 4858 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 469–481.

Nash, J. 1951. Non-cooperative games. *Annals of Mathematics* 54(2):286–295.

Ågotnes, T., and van Ditmarsch, H. 2011. What will they say?public announcement games. *Synthese* 179(1):57–85.

Rosenthal, R. 1973. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory* 2(1):65–67.

Schoenebeck, G. R., and Vadhan, S. 2012. The computational complexity of Nash equilibria in concisely represented games. *ACM Trans. Comput. Theory* 4(2):4:1–4:50.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Nested Regular Path Queries in Description Logics

Meghyn Bienvenu
Lab. de Recherche en Informatique
CNRS & Univ. Paris Sud, France

Diego Calvanese
KRDB Research Centre
Free Univ. of Bozen-Bolzano, Italy

Magdalena Ortiz
Mantas Šimkus
Institute of Information Systems
Vienna Univ. of Technology, Austria

Abstract

Two-way regular path queries (2RPQs) have received increased attention recently due to their ability to relate pairs of objects by flexibly navigating graph-structured data. They are present in property paths in SPARQL 1.1, the new standard RDF query language, and in the XML query language XPath. In line with XPath, we consider the extension of 2RPQs with nesting, which allows one to require that objects along a path satisfy complex conditions, in turn expressed through (nested) 2RPQs. We study the computational complexity of answering nested 2RPQs and conjunctions thereof (CN2RPQs) in the presence of domain knowledge expressed in description logics (DLs). We establish tight complexity bounds in data and combined complexity for a variety of DLs, ranging from lightweight DLs (*DL-Lite*, $\mathcal{EL}$) up to highly expressive ones. Interestingly, we are able to show that adding nesting to (C)2RPQs does not affect worst-case data complexity of query answering for any of the considered DLs. However, in the case of lightweight DLs, adding nesting to 2RPQs leads to a surprising jump in combined complexity, from P-complete to EXP-complete.

1 Introduction

Both in knowledge representation and in databases, there has been great interest recently in expressive mechanisms for querying data, while taking into account complex domain knowledge (Calvanese, De Giacomo, and Lenzerini 2008; Glimm et al. 2008). Description Logics (DLs) (Baader et al. 2003), which on the one hand underlie the W3C standard Web Ontology Language (OWL), and on the other hand are able to capture at the intensional level conceptual modeling formalisms like UML and ER, are considered particularly well suited for representing a domain of interest (Borgida and Brachman 2003). In DLs, instance data, stored in a so-called ABox, is constituted by ground facts over unary and binary predicates (*concepts* and *roles*, respectively), and hence resembles data stored in graph databases (Consens and Mendelzon 1990; Barceló et al. 2012). There is a crucial difference, however, between answering queries over graph databases and over DL ABoxes. In the former, the data is assumed to be complete, hence query answering amounts to the standard database task of query evaluation. In the latter, it is typically assumed that the data is incomplete and additional domain knowledge is provided by the DL ontology (or TBox). Hence query answering amounts to the more complex task of computing *certain answers*, i.e., those answers that are obtained from all databases that both contain the explicit facts in the ABox and satisfy the TBox constraints. This difference has driven research in different directions.

In databases, expressive query languages for querying graph-structured data have been studied, which are based on the requirement of relating objects by flexibly navigating the data. The main querying mechanism that has been considered for this purpose is that of one-way and two-way regular path queries (RPQs and 2RPQs) (Cruz, Mendelzon, and Wood 1987; Calvanese et al. 2003), which are queries returning pairs of objects related by a *path* whose sequence of edge labels belongs to a regular language over the (binary) database relations and their inverses. Conjunctive 2RPQs (C2RPQs) (Calvanese et al. 2000) are a significant extension of such queries that add to the navigational ability the possibility of expressing arbitrary selections, projections, and joins over objects related by 2RPQs, in line with conjunctive queries (CQs) over relational databases. Two-way RPQs are present in the property paths in SPARQL 1.1 (Harris and Seaborne 2013), the new standard RDF query language, and in the XML query language XPath (Berglund and others 2010). An additional construct that is present in XPath is the possibility of using *existential test operators*, also known as *nesting*, to express sophisticated conditions along navigation paths. When an existential test $\langle E \rangle$ is used in a 2RPQ E', there will be objects along the main navigation path for E' that match positions of E' where $\langle E \rangle$ appears; such objects are required to be the origin of a path conforming to the (nested) 2RPQ E. It is important to notice that existential tests in general cannot be captured even by C2RPQs, e.g., when tests appear within a transitive closure of an RPQ. Hence, adding nesting effectively increases the expressive power of 2RPQs and of C2RPQs.

In the DL community, query answering has been investigated extensively for a wide range of DLs, with much of the work devoted to CQs. With regards to the complexity of query answering, attention has been paid on the one hand to *combined complexity*, i.e., the complexity measured considering as input both the query and the DL knowledge base (constituted by TBox and ABox), and on the other

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

hand to *data complexity*, i.e., when only the ABox is considered as input. For expressive DLs that extend $\mathcal{ALC}$, CQ answering is typically coNP-complete in data-complexity (Ortiz, Calvanese, and Eiter 2008), and 2EXP-complete in combined complexity (Glimm et al. 2008; Lutz 2008; Eiter et al. 2009). For lightweight DLs, instead, CQ answering is in AC^0 in data complexity for *DL-Lite* (Calvanese et al. 2007), and P-complete for $\mathcal{EL}$ (Krisnadhi and Lutz 2007). For both logics, the combined complexity is dominated by the NP-completeness of CQ evaluation over plain relational databases. There has also been some work on (2)RPQs and C(2)RPQs. For the very expressive DLs $\mathcal{ZIQ}$, $\mathcal{ZOQ}$, and $\mathcal{ZOI}$, where regular expressions over roles are present also in the DL, a 2EXP upper bound has been shown via techniques based on alternating automata over infinite trees (Calvanese, Eiter, and Ortiz 2009). For the Horn fragments of $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$, P-completeness in data complexity and EXP/2EXP-completeness in combined complexity are known (Ortiz, Rudolph, and Simkus 2011). For lightweight DLs, tight bounds for answering 2RPQs and C2RPQs have only very recently been established by Bienvenu, Ortiz, and Simkus (2013): for (C)(2)RPQs, data complexity is NL-complete in *DL-Lite* and *DL-Lite*$_\mathcal{R}$, and P-complete in $\mathcal{EL}$ and $\mathcal{ELH}$. For all of these logics, combined complexity is P-complete for (2)RPQs and PSPACE-complete for C(2)RPQs.

Motivated by the expressive power of nesting in XPath and SPARQL, in this paper we significantly advance these latter lines of research on query answering in DLs, and study the impact of adding nesting to 2RPQs and C2RPQs. We establish tight complexity bounds in data and combined complexity for a variety of DLs, ranging from lightweight DLs of the *DL-Lite* and $\mathcal{EL}$ families up to the highly expressive ones of the $\mathcal{SH}$ and $\mathcal{Z}$ families. Our results are summarized in Table 1. For DLs containing at least $\mathcal{ELI}$, we are able to encode away nesting, thus showing that the worst-case complexity of query answering is not affected by this construct. Instead, for lightweight DLs (starting already from *DL-Lite*!), we show that adding nesting to 2RPQs leads to a surprising jump in combined complexity, from P-complete to EXP-complete. We then develop a sophisticated rewriting-based technique that builds on (but significantly extends) the one proposed by Bienvenu, Ortiz, and Simkus (2013), which we use to prove that the problem remains in NL for *DL-Lite*. We thus show that adding nesting to (C)2RPQs does not affect worst-case data complexity of query answering for lightweight DLs.

For lack of space, some proofs have been relegated to the appendix of the long version (Bienvenu et al. 2014).

2 Preliminaries

We briefly recall the syntax and semantics of description logics (DLs). As usual, we assume countably infinite, mutually disjoint sets $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ of *concept names*, *role names*, and *individuals*. We typically use A for concept names, p for role names, and a, b for individuals. An *inverse role* takes the form p^- where $p \in \mathsf{N_R}$. We let $\mathsf{N_R^\pm} = \mathsf{N_R} \cup \{p^- \mid p \in \mathsf{N_R}\}$ and denote by r elements of $\mathsf{N_R^\pm}$.

A DL knowledge base (KB) consists of a *TBox* and an *ABox*, whose forms depend on the DL in question. In the DL $\mathcal{ELHI}_\bot$, a TBox is defined as a set of *(positive) role inclusions* of the form $r \sqsubseteq r'$ and *negative role inclusions* of the form $r \sqcap r' \sqsubseteq \bot$ with $r, r' \in \mathsf{N_R^\pm}$, and *concept inclusions* of the form $C \sqsubseteq D$, where C and D are *complex concepts* formed according to the following syntax:[1]

$$C ::= \top \mid \bot \mid A \mid \exists r.C \mid C \sqcap C$$

with $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R^\pm}$.

Some of our results refer specifically to the *lightweight DLs* that we define next. $\mathcal{ELHI}$ is the fragment of $\mathcal{ELHI}_\bot$ that has no $\bot$. $\mathcal{ELH}$ and $\mathcal{ELI}$ are obtained by additionally disallowing inverse roles and role inclusions, respectively. *DL-Lite*$_\mathcal{R}$ is also a fragment of $\mathcal{ELHI}_\bot$, in which concept inclusions can only take the forms $B_1 \sqsubseteq B_2$ and $B_1 \sqcap B_2 \sqsubseteq \bot$, for B_i a concept name or concept of the form $\exists r.\top$ with $r \in \mathsf{N_R^\pm}$. *DL-Lite* is the fragment of *DL-Lite*$_\mathcal{R}$ that disallows (positive and negative) role inclusions.

An *ABox* is a set of assertions of the form $C(a)$ or $r(a, b)$, where C is a complex concept, $r \in \mathsf{N_R^\pm}$, and $a, b \in \mathsf{N_I}$. We use $\mathsf{Ind}(\mathcal{A})$ to refer to the set of individuals in $\mathcal{A}$.

Semantics. The semantics of DL KBs is based upon *interpretations*, which take the form $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty set and $\cdot^\mathcal{I}$ maps each $a \in \mathsf{N_I}$ to $a^\mathcal{I} \in \Delta^\mathcal{I}$, each $A \in \mathsf{N_C}$ to $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, and each $p \in \mathsf{N_R}$ to $p^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$.[2] The function $\cdot^\mathcal{I}$ can be straightforwardly extended to complex concepts and roles. In the case of $\mathcal{ELHI}_\bot$, this is done as follows: $\top^\mathcal{I} = \Delta^\mathcal{I}$, $\bot^\mathcal{I} = \emptyset$, $(p^-)^\mathcal{I} = \{(c, d) \mid (d, c) \in p^\mathcal{I}\}$, $(\exists r.C)^\mathcal{I} = \{c \mid \exists d : (c, d) \in r^\mathcal{I}, d \in C^\mathcal{I}\}$, and $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$. An interpretation $\mathcal{I}$ satisfies an inclusion $G \sqsubseteq H$ if $G^\mathcal{I} \subseteq H^\mathcal{I}$, and it satisfies an assertion $C(a)$ (resp., $r(a, b)$) if $a^\mathcal{I} \in A^\mathcal{I}$ (resp., $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$). A *model* of a KB $(\mathcal{T}, \mathcal{A})$ is an interpretation $\mathcal{I}$ which satisfies all inclusions in $\mathcal{T}$ and assertions in $\mathcal{A}$.

Complexity. In addition to P and (co)NP, our results refer to the complexity classes NL (non-deterministic logarithmic space), PSPACE (polynomial space), and (2)EXP ((double) exponential time), cf. (Papadimitriou 1993).

3 Nested Regular Path Queries

We now introduce our query languages. In RPQs, nested RPQs and their extensions, *atoms* are given by (nested) regular expressions whose symbols are *roles*. The set Roles of roles contains $\mathsf{N_R^\pm}$, and all *test roles* of the forms $\{a\}?$ and $A?$ with $a \in \mathsf{N_I}$ and $A \in \mathsf{N_C}$. They are interpreted as $(\{a\}?)^\mathcal{I} = (a^\mathcal{I}, a^\mathcal{I})$, and $(A?)^\mathcal{I} = \{(o, o) \mid o \in A^\mathcal{I}\}$.

Definition 3.1. *A* nested regular expression *(NRE), denoted by E, is constructed according to the following syntax:*

$$E ::= \sigma \mid E \cdot E \mid E \cup E \mid E^* \mid \langle E \rangle$$

where $\sigma \in \mathsf{Roles}$.

[1] We slightly generalize the usual $\mathcal{ELHI}_\bot$ by allowing for negative role inclusions.

[2] Note that we do not make the *unique name assumption* (UNA), but all of our results continue to hold if the UNA is adopted.

	2RPQ		C2RPQ		N2RPQ / CN2RPQ	
	data	combined	data	combined	data	combined
Graph DBs & RDFS	NL-c	NL-c	NL-c	NP-c	NL-c	**P-c** / NP-c
DL-Lite	NL-c	P-c	NL-c	PSPACE-c	**NL-c**	**EXP-c**
Horn DLs (e.g., $\mathcal{EL}$, Horn-$\mathcal{SHIQ}$)	P-c	P-c	P-c	PSPACE-c	**P-c**	**EXP-c**
Expressive DLs (e.g., $\mathcal{ALC}$, $\mathcal{SHIQ}$)	coNP-h	EXP-c	coNP-h	2EXP-c	coNP-h	**EXP-c / 2EXP-c**

Table 1: Complexity of query answering. The 'c' indicates completeness, the 'h' hardness. New results are marked in bold. For existing results, refer to (Bienvenu, Ortiz, and Simkus 2013; Pérez, Arenas, and Gutierrez 2010; Barceló Baeza 2013; Calvanese, Eiter, and Ortiz 2009; Ortiz, Rudolph, and Simkus 2011) and references therein.

We assume a countably infinite set $\mathsf{N_V}$ *of* variables *(disjoint from* $\mathsf{N_C}$*,* $\mathsf{N_R}$*, and* $\mathsf{N_I}$*). Each* $t \in \mathsf{N_V} \cup \mathsf{N_I}$ *is a* term*. An* atom *is either a* concept atom *of the form* $A(t)$*, with* $A \in \mathsf{N_C}$ *and* t *a term, or a* role atom *of the form* $E(t, t')$*, with* E *an NRE and* t, t' *two (possibly equal) terms.*

A nested two-way regular path query *(N2RPQ)* $q(x, y)$ *is an atom of the form* $E(x, y)$*, where* E *is an NRE and* x, y *are two distinct variables. A* conjunctive N2RPQ *(CN2RPQ)* $q(\vec{x})$ *with* answer variables $\vec{x}$ *has the form* $\exists \vec{y}.\varphi$*, where* φ *is a conjunction of atoms whose variables are among* $\vec{x} \cup \vec{y}$*.*

A (plain) regular expression (RE) is an NRE that does not have subexpressions of the form $\langle E \rangle$. Two-way regular path queries *(2RPQs) and* conjunctive 2RPQs *(C2RPQs) are defined analogously to N2RPQs and CN2RPQs but allowing only plain REs in atoms.*

Given an interpretation $\mathcal{I}$, the semantics of an NRE E is defined by induction on its structure:

$$(E_1 \cdot E_2)^{\mathcal{I}} = E_1^{\mathcal{I}} \circ E_2^{\mathcal{I}},$$
$$(E_1 \cup E_2)^{\mathcal{I}} = E_1^{\mathcal{I}} \cup E_2^{\mathcal{I}},$$
$$(E_1^*)^{\mathcal{I}} = (E_1^{\mathcal{I}})^*,$$
$$\langle E \rangle^{\mathcal{I}} = \{(o, o) \mid \text{there is } o' \in \Delta^{\mathcal{I}} \text{ s.t. } (o, o') \in E^{\mathcal{I}}\}.$$

A *match* for a C2NRPQ $q(\vec{x}) = \exists \vec{y}.\varphi$ in an interpretation $\mathcal{I}$ is a mapping from the terms in φ to $\Delta^{\mathcal{I}}$ such that (i) $\pi(a) = a^{\mathcal{I}}$ for every individual a of φ, (ii) $\pi(x) \in A^{\mathcal{I}}$ for every concept atom $A(x)$ of φ, and (iii) $(\pi(x), \pi(y)) \in E^{\mathcal{I}}$ for every role atom $E(x, y)$ of φ. Let $\mathsf{ans}(q, \mathcal{I}) = \{\pi(\vec{x}) \mid \pi \text{ is a match for } q \text{ in } \mathcal{I}\}$. An individual tuple $\vec{a}$ with the same arity as $\vec{x}$ is called a *certain answer* to q over a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ if $(\vec{a})^{\mathcal{I}} \in \mathsf{ans}(q, \mathcal{I})$ for every model $\mathcal{I}$ of $\langle \mathcal{T}, \mathcal{A} \rangle$. We use $\mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ to denote the set of all certain answers to q over $\langle \mathcal{T}, \mathcal{A} \rangle$. In what follows, by *query answering*, we will mean the problem of deciding whether $\vec{a} \in \mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$.

Example 3.1. We consider an ABox of advisor relationships of PhD holders[3]. We assume an *advisor* relation between nodes representing academics. There are also nodes for theses, universities, research topics, and countries, related in the natural way via roles *wrote*, *subm*(itted), *topic*, and *loc*(ation). We give two queries over this ABox.

$$q_1(x, y) = (advisor \cdot \langle wrote \cdot topic \cdot Physics? \rangle)^* (x, y)$$

[3]Our examples are inspired by the *Mathematics Genealogy Project (http://genealogy.math.ndsu.nodak.edu/)*.

Query q_1 is an N2RPQ that retrieves pairs of a person x and an academic ancestor y of x such that all people on the path from x to y (including y itself) wrote a thesis in Physics.

$$q_2(x, y, z) = advisor^-(x, z), \quad advisor^*(x, w),$$
$$advisor^- \cdot \langle wrote \cdot \langle topic \cdot DBs? \rangle \cdot subm \cdot loc \cdot \{usa\}? \rangle (y, z),$$
$$\left(advisor \cdot \langle wrote \cdot \langle topic \cdot Logic? \rangle \cdot subm \cdot loc \cdot EU? \rangle\right)^* (y, w)$$

Query q_2 is a CN2RPQ that looks for triples of individuals x, y, z such that x and y have both supervised z, who wrote a thesis on Databases and who submitted this thesis to a university in the USA. Moreover, x and y have a common ancestor w, and all people on the path from x to w, including w, must have written a thesis in Logic and must have submitted this thesis to a university in an EU country. ■

It will often be more convenient to deal with an automata-based representation of (C)N2RPQs, which we provide next.

Definition 3.2. *A* nested NFA *(n-NFA) has the form* $(\mathbf{A}, s_0, F_0)$ *where* $\mathbf{A}$ *is an indexed set* $\{\alpha_1, \ldots, \alpha_n\}$*, where each* $\alpha_l \in \mathbf{A}$ *is an automaton of the form* (S, s, δ, F)*, where* S *is a set of states,* $s \in S$ *is the initial state,* $F \subseteq S$ *is the set of final states, and*

$$\delta \subseteq S \times (\mathsf{Roles} \cup \{\langle j_1, \ldots, j_k \rangle \mid l < j_i \leq n, \text{ for } i \in \{1, \ldots, k\}\}) \times S$$

We assume that the sets of states of the automata in $\mathbf{A}$ *are pairwise disjoint, and we require that* $\{s_0\} \cup F_0$ *are states of a single automaton in* $\mathbf{A}$*. If in each transition* $(s, \langle j_1, \ldots, j_k \rangle, s')$ *of each automaton in* $\mathbf{A}$ *we have* $k = 1$*, then the n-NFA is called* reduced*.*

When convenient notationally, we will denote an *n*-NFA $(\mathbf{A}, s_0, F_0)$ by $\mathbf{A}_{s_0, F_0}$. Moreover, we will use S_i, δ_i, and F_i to refer to the states, transition relation, and final states of α_i.

Definition 3.3. *Given an interpretation* $\mathcal{I}$*, we define* $\mathbf{A}^{\mathcal{I}}_{s_0, F_0}$ *inductively as follows. Let* α_l *be the (unique) automaton in* $\mathbf{A}$ *such that* $\{s_0\} \cup F_0 \subseteq S_l$*. Then* $(o, o') \in \mathbf{A}^{\mathcal{I}}_{s_0, F_0}$ *if there is a sequence* $s_0 o_0 s_1 \cdots o_{k-1} s_k o_k$*, for* $k \geq 0$*, such that* $o_0 = o$*,* $o_k = o'$*,* $s_k \in F_0$*, and for* $i \in \{1, \ldots, k\}$ *there is a transition* $(s_{i-1}, \sigma_i, s_i) \in \delta_l$ *such that either*

- $\sigma_i \in \mathsf{Roles}$ *and* $(o_{i-1}, o_i) \in \sigma_i^{\mathcal{I}}$*, or*
- $\sigma_i = \langle j_1, \ldots, j_k \rangle$ *such that, for every* $m \in \{1, \ldots, k\}$*, there exists* $o_m \in \Delta^{\mathcal{I}}$ *with* $(o_i, o_m) \in \mathbf{A}^{\mathcal{I}}_{s', F'}$*, where* s' *and* F' *are the initial and final states of* α_{j_m} *respectively.*

Note that an n-NFA $\mathbf{A}_{s_0,F_0}$ such that there are no transitions of the form $(s, \langle j_1, \ldots, j_k \rangle, s')$ in the unique α_l with $\{s_0\} \cup F_0 \subseteq S_l$ is equivalent to a standard NFA.

For every NRE E one can construct in polynomial time an n-NFA $\mathbf{A}_{s_0,F_0}$ such that $E^{\mathcal{I}} = \mathbf{A}^{\mathcal{I}}_{s_0,F_0}$ for every interpretation $\mathcal{I}$. This is an almost immediate consequence of the correspondence between regular expressions and finite state automata. Moreover, any n-NFA can be transformed into an equivalent reduced n-NFA by introducing linearly many additional states. In the following, unless stated otherwise, we assume all n-NFAs are reduced.

4 Upper Bounds via Reductions

In this section, we derive some upper bounds on the complexity of answering (C)N2RPQs in different DLs, by means of reductions to other problems. For simplicity, we assume in the rest of this section that query atoms do not employ test roles of the form $\{a\}?$. This is without loss of generality, since each symbol $\{a\}?$ can be replaced by $A_a?$ for a fresh concept name A_a, by adding the ABox assertion $A_a(a)$.

We start by showing that answering CN2RPQs can be polynomially reduced to answering non-nested C2RPQs using TBox axioms that employ inverses, conjunction on the left, and qualified existential restrictions.

Proposition 4.1. *For each CN2RPQ q, one can compute in polynomial time an $\mathcal{ELI}$ TBox $\mathcal{T}'$ and C2RPQ q' such that $\mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \mathsf{ans}(q', \langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \rangle)$ for every KB $\langle \mathcal{T}, \mathcal{A} \rangle$.*

Proof. Let q be an arbitrary CN2RPQ whose role atoms are given by n-NFAs, that is, they take the form $\mathbf{A}_{s_0,F_0}(x, y)$.

For each atom $\mathbf{A}_{s_0,F_0}(x, y)$ in q and each $\alpha_i \in \mathbf{A}$, we use a fresh concept name A_s for each state $s \in S_i$, and define a TBox $\mathcal{T}_{\alpha_i}$ that contains:

- $\top \sqsubseteq A_f$ for each $f \in F_i$,
- $\exists r.A_{s'} \sqsubseteq A_s$ for each $(s, r, s') \in \delta_i$ with $r \in \mathsf{N}_\mathsf{R}^\pm$,
- $A_{s'} \sqcap A \sqsubseteq A_s$ for each $(s, A?, s') \in \delta_i$ with $A \in \mathsf{N}_\mathsf{C}$, and
- $A_{s'} \sqcap A_{s_j} \sqsubseteq A_s$ for each $(s, \langle j \rangle, s') \in \delta_i$, with s_j the initial state of α_j.

We denote by $\mathcal{T}_\mathbf{A}$ the union of all $\mathcal{T}_{\alpha_i}$ with $\alpha_i \in \mathbf{A}$, and define $\mathcal{T}'$ as the union of $\mathcal{T}_\mathbf{A}$ for all atoms $\mathbf{A}_{s_0,F_0}(x, y) \in q$. To obtain the query q' we replace each atom $\mathbf{A}_{s_0,F_0}(x, y)$ by the atom $\alpha'_i(x, y)$, where α_i is the unique automaton in $\mathbf{A}$ with $\{s_0\} \cup F_0 \subseteq S_i$, and α'_i is obtained from α_i by replacing each transition of the form $(s, \langle j \rangle, s') \in \delta_i$ with $(s, A_{s_j}?, s')$, for s_j the initial state of α_j. Note that each α'_i is a standard NFA. We show in the appendix that $\mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \mathsf{ans}(q', \langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \rangle)$, for every KB $\langle \mathcal{T}, \mathcal{A} \rangle$. □

It follows that in every DL that contains $\mathcal{ELI}$, answering CN2RPQs is no harder than answering plain C2RPQs. From existing upper bounds for C2RPQs (Calvanese, Eiter, and Ortiz 2009; Ortiz, Rudolph, and Simkus 2011), we obtain:

Corollary 4.2. *Answering CN2RPQs is:*

- *in* 2EXP *in combined complexity for all DLs contained in $\mathcal{SHIQ}$, $\mathcal{SHOI}$, $\mathcal{ZIQ}$, or $\mathcal{ZOI}$.*
- *in* EXP *in combined complexity and* P *in data complexity for all DLs contained in Horn-$\mathcal{SHOIQ}$.*

We point out that the 2EXP upper bound for expressive DLs can also be inferred, without using the reduction above, from the existing results for answering C2RPQs in $\mathcal{ZIQ}$ and $\mathcal{ZOI}$ (Calvanese, Eiter, and Ortiz 2009).[4] Indeed, these DLs support regular role expressions as concept constructors, and a nested expression $\langle E \rangle$ in a query can be replaced by a concept $\exists E.\top$ (or by a fresh concept name A_E if the axiom $\exists E.\top \sqsubseteq A_E$ is added to the TBox). Hence, in $\mathcal{ZIQ}$ and $\mathcal{ZOI}$, nested expressions provide no additional expressiveness and CN2RPQs and C2RPQs coincide.

The construction used in Proposition 4.1 also allows us to reduce the evaluation of a N2RPQ to standard reasoning in any DL that contains $\mathcal{ELI}$.

Proposition 4.3. *For every N2RPQ q and every pair of individuals a, b, one can compute in polynomial time an $\mathcal{ELI}$ TBox $\mathcal{T}'$, and a pair of assertions $A_b(b)$ and $A_s(a)$ such that $(a, b) \in \mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ iff $\langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \cup \{A_b(b)\} \rangle \models A_s(a)$, for every DL KB $\langle \mathcal{T}, \mathcal{A} \rangle$.*

From this and existing upper bounds for instance checking in DLs, we easily obtain:

Corollary 4.4. *Answering N2RPQs is in* EXP *in combined complexity for every DL that contains $\mathcal{ELI}$ and is contained in $\mathcal{SHIQ}$, $\mathcal{SHOI}$, $\mathcal{ZIQ}$, or $\mathcal{ZOI}$.*

5 Lower Bounds

The upper bounds we have stated in Section 4 are quite general, and in most cases worst-case optimal.

The 2EXP upper bound stated in the first item of Corollary 4.2 is optimal already for C2RPQs and $\mathcal{ALC}$. Indeed, the 2EXP hardness proof for conjunctive queries in $\mathcal{SH}$ by Eiter et al. (2009) can be adapted to use an $\mathcal{ALC}$ TBox and a C2RPQ. Also the EXP bounds in Corollaries 4.2 and 4.4 are optimal for all DLs that contain $\mathcal{ELI}$, because standard reasoning tasks like satisfiability checking are already EXP-hard in this logic (Baader, Brandt, and Lutz 2008). For the same reasons, the P bound for data complexity in Corollary 4.2 is tight for $\mathcal{EL}$ and its extensions (Calvanese et al. 2006).

However, for the lightweight DLs *DL-Lite*$_\mathcal{R}$ and $\mathcal{EL}$, the best combined complexity lower bounds we have are NL (resp., P) for N2RPQs and PSPACE for CN2RPQs, inherited from the lower bounds for (C)NRPQs (Bienvenu, Ortiz, and Simkus 2013). This leaves a significant gap with respect to the EXP upper bounds in Corollaries 4.2 and 4.4.

We show next that these upper bounds are tight. This is the one of the core technical results of this paper, and probably the most surprising one: already evaluating one N2RPQ in the presence of a *DL-Lite* or $\mathcal{EL}$ TBox is EXP-hard.

Theorem 5.1. *In DL-Lite and $\mathcal{EL}$, N2RPQ answering is* EXP*-hard in combined complexity.*

Proof. We provide a reduction from the word problem for *Alternating Turing Machines (ATMs)* with polynomially

[4] For queries that do not contain inverse roles, that is, (1-way) CRPQs, the same applies to $\mathcal{ZOQ}$ and its sublogics.

bounded space, which is known to be EXP-hard (Chandra, Kozen, and Stockmeyer 1981). An ATM is given as a tuple $M = (\Sigma, S_\exists, S_\forall, \delta, s_{init}, s_{acc}, s_{rej})$, where Σ is an *alphabet*, $S_\exists$ is a set of *existential states*, $S_\forall$ is a set of *universal states*, $\delta \subseteq (S_\exists \cup S_\forall) \times \Sigma \cup \{\mathfrak{b}\} \times (S_\exists \cup S_\forall) \times \Sigma \cup \{\mathfrak{b}\} \times \{-1, 0, +1\}$ is a *transition relation*, $\mathfrak{b}$ is the blank symbol, and $s_{init}, s_{acc}, s_{rej} \in S_\exists$ are the *initial state*, the *acceptance state* and the *rejection state*, respectively.

Consider a word $w \in \Sigma^*$. We can w.l.o.g. assume that $\Sigma = \{0, 1\}$, that M uses only $|w|$ tape cells and that $|w| \geq 1$. Let $m = |w|$, and, for each $1 \leq i \leq m$, let $w(i)$ denote the ith symbol of w. Let $S = S_\exists \cup S_\forall$. We make the following further assumptions:

(i) The initial state is not a final state: $s_{init} \notin \{s_{acc}, s_{rej}\}$.
(ii) Before entering a state s_{acc} or s_{rej}, M writes $\mathfrak{b}$ in all m tape cells.
(iii) There exist functions $\delta_1, \delta_2 : S \times \Sigma \cup \{\mathfrak{b}\} \to S \times \Sigma \cup \{\mathfrak{b}\} \times \{-1, 0, +1\}$ such that $\{\delta_1(s, \sigma), \delta_2(s, \sigma)\} = \{(s', \sigma', d) \mid (s, \sigma, s', \sigma', d) \in \delta\}$ for every $s \in S \setminus \{s_{acc}, s_{rej}\}$ and $\sigma \in \Sigma \cup \{\mathfrak{b}\}$. In other words, non-final states of M give rise to exactly two successor configurations described by the functions δ_1, δ_2.

Note that the machine M can be modified in polynomial time to ensure (i-iii), while preserving the acceptance of w.

We next show how to construct in polynomial time a *DL-Lite* KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a query q such that M accepts w iff $a \in \mathsf{ans}(q, \mathcal{K})$ (we return to $\mathcal{EL}$ later). The high-level idea underlying the reduction is to use a KB to enforce a tree that contains all possible computations of M on w. The query q selects a computation in this tree and verifies that it corresponds to a proper, error-free, accepting run.

Generating the tree of transitions. First we construct $\mathcal{K}$, which enforces a tree whose edges correspond to the possible transitions of M. More precisely, each edge encodes a transition together with the resulting position of the read/write head of M, and indicates whether the transition is given by δ_1 or δ_2. This is implemented using role names $r_{p,t,i}$, where $p \in \{1, 2\}$, $t \in \delta$, and $0 \leq i \leq m + 1$. To mark the nodes that correspond to the initial (resp., a final) configuration of M, we employ the concept name A_{init} (resp., A_{final}), and we use $A_\exists$ and $A_\forall$ to store the transition type.

We let $\mathcal{A} = \{A_{init}(a), A_\exists(a)\}$, and then we initiate the construction of the tree by including in $\mathcal{T}$ the axiom

$$A_{init} \sqsubseteq \exists r_{p,(s_{init},\sigma,s',\sigma',d),1+d}. \tag{1}$$

for each $\sigma \in \Sigma \cup \{\mathfrak{b}\}$ and $p \in \{1, 2\}$ such that $\delta_p(s_{init}, \sigma) = (s', \sigma', d)$. To generate further transitions, $\mathcal{T}$ contains

$$\exists r^-_{p,(s,\sigma,s',\sigma',d),i} \sqsubseteq \exists r_{p',(s',\sigma^*,s'',\sigma'',d'),i+d'} \tag{2}$$

for each $(s, \sigma, s', \sigma', d) \in \delta$, $1 \leq i \leq m$, $\sigma^* \in \Sigma \cup \{\mathfrak{b}\}$ and $p, p' \in \{1, 2\}$ such that $\delta_{p'}(s', \sigma^*) = (s'', \sigma'', d')$. Note that a transition $t' = (s', \sigma^*, s'', \sigma'', d') \in \delta$ can follow $t = (s, \sigma, s', \sigma', d) \in \delta$ only if σ^* is the symbol written on tape cell i, for i the position of the read/write head after executing t. This is not guaranteed by (2). Instead, we "overestimate" the possible successive transitions, and use the query q to select paths that correspond to a proper computation.

We complete the definition of $\mathcal{T}$ by adding inclusions to label the nodes according to the type of states resulting from transitions. For each $1 \leq i \leq m$, $p \in \{1, 2\}$ and transition $(s, \sigma, s', \sigma', d) \in \delta$, we have the axiom

$$\exists r^-_{p,(s,\sigma,s',\sigma',d),i} \sqsubseteq A_Q, \text{ where}$$

- $A_Q = A_{final}$ if $s' \in \{s_{acc}, s_{rej}\}$,
- $A_Q = A_\exists$, if $s' \in S_\exists \setminus \{s_{acc}, s_{rej}\}$, and
- $A_Q = A_\forall$, if $s' \in S_\forall \setminus \{s_{acc}, s_{rej}\}$.

We turn to the construction of the query q, for which we employ the *n*-NFA representation. We construct an *n*-NFA $\alpha_q = (\mathbf{A}, s, F)$ where $\mathbf{A}$ has $m+1$ automata $\{\alpha_0, \ldots, \alpha_m\}$. Intuitively, the automaton α_0 will be responsible for traversing the tree representing candidate computation paths. At nodes corresponding to the end of a computation path, α_0 launches $\alpha_1, \ldots, \alpha_m$ which "travel" back to the root of the tree and test for the absence of errors along the way. We start by defining the tests $\alpha_1, \ldots, \alpha_m$. Afterwards we define α_0, which selects a set of paths that correspond to a full computation, and launches these tests at the end of each path.

Testing the correctness of a computation path. For each $1 \leq l \leq m$, the automaton $\alpha_l = (S_l, s_l, \delta_l, F_l)$ is built as follows. We let $S_l = \{\sigma_l \mid \sigma \in \Sigma\} \cup \{\mathfrak{b}_l\} \cup \{s'_l\}$. That is, S_l contains a copy of $\Sigma \cup \{\mathfrak{b}\}$ plus the additional state s'_l. We define the initial state as $s_l = \mathfrak{b}_l$ and let $F_l = \{s'_l\}$. Finally, the transition relation δ_l contains the following tuples:

(T1) $(\sigma_l, r^-_{p,(s,\sigma,s',\sigma',d),i}, \sigma_l)$ for all $1 \leq i \leq m$, $p \in \{1, 2\}$, all transitions $(s, \sigma, s', \sigma', d) \in \delta$, and each $\sigma_l \in S_l \setminus \{s'_l\}$ with $l \neq i - d$;
(T2) $(\sigma'_l, r^-_{p,(s,\sigma,s',\sigma',d),i}, \sigma_l)$ for all $1 \leq i \leq m$, $s \in S$ and $p \in \{1, 2\}$ with $\delta_p(s, \sigma) = (s', \sigma', d)$ and $l = i - d$;
(T3) $(\sigma_l, A_{init}?, s'_l)$ for $\sigma = w(l)$.

The working of α_l can be explained as follows. Each state $\sigma_l \in S_l \setminus \{s'_l\}$ corresponds to one of the symbols that may be written in position l of the tape during a run of M. When α_l is launched at some node in a computation tree induced by $\mathcal{K}$, it attempts to travel up to the root node, and the only reason it may fail is when a wrong symbol is written in position l at some point in the computation path. Recall that in each final configuration of M, all symbols are set to the blank symbol, and thus the initial state of α_l is $\mathfrak{b}_l$.

Consider a word $w' \in \mathsf{Roles}^*$ of the form

$$r^-_{p_k,t_k,i_k} \cdots r^-_{p_1,t_1,i_1} \cdot A_{init}? \tag{3}$$

that describes a path from some node in the tree induced by $\mathcal{K}$ up to the root node a. We claim that w' is accepted by every α_l ($1 \leq l \leq m$) just in the case that $t_1, \ldots, t_k$ is a correct sequence of transitions. To see why, first suppose that every α_l accepts w', and let $(pos_0, st_0, tape_0)$ be the tuple with $pos_0 = 1$, $st_0 = s_{init}$ and $tape_0$ contains for each $1 \leq l \leq m$, the symbol σ_l corresponding to the state of α_l when reading A_{init}. Clearly, due to (T3), the tuple $(pos_0, st_0, tape_0)$ describes the initial configuration of M on input w. For $1 \leq j \leq k$, if $t_j = (s, \sigma, s', \sigma', d)$, then we define $(pos_j, st_j, tape_j)$ as follows: $pos_j = i_j$, $st_j = s'$, and $tape_j$ contains for each $1 \leq i \leq m$, the state of α_i when reading $r^-_{p_j,t_j,i_j}$. A simple inductive argument

shows that for every $1 \leq j \leq k$, the tuple $(pos_j, st_j, tape_j)$ describes the configuration of M after applying the transitions $t_1, \ldots, t_j$ from the initial configuration. Indeed, let us assume that $(pos_{j-1}, st_{j-1}, tape_{j-1})$ correctly describes the configuration after executing $t_1, \ldots, t_{j-1}$ and $t_j = (s, \sigma, s', \sigma', d)$. After executing t_j, the read/write head is in position $pos_{j-1} + d$ and the state is s'. Since the only way to enforce an $r^-_{p_j,t_j,i_j}$-edge is via axioms (1) and (2), we must have $pos_j = pos_{j-1} + d$ and $st_j = s'$. It remains to show that $tape_j$ describes the tape contents after executing t_j. Consider some position $1 \leq l \leq m$. There are two cases:

1. $l \neq i_j - d$. In this case, we know that the symbol in position l is not modified by executing t_j. We have to show that $\sigma_l \in tape_{j-1}$ implies $\sigma_l \in tape_j$. This follows from the construction of α_l. In particular, when reading $r_{p_j,t_j,i_j}{}^-$, it must employ a transition from (T1).
2. $l = i_j - d$. In this case, after executing t_j, we must have σ' in position l. We have to show that $\sigma_l \in tape_{j-1}$ implies $\sigma'_l \in tape_j$. This again follows from the construction of α_l. In particular, when reading $r_{p_j,t_j,i_j}{}^-$, there is only one possible transition available in (T2), namely $(\sigma'_l, r_{p_j,t_j,i_j}{}^-, \sigma_l)$.

Conversely, it is easy to see that any word of the form (3) that appears in the tree induced by $\mathcal{K}$ and represents a correct computation path will be accepted by all of the α_l.

Selecting a proper computation. It remains to define α_0, which selects a subtree corresponding to a full candidate computation of M, and then launches the tests defined above at the end of each path. We let $\alpha_0 = (S_0, s_0, \delta_0, F_0)$, where $S_0 = \{s_\downarrow, c_L, c_R, s_\uparrow, s_\updownarrow, s_{test}, s_f\}$, $s_0 = s_\downarrow$, $F_0 = \{s_f\}$, and δ_0 is defined next.

The automaton operates in two main modes: moving down the tree away from the root and moving back up towards the root. Depending on the type of the state of M, in state $s_\downarrow$ the automaton either selects a child node to process next, or chooses to launch the test automata. If the tests are successful, it switches to moving up. To this end, δ_0 has the following transitions:

$$(s_\downarrow, A_\exists?, c_L), (s_\downarrow, A_\exists?, c_R), (s_\downarrow, A_\forall?, c_L),$$
$$(s_\downarrow, A_{final}?, s_{test}), \text{ and } (s_{test}, \langle 1, \ldots, m\rangle, s_\uparrow).$$

The transitions that implement a step down or up are:

- $(c_L, r_{1,t,i}, s_\downarrow)$ for every $1 \leq i \leq m$ and $t \in \delta$,
- $(c_R, r_{2,t,i}, s_\downarrow)$ for every $1 \leq i \leq m$ and $t \in \delta$,
- $(s_\uparrow, r^-_{1,t,i}, s_\updownarrow)$ for every $1 \leq i \leq m$ and $t \in \delta$, and
- $(s_\uparrow, r^-_{2,t,i}, s_\uparrow)$ for every $1 \leq i \leq m$ and $t \in \delta$.

After making a step up from the state $s_\uparrow$ via an $r^-_{1,t,i}$-edge, the automaton enters the state $s_\updownarrow$. Depending on the encountered state of M, the automaton decides either to verify the existence of a computation tree for the alternative transition, to keep moving up, or to accept the word. This is implemented using the following transitions of δ_0:

$$(s_\updownarrow, ?A_\forall, c_R), (s_\updownarrow, ?A_\exists, s_\uparrow), \text{ and } (s_\updownarrow, ?A_{init}, s_f).$$

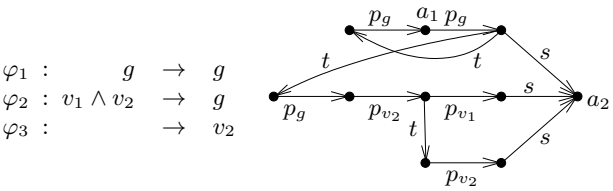


Figure 1: Example ABox in the proof of Theorem 5.2

To conclude the definition of $\alpha_q = (\mathbf{A}, s, F)$, set $s = s_\downarrow$ and $F = \{s_f\}$. Note that α_q has a constant number of states, so it can be converted into an equivalent NRE E_q in polynomial time. The desired query is $q(x, y) = E_q(x, y)$.

The above *DL-Lite* TBox $\mathcal{T}$ can be easily rephrased in $\mathcal{EL}$. Indeed, we simply take a fresh concept name $A_{p,t,i}$ for each role $r_{p,t,i}$, and replace every axiom $C \sqsubseteq \exists r_{p,t,i}$ by $C \sqsubseteq \exists r_{p,t,i}.A_{p,t,i}$ and every axiom $\exists r^-_{p,t,i} \sqsubseteq C$ by $A_{p,t,i} \sqsubseteq C$. $\square$

The above lower bound for answering N2RPQs hinges on the support for existential concepts in the right-hand-side of inclusions. If they are disallowed, then one can find a polynomial time algorithm (Pérez, Arenas, and Gutierrez 2010). However, it was open until now whether the polynomial-time upper bound is optimal. We next prove P-hardness of the problem, already for plain graph databases.

Theorem 5.2. *Given as input an N2RPQ q, a finite interpretation $\mathcal{I}$ and a pair $(o, o') \in \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, it is* P*-hard to check whether $(o, o') \in \mathsf{ans}(q, \mathcal{I})$.*

Proof. To simplify the presentation, we prove the lower bound for a slight reformulation of the problem. In particular, we show P-hardness of deciding $\vec{c} \in \mathsf{ans}(q, \langle \emptyset, \mathcal{A}\rangle)$, where q is an N2RPQ and $\mathcal{A}$ is an ABox with assertions only of the form $A(a)$ or $r(a, b)$, where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$.

We provide a logspace reduction from the classical P-complete problem of checking entailment in propositional definite Horn theories. Assume a set $T = \{\varphi_1, \ldots, \varphi_n\}$ of definite clauses over a set of propositional variables V, where each φ_i is represented as a rule $v_1 \wedge \ldots \wedge v_m \rightarrow v_{m+1}$.

Given a variable $g \in V$, we define an ABox $\mathcal{A}$, an N2RPQ q, and tuple (a_1, a_2) such that $T \models g$ iff $(a_1, a_2) \in \mathsf{ans}(q, \langle \emptyset, \mathcal{A}\rangle)$. We may assume w.l.o.g. that $\varphi_1 = g \rightarrow g$. We define the desired ABox as $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, using the role names s, t, and p_v, where $v \in V$. The ABox $\mathcal{A}_1$ simply encodes T and contains for every $\varphi_i = v_1 \wedge \ldots \wedge v_m \rightarrow v_{m+1}$, the following assertions:

$$p_{v_{m+1}}(e^i_{m+1}, e^i_m), \ldots, p_{v_1}(e^i_1, e^i_0), s(e^i_0, f).$$

The ABox $\mathcal{A}_2$ links variables in rule bodies with their occurrences in rule heads. For every pair of rules $\varphi_i = v_1 \wedge \ldots \wedge v_m \rightarrow v_{m+1}$ and $\varphi_j = w_1 \wedge \ldots \wedge w_n \rightarrow w_{n+1}$, and each $1 \leq l \leq m$ with $v_l = w_{n+1}$, it contains the assertion $t(e^i_{l-1}, e^j_{n+1})$. See Figure 1 for an example.

The existence of a proof tree for g, which can be limited to depth $|V|$, is expressed using the query $q(x, y) = E_{|V|}(x, y)$, with $E_1, E_2, \ldots, E_{|V|}$ defined inductively:

$$E_1 = \bigcup_{v \in V} (p_v \cdot t \cdot p_v) \cdot s$$

$$E_i = \bigcup_{v \in V} \big(p_v \cdot t \cdot p_v\big) \cdot \big(\langle E_{i-1} \rangle \cdot \bigcup_{v \in V} p_v\big)^* \cdot s \qquad (i > 1)$$

Finally, we let $a_1 = e_1^1$ and $a_2 = f$. □

6 Concrete Approach for Horn DLs

Our complexity results so far leave a gap for the data complexity of the *DL-Lite* family: we inherit NL-hardness from plain RPQs, but we only have the P upper bound stemming from Proposition 4.1. In this section, we close this gap by providing an NL upper bound.

This section has an additional goal. We recall that the upper bounds in Corollaries 4.2 and 4.4 rely on reductions to answering (C)2RPQs in extensions of $\mathcal{ELI}$, like Horn-$\mathcal{SHOIQ}$, $\mathcal{ZIQ}$, and $\mathcal{ZOI}$. Unfortunately, known algorithms for C2RPQ answering in these logics use automata-theoretic techniques that are best-case exponential and not considered suitable for implementation. Hence, we want to provide a direct algorithm that may serve as a basis for practicable techniques. To this end, we take an existing algorithm for answering C2RPQs in $\mathcal{ELH}$ and *DL-Lite*$_\mathcal{R}$ due to Bienvenu et al. (2013) and show how it can be extended to handle CN2RPQs and $\mathcal{ELHI}_\bot$ KBs.

For presenting the algorithm in this section, it will be useful to first recall the canonical model property of $\mathcal{ELHI}_\bot$.

Canonical Models

We say that an $\mathcal{ELHI}_\bot$ TBox $\mathcal{T}$ is in *normal form* if all of its concept inclusions are of one of the following forms:

$$A \sqsubseteq \bot \quad A \sqsubseteq \exists r.B \quad \top \sqsubseteq A \quad B_1 \sqcap B_2 \sqsubseteq A \quad \exists r.B \sqsubseteq A$$

with $A, B, B_1, B_2 \in \mathsf{N_C}$ and $r \in \mathsf{N_R^\pm}$.

By introducing fresh concept names to stand for complex concepts, every TBox $\mathcal{T}$ can be transformed in polynomial time into a TBox $\mathcal{T}'$ in normal form that is a model-conservative extension of $\mathcal{T}$. Hence, in what follows, we assume that $\mathcal{ELHI}_\bot$ TBoxes are in normal form.

The domain of the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ of a consistent KB $\langle \mathcal{T}, \mathcal{A} \rangle$ consists of all sequences $ar_1C_1 \dots r_nC_n$ ($n \geq 0$) such that:

- $a \in \mathsf{Ind}(\mathcal{A})$ and $r_i \in \mathsf{N_R^\pm}$ for each $1 \leq i \leq n$;
- each C_i is a finite conjunction of concept names;
- if $n \geq 1$, then $\mathcal{T}, \mathcal{A} \models (\exists r_1.C_1)(a)$;
- for $1 \leq i < n$, $\mathcal{T} \models C_i \sqsubseteq \exists r_{i+1}.C_{i+1}$.

For an $o \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A})$, we use $\mathsf{tail}(o)$ to denote its final concept. The interpretation $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is then defined as follows:

$$\begin{aligned}
a^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= a \text{ for all } a \in \mathsf{Ind}(\mathcal{A}) \\
A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T}, \mathcal{A} \models A(a)\} \\
&\quad \cup \{o \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} \setminus \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T} \models \mathsf{tail}(o) \sqsubseteq A\} \\
p^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} &= \{(a,b) \mid p(a,b) \in \mathcal{A}\} \cup \\
&\quad \{(o_1,o_2) \mid o_2 = o_1 r\, C \text{ and } \mathcal{T} \models r \sqsubseteq p\} \cup \\
&\quad \{(o_2,o_1) \mid o_2 = o_1 r\, C \text{ and } \mathcal{T} \models r \sqsubseteq p^-\}
\end{aligned}$$

Observe that $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is composed of a core part containing the individuals from $\mathcal{A}$ and an *anonymous part* consisting of (possibly infinite) trees rooted at the ABox individuals. We use $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_o$ to denote the restriction of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ to domain elements having o as a prefix.

It is well-known that the canonical model of a consistent $\mathcal{ELHI}_\bot$ KB $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ can be homomorphically embedded into any model of $\langle \mathcal{T}, \mathcal{A} \rangle$. Since CN2RPQs are preserved under homomorphisms, we have:

Lemma 6.1. *For every consistent $\mathcal{ELHI}_\bot$ KB $\langle \mathcal{T}, \mathcal{A} \rangle$, CN2RPQ q, and tuple $\vec{a}$ of individuals: $\vec{a} \in \mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ if and only if $\vec{a} \in \mathsf{ans}(q, \mathcal{I}_{\mathcal{T},\mathcal{A}})$.*

Computing Jump and Final Transitions

A crucial component of our algorithm is to compute relevant partial paths in a subtree $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_o$ rooted at an object o in the anonymous part of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. Importantly, we also need to remember which parts of the nested automata that have been partially navigated below o still need to be continued. This will allow us to 'forget' the tree below o.

In what follows, it will be convenient use *runs* to talk about the semantics of *n*-NFAs.

Definition 6.1. *Let $\mathcal{I}$ be an interpretation, and let $(\mathbf{A}, s_0, F_0)$ be an n-NFA. Then a* partial run for $\mathbf{A}$ on $\mathcal{I}$ *is a finite node-labelled tree (T, ℓ) such that every node is labelled with an element from $\Delta^\mathcal{I} \times (\bigcup_i S_i)$ and for each non-leaf node v having label $\ell(v) = (o, s)$ with $s \in S_i$, one of the following holds:*

- *v has a unique child v' with $\ell(v') = (o', s')$, and there exists $(s, \sigma, s') \in \delta_i$ such that $\sigma \in \mathsf{Roles}$ and $(o, o') \in \sigma^\mathcal{I}$;*
- *v has exactly two children v' and v'' with $\ell(v') = (o, s')$ and $\ell(v'') = (o, s'')$, with s'' the initial state of α_j, and there exists a transition $(s, \langle j \rangle, s') \in \delta_i$.*

*If T has root labelled (o_1, s_1) and a leaf node labelled (o_2, s_2) with s_1, s_2 states of the same α_i, then (T, ℓ) is called an (o_1, s_1, o_2, s_2)-*run*, and it is* full *if every leaf label $(o', s') \neq (o_2, s_2)$ is such that $s' \in F_k$ for some k.*

Full runs provide an alternative characterization of the semantics of *n*-NFAs in Definition 3.3.

Fact 6.2. *For every interpretation $\mathcal{I}$, $(o_1, o_2) \in (\mathbf{A}_{s_1,\{s_2\}})^\mathcal{I}$ if and only if there is a full (o_1, s_1, o_2, s_2)-run for $\mathbf{A}$ in $\mathcal{I}$.*

We use partial runs to characterize when an *n*-NFA $\mathbf{A}$ can be partially navigated inside a tree $\mathcal{I}_{\mathcal{T},\mathcal{A}}|_o$ whose root satisfies some conjunction of concepts C. Intuitively, $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ stores pairs s_1, s_2 of states of some $\alpha \in \mathbf{A}$ such that a path from s_1 to s_2 exists, while $\mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ stores states s_1 for which a path to some final state exists, no matter where the final state is reached. Both $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ and $\mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ store a set Γ of states s of other automata nested in α, for which a path from s to a final state remains to be found.

Definition 6.2. *Let $\mathcal{T}$ be an $\mathcal{ELHI}_\bot$ TBox in normal form and $(\mathbf{A}, s_0, F_0)$ an n-NFA. The set $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ consists of tuples (C, s_1, s_2, Γ) where C is either $\top$ or a conjunction of concept names from $\mathcal{T}$, s_1 and s_2 are states from $\alpha_i \in \mathbf{A}$, and $\Gamma \subseteq \bigcup_{j>i} S_j$. A tuple (C, s_1, s_2, Γ) belongs to $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ if there exists a partial run (T, ℓ) of $\mathbf{A}$ in the canonical model of $\langle \mathcal{T}, \{C(a)\} \rangle$ that satisfies the following conditions:*

- *the root of T is labelled (a, s_1);*
- *there is a leaf node v with $\ell(v) = (a, s_2)$;*
- *for every leaf node v with $\ell(v) = (o, s) \neq (a, s_2)$, either $s \in F_j$ for some $j > i$, or $o = a$ and $s \in \Gamma$.*

The set $\mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ contains all tuples (C, s_1, F, Γ) there is a partial run (T, ℓ) of $\mathbf{A}$ in the canonical model of $\langle \mathcal{T}, \{C(a)\}\rangle$ that satisfies the following conditions:

- *the root of T is labelled (a, s_1);*
- *there is a leaf node v with $\ell(v) = (o, s_f)$ and $s_f \in F$;*
- *for every leaf node v with $\ell(v) = (o, s)$, either s is a final state in some α_k, or $o = a$ and $s \in \Gamma$.*

Proposition 6.3. *It can be decided in exponential time if a tuple belongs to $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ or $\mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$.*

Proof idea. We first show how to use TBox reasoning to decide whether $(C, s_1, s_2, \Gamma) \in \mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$. For every $\alpha_j \in \mathbf{A}$, we introduce a fresh concept name A_s for each state $s \in S_j$. Intuitively, A_s expresses that there is an outgoing path that starts in s and reaches a final state. If $\{s_1, s_2\} \subseteq S_i$, then we add the following inclusions to $\mathcal{T}$:

- $\top \sqsubseteq A_s$, for every $s \in F_j$ with $j > i$;
- $\exists r.A_{s'} \sqsubseteq A_s$, whenever $(s, r, s') \in \delta_i$ with $r \in \mathsf{N}_\mathsf{R}^\pm$;
- $A_{s'} \sqcap B \sqsubseteq A_s$, whenever $(s, B?, s') \in \delta_i$;
- $A_{s'} \sqcap A_{s''} \sqsubseteq A_s$, whenever $(s, \langle j \rangle, s') \in \delta_i$ and s'' is the initial state of α_j.

Let $\mathcal{T}'$ be the resulting TBox. In the long version, we show that $(C, s_1, s_2, \Gamma) \in \mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ iff

$$\mathcal{T}' \models (C \sqcap A_{s_2} \sqcap \prod_{s \in \Gamma} A_s) \sqsubseteq A_{s_1}.$$

To decide if $(C, s_1, F, \Gamma) \in \mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$, we must also include in $\mathcal{T}'$ the following inclusions:

- $\top \sqsubseteq A_s$, for every $s \in F$.

We then show that $(C, s_1, F, \Gamma) \in \mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ iff

$$\mathcal{T}' \models (C \sqcap \prod_{s \in \Gamma} A_s) \sqsubseteq A_{s_1}.$$

To conclude the proof, we simply note that both problems can be decided in single-exponential time, as TBox reasoning in $\mathcal{ELHI}_\bot$ is known to be EXP-complete. □

Query Rewriting

The core idea of our query answering algorithm is to rewrite a given CN2RPQ q into a set of queries Q such that the answers to q and the union of the answers for all $q' \in Q$ coincide. However, for evaluating each $q' \in Q$, we only need to consider mappings from the variables to the individuals in the core of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$. Roughly, a rewriting step makes some assumptions about the query variables that are mapped deepest into the anonymous part and, using the structure of the canonical model, generates a query whose variables are matched one level closer to the core. Note that, even when we assume that no variables are mapped below some element o in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$, the satisfaction of the regular paths may require to go below o and back up in different ways. This is handled using jump and final transitions. The query rewriting algorithm is an adaptation of the algorithm for C2RPQs in (Bienvenu, Ortiz, and Simkus 2013), to which the reader may refer for more detailed explanations and examples.

The query rewriting algorithm is presented in Figure 2. In the algorithm, we use atoms of the form $\langle A_{s,F} \rangle(x)$, which are semantically equivalent to $A_{s,F}(x, z)$ for a variable z not occurring anywhere in the query. This alternative notation will spare us additional variables and make the complexity arguments simpler. To slightly simplify the notation, we may write $\mathbf{A}_{s,s'}$ instead of $\mathbf{A}_{s,\{s'\}}$.

The following proposition states the correctness of the rewriting procedure. Its proof follows the ideas outlined above and can be found in the appendix of the long version. Slightly abusing notation, we will also use $\mathsf{Rewrite}(q, \mathcal{T})$ to denote the set all of queries that can be obtained by an execution of the rewriting algorithm on q and $\mathcal{T}$.

Proposition 6.4. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHI}_\bot$ KB and $q(\vec{x})$ a C2NRPQ. Then $\vec{a} \in \mathsf{ans}(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ iff there exists $q' \in \mathsf{Rewrite}(q, \mathcal{T})$ and a match π for q' in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that $\pi(\vec{x}) = \vec{a}$ and $\pi(y) \in \mathsf{Ind}(\mathcal{A})$ for every variable y in q'.*

We note that the query rewriting does not introduce fresh terms. Moreover, it employs an at most quadratic number of linearly sized *n*-NFAs, obtained from the *n*-NFAs of the input query. Thus, the size of each $q' \in \mathsf{Rewrite}(q, \mathcal{T})$ is polynomial in the size of q and $\mathcal{T}$. Given that all the employed checks in Figure 2 can be done in exponential time (see Proposition 6.3), we obtain the following.

Proposition 6.5. *The set $\mathsf{Rewrite}(q, \mathcal{T})$ can be computed in exponential time in the size of q and $\mathcal{T}$.*

Query Evaluation

In Figure 3, we present an algorithm EvalAtom for evaluating N2RPQs. The idea is similar to the standard nondeterministic algorithm for deciding reachability: we guess a sequence $(c_0, s_0)(c_1, s_1) \cdots (c_m, s_m)$ of individual-state pairs, keeping only two successive elements in memory at any time. Every element (c_{i+1}, s_{i+1}) must be reached from the preceding element (c_i, s_i) by a single normal, jump, or final transition. Moreover, in order to use a jump or final transition, we must ensure that its associated conditions are satisfied. To decide if the current individual belongs to C, we can employ standard reasoning algorithms, but to determine whether an outgoing path exists for one of the states in Γ, we must make a recursive call to EvalAtom. Importantly, these recursive calls involve "lower" automata, and so the depth of recursion is bounded by the number of automata in the N2RPQ (and so is independent of $\mathcal{A}$). It follows that the whole procedure can be implemented in non-deterministic logarithmic space in $|\mathcal{A}|$, if we discount the concept and role membership tests. By exploiting known complexity results for instance checking in *DL-Lite*$_\mathcal{R}$ and $\mathcal{ELHI}_\bot$, we obtain:

Proposition 6.6. EvalAtom *is a sound and complete procedure for N2RPQ evaluation over satisfiable $\mathcal{ELHI}_\bot$ KBs. It can be implemented so as to run in non-deterministic logarithmic space (resp., polynomial time) in the size of the ABox for DL-Lite*$_\mathcal{R}$ *(resp., $\mathcal{ELHI}_\bot$) KBs.*

PROCEDURE Rewrite

Input: CN2RPQ q, $\mathcal{ELHI}_\bot$ TBox $\mathcal{T}$ in normal form

1. Choose either to output q or to continue.
2. Choose a non-empty set $\mathsf{Leaf} \subseteq \mathsf{vars}(q)$ and $y \in \mathsf{Leaf}$. Rename all variables in Leaf to y.
3. Choose a conjunction C of concept names from $\mathcal{T}$ such that $\mathcal{T} \models C \sqsubseteq B$ whenever $B(y)$ is an atom of q. Drop all such atoms from q.
4. For each atom $at \in q$ of the form $\langle \mathbf{A}_{s_0,F} \rangle(t)$ or $\mathbf{A}_{s_0,F}(t,t')$ with $y \in \{t,t'\}$:
   (a) let $\alpha_i \in \mathbf{A}$ be the automaton containing s_0, F
   (b) choose a sequence $s_1, \dots, s_{n-1}$ of distinct states from S_i and some $s_n \in F$
   (c) replace at by the atoms $\mathbf{A}_{s_0,s_1}(t,y)$, $\mathbf{A}_{s_1,s_2}(y,y)$, $\dots$, $\mathbf{A}_{s_{n-2},s_{n-1}}(y,y)$, and
   - $\mathbf{A}_{s_{n-1},s_n}(y,t')$ if $at = \mathbf{A}_{s_0,F}(t,t')$, or
   - $\langle \mathbf{A}_{s_{n-1},s_n} \rangle(y)$ if $at = \langle \mathbf{A}_{s_0,F} \rangle(y)$.
5. For each atom at_j of the form $\mathbf{A}_{s_j,s_{j+1}}(y,y)$ or $\langle \mathbf{A}_{s_j,s_{j+1}} \rangle(y)$ in q, either do nothing, or:
   - Choose some $(C, s_j, s_{j+1}, \Gamma) \in \mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ if $at_j = \mathbf{A}_{s_j,s_{j+1}}(y,y)$.
   - Choose some $(C, s_j, \{s_{j+1}\}, \Gamma) \in \mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ if $at_j = \langle \mathbf{A}_{s_j,s_{j+1}} \rangle(y)$.
   - Replace at_j by $\{\langle \mathbf{A}_{u,F_k} \rangle(y) \mid u \in \Gamma \cap S_k, \}$.
6. Choose a conjunction D of concept names from $\mathcal{T}$ and $r, r_1, r_2 \in \mathsf{N}_\mathsf{R}^\pm$ such that:
   (a) $\mathcal{T} \models D \sqsubseteq \exists r.C$, $\mathcal{T} \models r \sqsubseteq r_1$, and $\mathcal{T} \models r \sqsubseteq r_2$.
   (b) For each atom $\mathbf{A}_{u,U}(y,t)$ of q with $u \in S_i$, there exists $v \in S_i$ such that $(u, r_1^-, v) \in \delta_i$
   (c) For each atom $\mathbf{A}_{u,U}(t,y)$ of q with $u \in S_i$, there exists $v \in S_i$ and $v' \in U$ with $(v, r_2, v') \in \delta_i$.
   (d) For each atom $\langle \mathbf{A}_{u,U} \rangle(y)$ of q with $u \in S_i$, there exists $v \in S_i$ such that $(u, r_1^-, v) \in \delta_i$.

   For atoms $\mathbf{A}_{u,U}(y,y)$, both (b) and (c) apply.
7. Replace
   - each atom $\mathbf{A}_{u,U}(y,t)$ with $t \neq y$ by $\mathbf{A}_{v,U}(y,t)$,
   - each atom $\mathbf{A}_{u,U}(t,y)$ with $t \neq y$ by $\mathbf{A}_{u,v}(y,t)$,
   - each atom $\mathbf{A}_{u,U}(y,y)$ by atom $\mathbf{A}_{v,v'}(y,y)$, and
   - each atom $\langle \mathbf{A}_{u,U} \rangle(y)$ by atom $\langle \mathbf{A}_{v,U} \rangle(y)$

   with v, v' as in Step 6.
8. Add $A(y)$ to q for each $A \in D$ and return to Step 1.

Figure 2: Query rewriting procedure Rewrite.

We present in Figure 4 the complete procedure EvalQuery for deciding CN2RPQ entailment.

Theorem 6.7. EvalQuery *is a sound and complete procedure for deciding CN2RPQ entailment over $\mathcal{ELHI}_\bot$ KBs. In the case of DL-Lite$_\mathcal{R}$ KBs, it runs in non-deterministic logarithmic space in the size of the ABox.*

Proof idea. Soundness, completeness, and termination of

PROCEDURE EvalAtom

Input: n-NFA $(\mathbf{A}, s_0, F_0)$, $\mathcal{ELHI}_\bot$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ in normal form, $(a,b) \in \mathsf{Ind}(\mathcal{A}) \times (\mathsf{Ind}(\mathcal{A}) \cup \{\mathsf{anon}\})$

1. Let i be such that $s_0 \in S_i$, and set $\mathsf{max} = |\mathcal{A}| \times |S_i| + 1$.
2. Initialize $\mathsf{current} = (a, s_0)$ and $\mathsf{count} = 0$.
3. While $\mathsf{count} < \mathsf{max}$ and $\mathsf{current} \neq (b, s_f)$ for $s_f \in F_0$
   (a) Let $\mathsf{current} = (c, s)$.
   (b) Guess a pair $(d, s') \in (\mathsf{Ind}(\mathcal{A}) \cup \{\mathsf{anon}\}) \times S_i$ such that one of the following holds:
      i. $d \in \mathsf{Ind}(\mathcal{A})$ and there exists $(s, \sigma, s') \in \delta_i$ with $\sigma \in \mathsf{Roles}$ such that $(c,d) \in \sigma^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$
      ii. $d = c$ and $\mathsf{JumpTrans}(\mathbf{A}, \mathcal{T})$ contains a tuple (C, s, s', Γ) such that $c \in C^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and for every $j > i$ and every $u \in \Gamma \cap S_j$,
      $$\mathsf{EvalAtom}((\mathbf{A}, u, F_j), \mathcal{K}, (c, \mathsf{anon})) = \mathtt{yes}$$
      iii. $d = \mathsf{anon}$, $s' \in F_0$, and $\mathsf{FinalTrans}(\mathbf{A}, \mathcal{T})$ contains a tuple (C, s, F_0, Γ) such that $c \in C^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ and for every $j > i$ and every $u \in \Gamma \cap S_j$,
      $$\mathsf{EvalAtom}((\mathbf{A}, u, F_j), \mathcal{K}, (c, \mathsf{anon})) = \mathtt{yes}$$
   (c) Set $\mathsf{current} = (d, s')$ and increment count.
4. If $\mathsf{current} = (d, s_f)$ for some $s_f \in F_0$, and either $b = d$ or $b = \mathsf{anon}$, return yes. Else return no.

Figure 3: N2RPQ evaluation procedure EvalAtom.

PROCEDURE EvalQuery

Input: Boolean CN2RPQ q, $\mathcal{ELHI}_\bot$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ in normal form

1. Test whether $\mathcal{K}$ is satisfiable, output yes if not.
2. Set $Q = \mathsf{Rewrite}(q, \mathcal{T})$. Replace all atoms in Q of types $C(a)$, $R(a,b)$ by equivalent atoms of type $\mathbf{A}_{s_0,F_0}(a,b)$.
3. Guess some $q' \in Q$ and an assignment $\vec{a}$ of individuals to the quantified variables $\vec{v}$ in q'
   - Let q'' be obtained by substituting $\vec{a}$ for $\vec{v}$.
   - For every atom $\mathbf{A}_{s_0,F_0}(a,b)$ in q''
     check if $\mathsf{EvalAtom}((\mathbf{A}, s_0, F_0), \mathcal{K}, (a,b)) = \mathtt{yes}$
   - If all checks succeed, return yes.
4. Return no.

Figure 4: CN2RPQ entailment procedure EvalQuery.

EvalQuery follow easily from the corresponding properties of the component procedures Rewrite and EvalAtom (Propositions 6.4, 6.5, and 6.6). In *DL-Lite$_\mathcal{R}$*, KB satisfiability is known to be NL-complete in data complexity. Since the rewriting step is ABox-independent, the size of queries in $\mathcal{Q}$ can be treated as a constant. It follows that the query q' and assignment $\vec{a}$ guessed in Step 3 can be stored in logarithmic space in $|\mathcal{A}|$. By Theorem 6.7, each call to EvalAtom runs in non-deterministic logarithmic space. □

Corollary 6.8. *CN2RPQ entailment over DL-Lite$_\mathcal{R}$ knowl-*

edge bases is NL*-complete in data complexity.*

7 Conclusions and Future Work

We have studied the extension of (C)2RPQs with a nesting construct inspired by XPath, and have characterized the data and combined complexity of answering nested 2RPQs and C2RPQs for a wide range of DLs. The only complexity bound we leave open is whether the coNP lower-bound in data complexity for expressive DLs is tight; indeed, the automata-theoretic approach used to obtain optimal bounds in combined complexity for these logics does not seem to provide the right tool for tight bounds in data complexity.

In light of the surprising jump from P to EXP in the combined complexity of answering N2RPQs in lightweight DLs, a relevant research problem is to identify classes of N2RPQs that exhibit better computational properties. We are also interested in exploring whether the techniques developed in Section 6 can be extended to deal with additional query constructs, such as existential "loop-tests" or forms of role-value maps. Finally, containment of N2RPQs has been studied very recently (Reutter 2013), but only for plain graph databases, so it would be interesting to investigate containment also in the presence of DL constraints.

Acknowledgments. This work has been partially supported by ANR project PAGODA (ANR-12-JS02-007-01), by the EU IP Project FP7-318338 *Scalable End-user Access to Big Data* (Optique), by the FWF projects T515-N23 and P25518-N23, and by the WWTF project ICT12-015.

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Baader, F.; Brandt, S.; and Lutz, C. 2008. Pushing the $\mathcal{EL}$ envelope further. In *Proc. of OWLED*.

Barceló Baeza, P. 2013. Querying graph databases. In *Proc. of PODS*, 175–188.

Barceló, P.; Libkin, L.; Lin, A. W.; and Wood, P. T. 2012. Expressive languages for path queries over graph-structured data. *ACM TODS* 37(4):31.

Berglund, A., et al. 2010. XML Path Language (XPath) 2.0 (Second Edition). W3C Recommendation. Available at http://www.w3.org/TR/xpath20.

Bienvenu, M.; Calvanese, D.; Ortiz, M.; and Simkus, M. 2014. Nested Regular Path Queries in Description Logics. CoRR Technical Report arXiv:1402.7122. Available at http://arxiv.org/abs/1402.7122.

Bienvenu, M.; Ortiz, M.; and Simkus, M. 2013. Conjunctive regular path queries in lightweight description logics. In *Proc. of IJCAI*.

Borgida, A., and Brachman, R. J. 2003. Conceptual modeling with description logics. In Baader et al. (2003). chapter 10, 349–372.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. Y. 2000. Containment of conjunctive regular path queries with inverse. In *Proc. of KR*, 176–185.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Vardi, M. Y. 2003. Reasoning on regular path queries. *SIGMOD Record* 32(4):83–92.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2006. Data complexity of query answering in description logics. In *Proc. of KR*, 260–270.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *JAR* 39(3):385–429.

Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2008. Conjunctive query containment and answering under description logics constraints. *ACM TOCL* 9(3):22.1–22.31.

Calvanese, D.; Eiter, T.; and Ortiz, M. 2009. Regular path queries in expressive description logics with nominals. In *Proc. of IJCAI*, 714–720.

Chandra, A. K.; Kozen, D.; and Stockmeyer, L. J. 1981. Alternation. *JACM* 28(1):114–133.

Consens, M. P., and Mendelzon, A. O. 1990. GraphLog: a visual formalism for real life recursion. In *Proc. of PODS*, 404–416.

Cruz, I. F.; Mendelzon, A. O.; and Wood, P. T. 1987. A graphical query language supporting recursion. In *Proc. of ACM SIGMOD*, 323–330.

Eiter, T.; Lutz, C.; Ortiz, M.; and Simkus, M. 2009. Query answering in description logics with transitive roles. In *Proc. of IJCAI*, 759–764.

Glimm, B.; Lutz, C.; Horrocks, I.; and Sattler, U. 2008. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. *JAIR* 31:157–204.

Harris, S., and Seaborne, A. 2013. SPARQL 1.1 Query Language. W3C Recommendation, World Wide Web Consortium. Available at http://www.w3.org/TR/sparql11-query.

Hustadt, U.; Motik, B.; and Sattler, U. 2005. Data complexity of reasoning in very expressive description logics. In *Proc. of IJCAI*. 466–471

Krisnadhi, A., and Lutz, C. 2007. Data complexity in the $\mathcal{EL}$ family of description logics. In *Proc. of LPAR*, 333–347.

Lutz, C. 2008. The complexity of conjunctive query answering in expressive description logics. In *Proc. of IJCAR*, volume 5195 of *LNAI*, 179–193. Springer.

Ortiz, M.; Calvanese, D.; and Eiter, T. 2008. Data complexity of query answering in expressive description logics via tableaux. *JAR* 41(1):61–98.

Ortiz, M.; Rudolph, S.; and Simkus, M. 2011. Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In *Proc. of IJCAI*, 1039–1044.

Papadimitriou, C. H. 1993. *Computational Complexity*. Addison Wesley.

Pérez, J.; Arenas, M.; and Gutierrez, C. 2010. nSPARQL: A navigational language for RDF. *J. of Web Semantics* 8(4):255–270.

Reutter, J. L. 2013. Containment of nested regular expressions. CoRR Technical Report arXiv:1304.2637, arXiv.org e-Print archive. Available at http://arxiv.org/abs/1304.2637.

Decidable Gödel Description Logics without the Finitely-Valued Model Property*

Stefan Borgwardt and **Felix Distel** and **Rafael Peñaloza**†
Theoretical Computer Science, TU Dresden, Germany
{stefborg,felix,penaloza}@tcs.inf.tu-dresden.de

Abstract

In the last few years, there has been a large effort for analyzing the computational properties of reasoning in fuzzy description logics. This has led to a number of papers studying the complexity of these logics, depending on the chosen semantics. Surprisingly, despite being arguably the simplest form of fuzzy semantics, not much is known about the complexity of reasoning in fuzzy description logics w.r.t. witnessed models over the Gödel t-norm. We show that in the logic G-$\mathfrak{I}\mathcal{ALC}$, reasoning cannot be restricted to finitely-valued models in general. Despite this negative result, we also show that all the standard reasoning problems can be solved in exponential time, matching the complexity of reasoning in classical $\mathcal{ALC}$.

1 Introduction

Fuzzy Description Logics (DLs) have been studied as a means of representing vague or imprecise knowledge in a formal and well-understood manner. As for classical DLs (Baader et al. 2007), knowledge is expressed with the help of *concepts* and *roles*. What distinguishes fuzzy DLs from classical DLs are their semantics, which are based on *fuzzy sets*. Fuzzy sets associate every element of the domain of interest with a number from the interval $[0, 1]$, which intuitively represents the *degree* to which the element belongs to the fuzzy set.

When defining a fuzzy DL, one must also decide how to interpret the logical constructors, such as conjunction and implication, to handle the truth degrees. The simplest approach is to use the *minimum* operator to generalize intersection to fuzzy sets. Thus, the degree of membership of a conjunction is interpreted as the minimum of the membership degrees of the conjuncts. This operation, also known as the *Gödel t-norm*, can be used as a base to interpret all other logical constructors in a formally justified manner (Klement, Mesiar, and Pap 2000; Hájek 2001). The quantifiers $\forall$ and $\exists$ are interpreted as infima and suprema of truth values, respectively. To avoid issues arising from having infinitely many truth values, reasoning in fuzzy DLs is usually restricted to so-called *witnessed models* (Hájek 2007).

The study of fuzzy DLs underwent a large change in recent years, after some relatively inexpressive fuzzy DLs were shown to be undecidable when reasoning w.r.t. general ontologies (Baader and Peñaloza 2011a; 2011b; Cerami and Straccia 2013). Since then, the limits of decidability have been explored, yielding very expressive decidable logics on the one hand (Borgwardt, Distel, and Peñaloza 2012), and inexpressive undecidable logics on the other (Borgwardt and Peñaloza 2012). Despite being widely regarded as the simplest t-norm, surprisingly little is known about fuzzy DLs based on Gödel semantics. It is generally believed that—at least w.r.t. witnessed models—these logics are decidable, but no proof exists to support this claim. The only results for similar logics restrict reasoning *a priori* to a finite subset of $[0, 1]$; in this case, a reduction to classical reasoning then yields decidability (Bobillo et al. 2009; 2012).

All existing approaches for reasoning in fuzzy DLs depend on limiting models to use only finitely many different truth degrees. Indeed, for these approaches to work, one must either (i) restrict the semantics to a finite set of truth degrees (Bobillo et al. 2009; 2012; Bobillo and Straccia 2011; 2013; Borgwardt and Peñaloza 2013a; 2013b; Straccia 2006); (ii) prove that reasoning can be restricted to a finite set of degrees (Bobillo, Delgado, and Gómez-Romero 2008; Borgwardt, Distel, and Peñaloza 2012; Straccia 2001); or (iii) prove that models can be built from a finite pattern (Stoilos et al. 2007; Straccia and Bobillo 2007). In all three cases, the proofs of correctness of these algorithms imply the *finitely-valued model property*: an ontology has a model iff it has a model using only finitely many truth values. Conversely, the proofs of undecidability (Baader and Peñaloza 2011a; 2011b; Borgwardt and Peñaloza 2012; Cerami and Straccia 2013) construct a model that uses infinitely many truth degrees. Thus, this *finitely-valued model property* appears to be a good indicator of the decidability of a fuzzy DL.

In this paper we study the standard reasoning problems for the DL G-$\mathfrak{I}\mathcal{ALC}$, a fuzzy extension of $\mathcal{ALC}$ based on the Gödel semantics w.r.t. witnessed models. First, we show that

*Partially supported by the DFG under grant BA 1122/17-1, in the Research Training Group 1763, the Cluster of Excellence 'Center for Advancing Electronics Dresden', and the Collaborative Research Center 912 'Highly Adaptive Energy-Efficient Computing'.

†Associated to the Center for Advancing Electronics Dresden.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

this logic does not have the finitely-valued model property. In fact, we provide very simple consistent ontologies that only have infinitely-valued models (see Section 3). The absence of the finitely-valued model property for these logics is a surprising result in itself, contradicting the common lore of the field. In contrast, we show in Sections 4 and 5 that consistency is decidable in exponential time for this logic. Our algorithm is based on the insight that under Gödel semantics, it is only necessary to know an ordering between the relevant truth degrees, rather the precise values they take. This idea has already been used for deciding validity of formulae in propositional Gödel logic (Guller 2012). We then extend our algorithm to also compute best subsumption degrees and best satisfiability degrees w.r.t. an ontology. The last section provides some pointers to future work.

2 Preliminaries

Before introducing fuzzy description logics, we briefly consider the operators of Gödel fuzzy logic and introduce auxiliary notions that will be useful for the reasoning procedures described in the following sections.

The two basic operators of Gödel fuzzy logic are conjunction and implication, interpreted by the *Gödel t-norm* and *residuum*, respectively. The Gödel t-norm of two fuzzy degrees $x, y \in [0,1]$ is defined as minimum function $\min(x,y)$. The residuum $\Rightarrow$ is uniquely defined by the equivalence $\min(x,y) \leq z$ iff $y \leq (x \Rightarrow z)$ for all $x, y, z \in [0,1]$, and can be computed as

$$x \Rightarrow y = \begin{cases} 1 & \text{if } x \leq y, \\ y & \text{otherwise.} \end{cases}$$

For a deeper introduction to t-norms and t-norm-based fuzzy logics, see (Cintula, Hájek, and Noguera 2011; Hájek 2001; Klement, Mesiar, and Pap 2000).

A *total preorder* over a set S is a transitive and total binary relation $\lesssim_* \subseteq S \times S$. For $x, y \in S$, we write $x \equiv_* y$ if $x \lesssim_* y$ and $y \lesssim_* x$. Notice that $\equiv_*$ is an equivalence relation on S. Similarly, we write $x <_* y$ if $x \lesssim_* y$, but not $y \lesssim_* x$. By the symbol $\bowtie$ we denote an arbitrary element of $\{=, \geq, >, \leq, <\}$, and by $\bowtie_*$ the corresponding relation induced by the total preorder $\lesssim_*$, i.e. $\equiv_*$, $\gtrsim_*$, $>_*$, $\lesssim_*$, or $<_*$. Subscripts are used to distinguish these relations for different total preorders over the same carrier set S.

An *order structure* S is a finite set containing at least the numbers 0 and 1, together with an involutive unary operation $\mathrm{inv}\colon S \to S$ such that $\mathrm{inv}(x) = 1 - x$ for all $x \in S \cap [0,1]$. For an order structure S, $\mathrm{order}(S)$ denotes the set of all total preorders $\lesssim_*$ over S that

- have 0 and 1 as least and greatest element, respectively,
- preserve the order of real numbers on $S \cap [0,1]$, and
- satisfy $x \lesssim_* y$ iff $\mathrm{inv}(y) \lesssim_* \mathrm{inv}(x)$ for all $x, y \in S$.

Given $\lesssim_* \in \mathrm{order}(S)$, the following functions on S that mimic the operators of Gödel fuzzy logic over $[0,1]$ are well-defined since $\lesssim_*$ is total:

$$\min_*(x,y) := \begin{cases} x & \text{if } x \lesssim_* y \\ y & \text{otherwise,} \end{cases}$$

Table 1: Semantics of G-$\mathfrak{I}\mathcal{ALC}$

Constructor	Syntax	Semantics
top concept	$\top$	1
involutive negation	$\neg C$	$1 - C^{\mathcal{I}}(x)$
conjunction	$C \sqcap D$	$\min(C^{\mathcal{I}}(x), D^{\mathcal{I}}(x))$
implication	$C \to D$	$C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x)$
existential restriction	$\exists r.C$	$\sup_{y \in \Delta^{\mathcal{I}}} \min(r^{\mathcal{I}}(x,y), C^{\mathcal{I}}(y))$
value restriction	$\forall r.C$	$\inf_{y \in \Delta^{\mathcal{I}}} r^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)$

$$\mathrm{res}_*(x,y) := \begin{cases} 1 & \text{if } x \lesssim_* y \\ y & \text{otherwise.} \end{cases}$$

It is easy to see that these operators agree with $\min$ and $\Rightarrow$ on the set $S \cap [0,1]$.

The fuzzy description logic G-$\mathfrak{I}\mathcal{ALC}$ is based on concepts and roles, which are interpreted as (fuzzy) unary and binary relations, respectively. Given the mutually disjoint sets $\mathsf{N_I}$, $\mathsf{N_R}$, and $\mathsf{N_C}$ of *individual*, *role*, and *concept names*, respectively, G-$\mathfrak{I}\mathcal{ALC}$ *concepts* are built through the rule

$$C ::= A \mid \top \mid \neg C \mid C \sqcap C \mid C \to C \mid \exists r.C \mid \forall r.C,$$

where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. We call concepts of the form $\exists r.C$ or $\forall r.C$ *quantified concepts*. The semantics of this logic is given by means of interpretations. An *interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty *domain*, and $\cdot^{\mathcal{I}}$ is a function that maps every $a \in \mathsf{N_I}$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, every $A \in \mathsf{N_C}$ to a fuzzy set $A^{\mathcal{I}}\colon \Delta^{\mathcal{I}} \to [0,1]$, and every role name $r \in \mathsf{N_R}$ to a fuzzy binary relation $r^{\mathcal{I}}\colon \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0,1]$. This function is extended to arbitrary concepts using the Gödel operators as shown in Table 1.

Notice that we have not introduced an explicit constructor for the *residual negation* $\ominus x := x \Rightarrow 0$ or disjunction, as they are expressible using $\top$, $\neg$, $\sqcap$, and $\to$. The residual negation is often used in fuzzy logics, but under Gödel semantics it is much less expressive than the involutive negation since we have $\ominus 0 = 1$ and $\ominus x = 0$ for all $x \in (0,1]$.

In the literature on fuzzy DLs, interpretations are usually restricted to be witnessed (Hájek 2005), which means that existential and value restrictions must be interpreted as maxima and minima, respectively. More formally, an interpretation $\mathcal{I}$ is *witnessed* if for every existential restriction $\exists r.C$ and every $x \in \Delta^{\mathcal{I}}$ there is a *witness* $y \in \Delta^{\mathcal{I}}$ such that $(\exists r.C)^{\mathcal{I}}(x) = \min(r^{\mathcal{I}}(x,y), C^{\mathcal{I}}(y))$, and similarly for value restrictions. We also adopt this restriction here, and for the rest of this paper consider only witnessed interpretations. For brevity, we call them simply *interpretations*.

The knowledge of a domain is represented using axioms that restrict the class of interpretations that are relevant for the different reasoning tasks.

Definition 1 (axioms). A *crisp assertion* is either a *concept assertion* of the form $a\colon C$ or a *role assertion* of the form $(a,b)\colon r$ for a concept C, $r \in \mathsf{N_R}$, and $a, b \in \mathsf{N_I}$. An *(order) assertion* is of the form $\langle \alpha \bowtie \beta \rangle$, where α is a crisp assertion and β is either a crisp assertion or a value from $[0,1]$. An interpretation $\mathcal{I}$ *satisfies* an order assertion $\langle \alpha \bowtie \beta \rangle$ if $\alpha^{\mathcal{I}} \bowtie \beta^{\mathcal{I}}$, where $(a\colon C)^{\mathcal{I}} := C^{\mathcal{I}}(a^{\mathcal{I}})$,

$((a,b):r)^{\mathcal{I}} := r^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}})$, and $q^{\mathcal{I}} := q$ for all $q \in [0,1]$. An *ordered ABox* $\mathcal{A}$ is a finite set of order assertions. An interpretation is a *model* of $\mathcal{A}$ if it satisfies all order assertions in $\mathcal{A}$.

A *general concept inclusion (GCI)* is an expression of the form $\langle C \sqsubseteq D \geq q \rangle$ for concepts C, D, and $q \in [0,1]$. An interpretation $\mathcal{I}$ satisfies this GCI if $C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x) \geq q$ holds for all $x \in \Delta^{\mathcal{I}}$. A *TBox* is a finite set of GCIs. An *ontology* is a pair $\mathcal{O} = (\mathcal{A}, \mathcal{T})$, where $\mathcal{A}$ is an ordered ABox and $\mathcal{T}$ is a TBox. An interpretation is a *model* of a TBox $\mathcal{T}$ if it satisfies all GCIs in $\mathcal{T}$, and it is a *model* of an ontology $\mathcal{O} = (\mathcal{A}, \mathcal{T})$ if it is a model of both $\mathcal{A}$ and $\mathcal{T}$.

We will usually abbreviate $\langle C \sqsubseteq D \geq 1 \rangle$ as $\langle C \sqsubseteq D \rangle$. Ordered ABoxes are more expressive than ABoxes usually considered for fuzzy DLs (Straccia 2001) since they allow to state order relations between concepts. This more general kind of ABox is better suited for our algorithms.

We denote by $\mathsf{sub}(\mathcal{O})$ the closure under negation of the set of all subconcepts appearing in an ontology $\mathcal{O}$. The concepts $\neg\neg C$ and C are equivalent, and we regard them here as equal, which means that $\mathsf{sub}(\mathcal{O})$ is always finite. We further denote by $\mathcal{V}_{\mathcal{O}}$ the closure of the set of all truth degrees appearing in $\mathcal{O}$, together with 0, 0.5, and 1, under the operator $x \mapsto 1 - x$. Since this operator is involutive, $\mathcal{V}_{\mathcal{O}}$ is also always finite. We often denote the elements of $\mathcal{V}_{\mathcal{O}} \subseteq [0,1]$ as $0 = q_0 < q_1 < \cdots < q_k = 1$.

As with classical DLs, the most basic reasoning task in G-$\mathfrak{I}\mathcal{ALC}$ is to decide whether a given ontology has a (witnessed) model. However, one might also be interested in computing the degree to which an entailment holds.

Definition 2 (reasoning). An ontology $\mathcal{O}$ is *consistent* if it has a model. Given $p \in [0,1]$, a concept C is *p-satisfiable* w.r.t. $\mathcal{O}$ if there is a model $\mathcal{I}$ of $\mathcal{O}$ and an $x \in \Delta^{\mathcal{I}}$ with $C^{\mathcal{I}}(x) \geq p$. The *best satisfiability degree* of C w.r.t. $\mathcal{O}$ is the supremum over all p such that C is p-satisfiable w.r.t. $\mathcal{O}$. Furthermore, C is *p-subsumed* by a concept D w.r.t. $\mathcal{O}$ if all models of $\mathcal{O}$ satisfy the GCI $\langle C \sqsubseteq D \geq p \rangle$. The *best subsumption degree* of C and D w.r.t. $\mathcal{O}$ is the supremum over all p such that C is p-subsumed by D w.r.t. $\mathcal{O}$.

If consistency is decidable, then satisfiability and subsumption can be restricted without loss of generality to ontologies containing an empty ABox: if $\mathcal{O}$ is inconsistent, then these two problems are trivial, and if $\mathcal{O}$ is consistent, then the ABox assertions cannot contradict the p-satisfiability of C, and therefore C is p-satisfiable w.r.t. $\mathcal{O} = (\mathcal{A}, \mathcal{T})$ iff it is p-satisfiable w.r.t. $(\emptyset, \mathcal{T})$. A similar argument can be made for subsumptions.

We show in Section 5 that ontology consistency has the same complexity in G-$\mathfrak{I}\mathcal{ALC}$ as in classical $\mathcal{ALC}$: it is ExpTime-complete. As a first step, we establish the complexity of consistency for the special case of ontologies with so-called *local* ordered ABoxes in Section 4, adapting an automata-based technique known from classical and finitely-valued DLs (Baader, Hladik, and Peñaloza 2008; Borgwardt and Peñaloza 2013a). We later lift these results to the satisfiability and subsumption problems. But first, we illustrate why the naïve approach of restricting reasoning to finitely-valued reasoning cannot work in this logic.

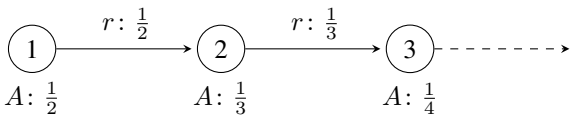


Figure 1: The model $\mathcal{I}_1$ from Example 3

3 Restricting to Finitely Many Values

It is a simple observation that any set of truth values that contains 0 and 1 is closed w.r.t. the Gödel connectives. Owing to this observation, it is common to restrict reasoning in fuzzy DLs with Gödel semantics to the finitely many truth values occurring in the ontology (Bobillo et al. 2012; Bobillo and Straccia 2013). This restriction is also sometimes justified by the "limited precision of computers" (Bobillo et al. 2009).

Earlier works have, however, neglected to study whether the restriction to a fixed finite set of values preserves the semantics of the logic. We now show that this is not the case, even for the simple description logics G-$\mathcal{AL}$, which allows only $\exists$, $\forall$, $\sqcap$, and $\top$. and G-$\mathfrak{IEL}$, where concepts are built using only $\exists$, $\sqcap$, $\rightarrow$, and $\top$. We show even stronger results: reasoning in these logics cannot, without loss of generality, be restricted to *finitely-valued* models, i.e. models that only use values from an arbitrary finite subset of $[0,1]$. We note that for the related *Zadeh semantics*, which differ from the Gödel semantics only in the operator used for implications, reasoning can be restricted to finitely-valued models without loss of generality (Straccia 2001; Bobillo, Delgado, and Gómez-Romero 2008).

Example 3. Let $\mathcal{T}_1$ be the G-$\mathcal{AL}$ TBox

$$\mathcal{T}_1 = \{\langle \forall r.A \sqsubseteq A \geq 1 \rangle,\ \langle \exists r.\top \sqsubseteq A \geq 1 \rangle\}.$$

We show that $\top$ is *not* 1-subsumed by A w.r.t. the ontology $\mathcal{O} = (\emptyset, \mathcal{T}_1)$, but every finitely-valued model of this ontology also satisfies $\langle \top \sqsubseteq A \geq 1 \rangle$.

For the former, we construct a model $\mathcal{I}_1$ of $\mathcal{T}_1$ as follows (see Figure 1). Let $\Delta^{\mathcal{I}_1}$ be the set of all natural numbers. We define $A^{\mathcal{I}_1}(n) := r^{\mathcal{I}_1}(n, n+1) := \frac{1}{n+1}$ for all $n \in \mathbb{N}$ and $r^{\mathcal{I}_1}(n,m) := 0$ if $m \neq n+1$. It is straightforward to check that this is a witnessed model of $\mathcal{T}_1$ that violates $\langle \top \sqsubseteq A \geq 1 \rangle$. Thus, $\top$ is not 1-subsumed by A w.r.t. $\mathcal{O}$. In fact, the best subsumption degree of $\top$ and A w.r.t. $\mathcal{O}$ is 0.

Assume now that there is a witnessed model $\mathcal{I}$ of $\mathcal{T}_1$ using only finitely many truth values that violates $\langle \top \sqsubseteq A \geq 1 \rangle$. Since $\mathcal{I}$ uses only finitely many truth values, there exists an element $y \in \Delta^{\mathcal{I}}$ for which $A^{\mathcal{I}}(y)$ is minimal, that is, $A^{\mathcal{I}}(y) \leq A^{\mathcal{I}}(x)$ holds for all $x \in \Delta^{\mathcal{I}}$. Furthermore, since $\mathcal{I}$ violates $\langle \top \sqsubseteq A \geq 1 \rangle$ there must be some $x_0 \in \Delta^{\mathcal{I}}$ satisfying $A^{\mathcal{I}}(x_0) < 1$. In particular, this yields $A^{\mathcal{I}}(y) < 1$.

As $\mathcal{I}$ is witnessed, there must exist a $z \in \Delta^{\mathcal{I}}$ such that $(\forall r.A)^{\mathcal{I}}(y) = r^{\mathcal{I}}(y,z) \Rightarrow A^{\mathcal{I}}(z)$. The first axiom of $\mathcal{T}_1$ entails $r^{\mathcal{I}}(y,z) \Rightarrow A^{\mathcal{I}}(z) \leq A^{\mathcal{I}}(y) < 1$, and in particular

$$r^{\mathcal{I}}(y,z) > A^{\mathcal{I}}(z). \tag{1}$$

The second axiom from $\mathcal{T}_1$ yields

$$r^{\mathcal{I}}(y,z) = \min(r^{\mathcal{I}}(y,z), 1) \leq (\exists r.\top)^{\mathcal{I}}(y) \leq A^{\mathcal{I}}(y). \tag{2}$$

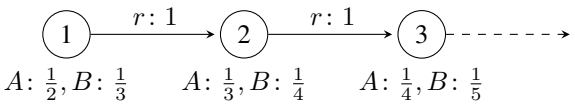


Figure 2: The model $\mathcal{I}_2$ from Example 4

From (1) and (2) we obtain $A^{\mathcal{I}}(y) > A^{\mathcal{I}}(z)$, contradicting the minimality of $A^{\mathcal{I}}(y)$. We have thus shown that a witnessed model of $\mathcal{T}_1$ with only finitely many truth values cannot violate $\langle \top \sqsubseteq A \geq 1 \rangle$. That is, $\mathcal{T}_1$ entails $\langle \top \sqsubseteq A \geq 1 \rangle$ when reasoning is restricted to finite sets of values.

It is thus not possible to restrict reasoning in G-$\mathcal{AL}$ to only finitely-valued models without changing the consequences. A similar example shows that this also holds for G-$\mathfrak{IEL}$.

Example 4. Consider the TBox

$$\mathcal{T}_2 = \{\langle B \sqsubseteq A \rangle, \langle A \to B \sqsubseteq B \rangle, \langle \top \sqsubseteq \exists r.\top \rangle, \langle \exists r.A \sqsubseteq B \rangle\}.$$

As in Example 3, one can show that $\top$ is not 1-subsumed by A w.r.t. $\mathcal{O} := (\emptyset, \mathcal{T}_2)$, but every finitely-valued model of $\mathcal{O}$ satisfies $\langle \top \sqsubseteq A \geq 1 \rangle$. A witnessed model $\mathcal{I}_2$ of $\mathcal{T}_2$ can be built as follows (see Figure 2). Let $\Delta^{\mathcal{I}_2}$ be the set of all natural numbers. Set $A^{\mathcal{I}_2}(n) := \frac{1}{n+1}$, $B^{\mathcal{I}_2}(n) := \frac{1}{n+2}$, $r^{\mathcal{I}_2}(n, n+1) := 1$ for all $n \in \mathbb{N}$, and $r^{\mathcal{I}_2}(n, m) := 0$ if $m \neq n+1$. It is straightforward to check that this is indeed a witnessed model of $\mathcal{T}_2$ that violates $\langle \top \sqsubseteq A \geq p \rangle$ for every $p > 0$; in particular for $p = 1$. Using a proof by contradiction similar to Example 3 it can be shown that no witnessed model of $\mathcal{T}_2$ with only finitely many truth values can violate $\langle \top \sqsubseteq A \geq 1 \rangle$. All details can be found in (Borgwardt, Distel, and Peñaloza 2013).

Recall that a (fuzzy) DL has the *finite model property* if every consistent ontology has a model with finite domain. A simple consequence of the last two examples is that G-$\mathcal{AL}$ and G-$\mathfrak{IEL}$ do not have the finite model property. Indeed, each $\mathcal{I}_i$ is a model of the ontology $(\{\langle a : A = 0.5 \rangle\}, \mathcal{T}_i)$ if we interpret the individual name a as $a^{\mathcal{I}_i} := 1$. This shows that these ontologies are consistent. However, any finite model $\mathcal{I}$ of $\mathcal{T}_i$ uses only finitely many truth degrees. As shown in the examples, such an interpretation must satisfy $A^{\mathcal{I}}(x) = 1$ for all $x \in \Delta^{\mathcal{I}}$, and hence violate the assertion $\langle a : A = 0.5 \rangle$. We thus obtain the following result.

Theorem 5. *G-$\mathcal{AL}$ and G-$\mathfrak{IEL}$ do not have the finite model property or the finitely-valued model property.*

The lack of the finitely-valued model property implies that some of the standard techniques used for reasoning in fuzzy DLs cannot be directly applied to any logic that contains G-$\mathcal{AL}$ or G-$\mathfrak{IEL}$. For example, termination of the tableaux-based approach (Stoilos et al. 2007; Straccia and Bobillo 2007) relies on the existence of finitely many *types* that can describe domain elements by specifying the membership degrees for all relevant concepts, while any sound and complete reduction to crisp reasoning (Bobillo, Delgado, and Gómez-Romero 2008; Bobillo et al. 2012) implies the finitely-valued model property.

Moreover, all known undecidability proofs for fuzzy DLs (Baader and Peñaloza 2011a; 2011b; Borgwardt and Peñaloza 2012; Cerami and Straccia 2013) are based on the fact that one can enforce models to have infinitely many values. One could thus be inclined to believe that consistency in G-$\mathfrak{IALC}$ is also undecidable. In the rest of this paper, we show that this is not the case, providing EXPTIME automata-based algorithms that decide consistency, subsumption, and satisfiability.

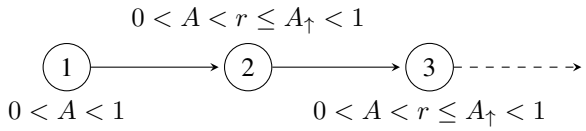


Figure 3: An abstract description of $\mathcal{I}_1$ from Example 3

4 Deciding Local Consistency

In this section, we consider only the special case where the ontology $\mathcal{O} = (\mathcal{A}, \mathcal{T})$ is such that $\mathcal{A}$ is a *local* ordered ABox, which means that it contains no role assertions and uses only a single individual name a. In Section 5, we extend the approach to handle arbitrary ontologies.

The algorithm is based on the observation that the axioms and the semantics of the constructors only introduce restrictions on the *order* of the values that models can assign to concepts, not on the values themselves. For example, an interpretation $\mathcal{I}$ satisfies an assertion $\langle a : (A \to B) = p \rangle$ iff $A^{\mathcal{I}}(a^{\mathcal{I}}) > B^{\mathcal{I}}(a^{\mathcal{I}})$ and $B^{\mathcal{I}}(a^{\mathcal{I}}) = p$. Thus, rather than building a model directly, we first create an abstract representation of a model that encodes for each domain element only the order between concepts.

Example 6. Consider again the TBox $\mathcal{T}_1$ from Example 3. When trying to construct a model violating $\langle \top \sqsubseteq A \geq 1 \rangle$, we start with a domain element satisfying the restriction that the value of A is strictly smaller than 1 (see Figure 3). The second axiom implies that the degree of any outgoing r-connection is bounded by the value of A. Moreover, the first axiom states that the witness of $\forall r.A$ must satisfy A to a degree strictly smaller than the value of the r-connection, and thus strictly smaller than the original value of A.

This yields an abstract description of two domain elements in terms of order relations between values of concepts at the current node and the parent node (denoted by a subscript $\uparrow$). Applying the same argument to the new element yields another element with the same restrictions. In order to construct a model, it is easy to see that the value of A at all considered elements has to be strictly greater than 0—once the value of A is 0, there can be no successors with smaller values for A. Note that it suffices to consider order relations between concepts of neighboring elements, which are directly connected by some role to a degree greater than 0.

As in this example, we use the subscript $\uparrow$ to refer to values of the parent node in the tree-like model that we will construct. We additionally use a new element λ to represent the degree of the role connection from the parent node.

Definition 7 (order structure $\mathcal{U}$). We define the set $\mathsf{sub}_\uparrow(\mathcal{O}) := \{C_\uparrow \mid C \in \mathsf{sub}(\mathcal{O})\}$ and the order structure

$\mathcal{U} := \mathcal{V}_\mathcal{O} \cup \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}_\uparrow(\mathcal{O}) \cup \{\lambda, \neg\lambda\}$ with the involutive operation inv given by $\mathrm{inv}(\lambda) := \neg\lambda$, $\mathrm{inv}(C) := \neg C$, and $\mathrm{inv}(C_\uparrow) := (\neg C)_\uparrow$ for all $C \in \mathsf{sub}(\mathcal{O})$.

For convenience, we extend the notation of $\mathsf{sub}_\uparrow(\mathcal{O})$ to the elements of $\mathcal{V}_\mathcal{O}$ by setting $q_\uparrow := q$ for all $q \in \mathcal{V}_\mathcal{O}$.

Using total preorders from $\mathrm{order}(\mathcal{U})$, we can now describe the relationships between all the subconcepts from $\mathcal{O}$ and the truth degrees from $\mathcal{V}_\mathcal{O}$ at given domain elements. One can think of such a preorder as the *type* of a domain element, from which a tree-shaped interpretation can be built, represented by a *Hintikka tree*.

In the following, let n be the number of quantified concepts in $\mathsf{sub}(\mathcal{O})$ and ϕ an arbitrary but fixed bijection between the set of all quantified concepts in $\mathsf{sub}(\mathcal{O})$ and $\{1, \dots, n\}$. This bijection specifies which quantified concept is witnessed by which successor in the Hintikka tree. For a given role $r \in \mathsf{N_R}$, we denote by Φ_r the set of all indices $\phi(E)$ where $E \in \mathsf{sub}(\mathcal{O})$ is a quantified concept of the form $\exists r.C$ or $\forall r.C$. Our algorithm will try to decide the existence of an n-ary infinite tree whose nodes are labeled with preorders from $\mathrm{order}(\mathcal{U})$, such that the semantics of the constructors and all the axioms in $\mathcal{O}$ are preserved.

Definition 8 (Hintikka ordering). A *Hintikka ordering* is a total preorder $\lesssim_H \in \mathrm{order}(\mathcal{U})$ that satisfies the following conditions for every $C \in \mathsf{sub}(\mathcal{O})$:

- $C = \top$ implies $C \equiv_H 1$,
- if $C = D_1 \sqcap D_2$, then $C \equiv_H \min_H(D_1, D_2)$,
- if $C = D_1 \to D_2$, then $C \equiv_H \mathrm{res}_H(D_1, D_2)$.

This preorder is *compatible* with the TBox $\mathcal{T}$ if for every GCI $\langle C \sqsubseteq D \geq q\rangle \in \mathcal{T}$ we have $\mathrm{res}_H(C, D) \gtrsim_H q$. It is *compatible* with $\mathcal{A}$ if for every order assertion $\langle a\!:\!C \bowtie q\rangle$ or $\langle a\!:\!C \bowtie a\!:\!D\rangle$ in $\mathcal{A}$, we have $C \bowtie_H q$ or $C \bowtie_H D$, respectively.

The conditions imposed on Hintikka orderings ensure that they preserve the semantics of all the *propositional* constructors. For every *quantified* concept E, we still need to ensure the existence of a witness. This is achieved through ϕ and the following Hintikka condition.

Definition 9 (Hintikka condition). The *Hintikka condition* consists of the following requirements for an $(n{+}1)$-tuple $(\lesssim_0, \lesssim_1, \dots, \lesssim_n)$ of Hintikka orderings:

- for every $1 \leq i \leq n$ and all $\alpha, \beta \in \mathcal{V}_\mathcal{O} \cup \mathsf{sub}(\mathcal{O})$, we have $\alpha \lesssim_0 \beta$ iff $\alpha_\uparrow \lesssim_i \beta_\uparrow$, where $q_\uparrow := q$ for all $q \in \mathcal{V}_\mathcal{O}$;
- for every $\exists r.D \in \mathsf{sub}(\mathcal{O})$, we have
  - $(\exists r.D)_\uparrow \equiv_i \min_i(\lambda, D)$ for $i = \phi(\exists r.D)$, and
  - $(\exists r.D)_\uparrow \gtrsim_i \min_i(\lambda, D)$ for all $i \in \Phi_r$; and
- for every $\forall r.D \in \mathsf{sub}(\mathcal{O})$, we have
  - $(\forall r.D)_\uparrow \equiv_i \mathrm{res}_i(\lambda, D)$ for $i = \phi(\forall r.D)$, and
  - $(\forall r.D)_\uparrow \lesssim_i \mathrm{res}_i(\lambda, D)$ for all $i \in \Phi_r$.

A *Hintikka tree* for $\mathcal{O}$ is an infinite n-ary tree,[1] where every node u is associated with a Hintikka ordering $\lesssim_u$ compatible with $\mathcal{T}$, such that:

[1] We will use words from $\{1, \dots, n\}^*$ to denote the nodes in an infinite n-ary tree.

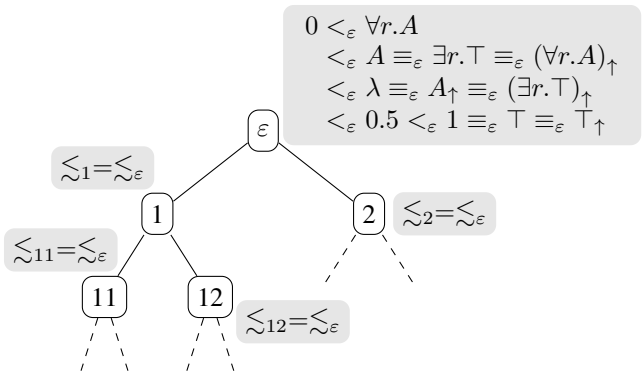


Figure 4: A Hintikka tree for Example 3

- every tuple $(\lesssim_u, \lesssim_{u1}, \dots, \lesssim_{un})$ satisfies the Hintikka condition, and
- $\lesssim_\varepsilon$ is compatible with $\mathcal{A}$.

For instance, Figure 4 shows a Hintikka tree for the TBox $\mathcal{T}_1$ from Example 3 and the ABox $\mathcal{A} = \{\langle a\!:\!A < 1\rangle\}$. Notice that every node is labeled with the same preorder and the tree is invariant w.r.t. the choice of ϕ. We now show that the existence of a Hintikka tree for an ontology $\mathcal{O}$ characterizes the consistency of $\mathcal{O}$.

Proposition 10. *If there is a Hintikka tree for $\mathcal{O}$, then $\mathcal{O}$ has a model.*

Proof. Given a Hintikka tree, we construct a model in two steps. In the first step, we recursively define a function $v\colon \mathcal{U} \times \{1, \dots, n\}^* \to [0, 1]$ satisfying the following conditions for all nodes u and all $\alpha, \beta \in \mathcal{U}$:

(P1) for all values $q \in \mathcal{V}_\mathcal{O}$ we have $v(q, u) = q$,
(P2) $v(\alpha, u) \leq v(\beta, u)$ iff $\alpha \lesssim_u \beta$,
(P3) $v(\mathrm{inv}(\alpha), u) = 1 - v(\alpha, u)$,
(P4) for all $C \in \mathsf{sub}(\mathcal{O})$ and all $i \in \{1, \dots, n\}$

$$v(C, u) = v(C_\uparrow, ui).$$

In the second step, we construct, with the help of this function v, an interpretation $\mathcal{I}_v = (\{1, \dots, n\}^*, \cdot^{\mathcal{I}_v})$ satisfying $C^{\mathcal{I}_v}(u) = v(C, u)$ for all concepts C and all nodes u, and show that $\mathcal{I}_v$ is indeed a model of $\mathcal{O}$.

Step 1 The function v is defined recursively, starting from the root node ε. Let $\mathcal{U}/{\equiv_\varepsilon}$ be the set of all equivalence classes of $\equiv_\varepsilon$. Then $\lesssim_\varepsilon$ yields a total order on $\mathcal{U}/{\equiv_\varepsilon}$. In particular, $[0]_\varepsilon <_\varepsilon [q_1]_\varepsilon <_\varepsilon [q_2]_\varepsilon <_\varepsilon \cdots <_\varepsilon [q_{k-1}]_\varepsilon <_\varepsilon [1]_\varepsilon$ holds if we extend $<_\varepsilon$ to $\mathcal{U}/{\equiv_\varepsilon}$ in the obvious way. For an equivalence class $[\alpha]_\varepsilon$, we set $\mathrm{inv}([\alpha]_\varepsilon) := [\mathrm{inv}(\alpha)]_\varepsilon$, which is well-defined since $\lesssim_\varepsilon$ is an element of $\mathrm{order}(\mathcal{U})$.

We first define an auxiliary function $\tilde{v}_\varepsilon\colon \mathcal{U}/{\equiv_\varepsilon} \to [0, 1]$. For all $q \in \mathcal{V}_\mathcal{O}$ we define $\tilde{v}_\varepsilon([q]_\varepsilon) := q$. It remains to define a value for all equivalence classes that do not contain a value from $\mathcal{V}_\mathcal{O}$. Notice that due to the minimality of $[0]_\varepsilon$ and maximality of $[1]_\varepsilon$ every such class must be strictly between $[q_i]_\varepsilon$ and $[q_{i+1}]_\varepsilon$ for two adjacent truth degrees q_i, q_{i+1}. For

every $i \in \{0, \dots, k-1\}$, let ν_i be the number of equivalence classes that are strictly between $[q_i]_\varepsilon$ and $[q_{i+1}]_\varepsilon$. We assume that these classes are denoted by E^i_j such that $[q_i]_\varepsilon <_\varepsilon E^i_1 <_\varepsilon E^i_2 <_\varepsilon \cdots <_\varepsilon E^i_{\nu_i} <_\varepsilon [q_{i+1}]_\varepsilon$. We then define values $q_i < s^i_1 < s^i_2 < \cdots < s^i_{\nu_i} < q_{i+1}$ as

$$s^i_j := q_i + \tfrac{j}{\nu_i+1}(q_{i+1} - q_i) \tag{3}$$

and set $\tilde{v}_\varepsilon(E^i_j) := s^i_j$ for every j, $1 \le j \le \nu_i$. Finally, we define $v(\alpha, \varepsilon) := \tilde{v}_\varepsilon([\alpha]_\varepsilon)$ for all $\alpha \in \mathcal{U}$. This construction ensures that (P1) and (P2) hold at the node ε. To see that (P3) is also satisfied, note that $1-q_{i+1}$ and $1-q_i$ are also adjacent in $\mathcal{V}_\mathcal{O}$ and have exactly the inverses $\mathrm{inv}(E^i_j)$ between them in reversed order.

For the recursion step, assume that we have already defined v for a node u, such that (P1)–(P3) are satisfied at u and let $i \in \{1, \dots, n\}$. We initialize the auxiliary function $\tilde{v}_{ui} \colon \mathcal{U}/{\equiv_{ui}} \to [0,1]$ by setting $\tilde{v}_{ui}([q]_{ui}) := q$ for all $q \in \mathcal{V}_\mathcal{O}$ and $\tilde{v}_{ui}([C_\uparrow]_{ui}) := v(C, u)$ for all $C \in \mathsf{sub}(\mathcal{O})$. To see that this is well-defined, consider $[C_\uparrow]_{ui} = [D_\uparrow]_{ui}$, i.e. $C_\uparrow \equiv_{ui} D_\uparrow$. From the Hintikka condition, it follows that $C \equiv_u D$, and from (P2) at u we obtain $v(C,u) = v(D,u)$. A similar argument can be used to show that $[q]_{ui} = [C_\uparrow]_{ui}$ implies $v(q,u) = v(C,u)$. For the remaining equivalence classes, we can use a construction analogous to the case for ε by considering the two unique neighboring equivalence classes that contain an element of $\mathcal{V}_\mathcal{O} \cup \mathsf{sub}_\uparrow(\mathcal{O})$. We now define $v(\alpha, ui) := \tilde{v}_{ui}([\alpha]_{ui})$. This construction ensures that (P1)–(P3) hold at ui, and that (P4) holds for u.

Step 2 We define the interpretation $\mathcal{I}_v$ over the domain $\{1, \dots, n\}^*$ as follows. For every concept name $A \in \mathsf{N_C}$ and all domain elements u, we set

$$A^{\mathcal{I}_v}(u) := \begin{cases} v(A,u) & \text{if } A \in \mathsf{sub}(\mathcal{O}), \\ 0 & \text{otherwise.} \end{cases}$$

For every role name $r \in \mathsf{N_R}$ and all domain elements u, we likewise define

$$r^{\mathcal{I}_v}(u,w) := \begin{cases} v(\lambda, ui) & \text{if } w = ui \text{ with } i \in \Phi_r, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we define $a^{\mathcal{I}_v} := \varepsilon$ for the individual name a. It can be shown by induction on the structure of C that

$$C^{\mathcal{I}_v}(u) = v(C,u) \text{ for all } C \in \mathsf{sub}(\mathcal{O}), u \in \{1, \dots, n\}^* \tag{4}$$

holds. In this proof by induction

- the base case follows trivially from the definition of $\mathcal{I}_v$,
- the cases $\top$, $C \sqcap D$, and $C \to D$ follow from (P1), (P2), and Definition 8,
- the case $\neg C$ follows from (P3), and
- Definition 9 and (P4) entail the cases $\exists r.C$ and $\forall r.C$.

For details, we refer the reader to the technical report (Borgwardt, Distel, and Peñaloza 2013).

It remains to show that $\mathcal{I}_v$ is indeed a model of $\mathcal{O}$. For every $\langle a\colon C \bowtie q\rangle \in \mathcal{A}$, the Hintikka tree satisfies $C \bowtie_\varepsilon q$, and thus we obtain from (4), (P1), and (P2):

$$C^{\mathcal{I}_v}(a^{\mathcal{I}_v}) = v(C, \varepsilon) \bowtie v(q, \varepsilon) = q,$$

and similarly for assertions of the form $\langle a\colon C \bowtie a\colon D\rangle$.

Now, let $u \in \{1, \dots, n\}^*$ be a domain element of $\mathcal{I}_v$ and $\langle C \sqsubseteq D \ge q\rangle \in \mathcal{T}$. Since $p \in \mathcal{V}_\mathcal{O}$ and $\lesssim_u$ is compatible with $\mathcal{T}$, it must hold that

$$q \lesssim_u \mathrm{res}_u(C,D) = \begin{cases} 1 & \text{if } C \lesssim_u D \\ D & \text{if } D <_u C \end{cases} \overset{\text{(P2)}}{=} \begin{cases} 1 & \text{if } v(C,u) \le v(D,u) \\ D & \text{if } v(D,u) < v(C,u). \end{cases}$$

Thus, (P1) and (P2) yield

$$\begin{aligned} q = v(q,u) &\le \begin{cases} v(1,u) & \text{if } v(C,u) \le v(D,u) \\ v(D,u) & \text{if } v(D,u) < v(C,u) \end{cases} \\ &= v(C,u) \Rightarrow v(D,u) \\ &= C^{\mathcal{I}_v}(u) \Rightarrow D^{\mathcal{I}_v}(u). \end{aligned}$$

□

Conversely, every model can be transformed into a Hintikka tree. The idea is to *unravel* the model into an infinite tree, and then abstract from the specific values by just considering the ordering between the elements of $\mathcal{U}$. This idea is formalized next.

Proposition 11. *If $\mathcal{O}$ has a model, then there is a Hintikka tree for $\mathcal{O}$.*

Proof. Let $\mathcal{I}$ be a model of $\mathcal{O}$. We use this model to guide the construction of a Hintikka tree for $\mathcal{O}$. During this construction, we will recursively generate a mapping $g\colon \{1, \dots, n\}^* \to \Delta^\mathcal{I}$ specifying which domain elements correspond to the nodes in the tree. This mapping will satisfy the following condition for all $\alpha, \beta \in \mathcal{V}_\mathcal{O} \cup \mathsf{sub}(\mathcal{O})$ and all $u \in \{1, \dots, n\}^*$:

(P5) $\alpha \lesssim_u \beta$ iff $\alpha^\mathcal{I}(g(u)) \le \beta^\mathcal{I}(g(u))$,

where we define $q^\mathcal{I}(x) := q$ for all $q \in \mathcal{V}_\mathcal{O}$ and $x \in \Delta^\mathcal{I}$.

We first consider the root node ε of the tree. Recall that the ontology contains a local ordered ABox, using only the individual name a. We define $g(\varepsilon) := a^\mathcal{I}$ and the Hintikka ordering $\lesssim_\varepsilon$ as follows for all $\alpha, \beta \in \mathcal{V}_\mathcal{O} \cup \mathsf{sub}(\mathcal{O})$:

$$\alpha \lesssim_\varepsilon \beta \text{ iff } \alpha^\mathcal{I}(a^\mathcal{I}) \le \beta^\mathcal{I}(a^\mathcal{I}).$$

We extend this order to the elements in $\mathsf{sub}_\uparrow(\mathcal{O}) \cup \{\lambda, \neg\lambda\}$ arbitrarily, in such a way that for all $\alpha, \beta \in \mathcal{U}$ we have $\alpha \lesssim_\varepsilon \beta$ iff $\mathrm{inv}(\beta) \lesssim_\varepsilon \mathrm{inv}(\alpha)$. It is straightforward to show that $\lesssim_\varepsilon$ is an element of $\mathrm{order}(\mathcal{U})$ satisfying (P5) at ε, and that furthermore $\lesssim_\varepsilon$ is a Hintikka ordering that is compatible with $\mathcal{T}$ (cf. (Borgwardt, Distel, and Peñaloza 2013)).

Assume now that we have already defined $g(u)$ and $\lesssim_u$ for a node $u \in \{1, \dots, n\}^*$ such that (P5) is satisfied. For all $i \in \{1, \dots, n\}$, we now construct $\lesssim_{ui}$ in such a way that the tuple $(\lesssim_u, \lesssim_{u1}, \dots, \lesssim_{un})$ satisfies the Hintikka condition. For brevity, we consider only the case that $i = \phi(\exists r.D)$;

value restrictions can be handled using similar arguments. Since $\mathcal{I}$ is witnessed, there must be a $y_i \in \Delta^{\mathcal{I}}$ such that $(\exists r.D)^{\mathcal{I}}(g(u)) = \min(r^{\mathcal{I}}(g(u), y_i), D^{\mathcal{I}}(y_i))$. We define $g(ui) := y_i$, and $\lesssim_{ui}$ for all $\alpha, \beta \in \mathcal{U}$ by

$$\alpha \lesssim_{ui} \beta \text{ iff } \alpha^{\mathcal{I}}(g(ui)) \leq \beta^{\mathcal{I}}(g(ui)), \tag{5}$$

where we abbreviate $\lambda^{\mathcal{I}}(g(ui)) := r^{\mathcal{I}}(g(u), g(ui))$ and $(C_\uparrow)^{\mathcal{I}}(g(ui)) := C^{\mathcal{I}}(g(u))$ for all concepts $C \in \mathsf{sub}(\mathcal{O})$. It is clear that $\lesssim_{ui}$ behaves on $\mathcal{V}_\mathcal{O} \cup \mathsf{sub}_\uparrow(\mathcal{O})$ exactly as $\lesssim_u$ does on $\mathcal{V}_\mathcal{O} \cup \mathsf{sub}(\mathcal{O})$. Following the same arguments used for the root node, it is easy to show that $\lesssim_{ui}$ is actually a Hintikka ordering compatible with $\mathcal{T}$.

We show the Hintikka condition for $(\lesssim_u, \lesssim_{u1}, \ldots, \lesssim_{un})$. For the case $i = \phi(\exists r.D)$, the construction of g yields that $(\exists r.D)^{\mathcal{I}}(g(u)) = \min\left(r^{\mathcal{I}}(g(u), g(ui)), D^{\mathcal{I}}(g(ui))\right)$, and thus

$$((\exists r.D)_\uparrow)^{\mathcal{I}}(g(ui)) = \min\left(\lambda^{\mathcal{I}}(g(ui)), D^{\mathcal{I}}(g(ui))\right).$$

Using (5), we obtain $(\exists r.D)_\uparrow \equiv_{ui} \min_{ui}(\lambda, D)$ as required. Furthermore, for all $i \in \Phi_r$, it holds that

$$\begin{aligned}(\exists r.D)^{\mathcal{I}}(g(u)) &= \sup_{y \in \Delta^{\mathcal{I}}} \min\left(r^{\mathcal{I}}(g(u), y), D^{\mathcal{I}}(y)\right) \\ &\geq \min\left(r^{\mathcal{I}}(g(u), g(ui)), D^{\mathcal{I}}(g(ui))\right),\end{aligned}$$

which similarly shows $(\exists r.D)_\uparrow \gtrsim_{ui} \min_{ui}(\lambda, D)$. Similar arguments apply to the value restrictions in $\mathsf{sub}(\mathcal{O})$.

Finally, for every $\langle a\!:\!C \bowtie q\rangle \in \mathcal{A}$, we have $C^{\mathcal{I}}(a^{\mathcal{I}}) \bowtie q$, and thus $C \bowtie_\varepsilon q$ by definition of $\lesssim_\varepsilon$, and similarly for assertions of the form $\langle a\!:\!C \bowtie a\!:\!D\rangle$. Hence, the tree defined by $\lesssim_u$, for $u \in \{1, \ldots, n\}^*$, is a Hintikka tree for $\mathcal{O}$. □

Propositions 10 and 11 show that Hintikka trees characterize consistency of an ontology with a local ordered ABox. In other words, deciding the existence of a Hintikka tree for $\mathcal{O}$ suffices for deciding consistency of $\mathcal{O}$. We now turn our attention to the former problem, and show that it can be solved in exponential time in the size of $\mathcal{O}$. For this, we construct a *looping tree automaton* whose runs correspond exactly to such Hintikka trees. Thus, the automaton accepts a non-empty language iff the ontology $\mathcal{O}$ is consistent.

A *looping automaton* over n-ary (infinite) trees is a tuple $\mathbf{A} = (Q, I, \Delta)$, consisting of a non-empty set Q of *states*, a subset $I \subseteq Q$ of *initial states*, and a *transition relation* $\Delta \subseteq Q^{n+1}$. A *run* of this automaton is a mapping $\rho\colon \{1, \ldots, n\}^* \to Q$ such that (i) $\rho(\varepsilon) \in I$, and (ii) for all $u \in \{1, \ldots, n\}^*$, we have $\left(\rho(u), \rho(u1), \ldots, \rho(un)\right) \in \Delta$. $\mathbf{A}$ is *non-empty* iff it has a run.

Definition 12. The *Hintikka automaton* for an ontology $\mathcal{O}$ is the looping tree automaton $\mathbf{A}_\mathcal{O} := (Q_\mathcal{O}, I_\mathcal{O}, \Delta_\mathcal{O})$, where

- $Q_\mathcal{O}$ is the set of all Hintikka orderings compatible with $\mathcal{T}$,
- $I_\mathcal{O} := \{\lesssim_H \in Q_\mathcal{O} \mid \lesssim_H \text{ is compatible with } \mathcal{A}\}$, and
- $\Delta_\mathcal{O}$ contains all tuples from $Q_\mathcal{O}^{n+1}$ that satisfy the Hintikka condition.

It is easy to see that the runs of $\mathbf{A}_\mathcal{O}$ are exactly the Hintikka trees for $\mathcal{O}$. Observe that the number of Hintikka orderings for $\mathcal{O}$ is bounded by $2^{|\mathcal{U}|^2}$ and the cardinality of $\mathcal{U} = \mathcal{V}_\mathcal{O} \cup \mathsf{sub}(\mathcal{O}) \cup \mathsf{sub}_\uparrow(\mathcal{O}) \cup \{\lambda, \neg\lambda\}$ is linear in the size of $\mathcal{O}$. Likewise, the arity n of the automaton is bounded by $|\mathsf{sub}(\mathcal{O})|$, which is linear in the size of $\mathcal{O}$. Thus, the size of the Hintikka automaton $\mathbf{A}_\mathcal{O}$ is exponential in the size of $\mathcal{O}$. Since (non-)emptiness of looping tree automata can be decided in polynomial time (Vardi and Wolper 1986), we obtain overall an ExpTime-decision procedure for consistency of ontologies with local ordered ABoxes in G-$\mathfrak{I}\mathcal{ALC}$. Note that concept satisfiability in classical $\mathcal{ALC}$ is already ExpTime-hard w.r.t. general TBoxes (Schild 1991), and hence our complexity bounds are tight.

Theorem 13. *Consistency in* G-$\mathfrak{I}\mathcal{ALC}$ *w.r.t. local ordered ABoxes and witnessed models is* ExpTime*-complete.*

In the following section, we remove the restriction to local ordered ABoxes and show that consistency remains ExpTime-complete in the general case.

5 Reducing Consistency to Local Consistency

To decide consistency of G-$\mathfrak{I}\mathcal{ALC}$-ontologies containing more that one individual name, we adapt a technique from classical DLs known as *pre-completion* (Hollunder 1996). Intuitively, we try to build a forest-shaped model that satisfies the ontology. This model is composed of a finite set of trees, one for each individual name appearing in the ABox, whose roots can be arbitrarily interconnected due to the presence of role assertions. As before, rather than explicitly building such models, we use total preorders to represent them in an abstract manner.

The idea of pre-completion is to extend the input ABox to a full specification of each individual, and then decide consistency w.r.t. the local ABoxes associated with each individual name. In our setting, this amounts to extending the input ABox to a total preorder $\lesssim_\mathcal{A}$. This preorder represents the nucleus of a model of the ontology. To extend this to a full model, we check an (ordered) local consistency condition for each of the individual names, and use $\lesssim_\mathcal{A}$ to combine the resulting tree-shaped interpretations.

More formally, let $\mathcal{O} = (\mathcal{A}, \mathcal{T})$ be an ontology, and let $\mathsf{Ind}(\mathcal{A})$ denote the set of individual names occurring in $\mathcal{A}$. We define the order structure

$$\begin{aligned}\mathcal{W} := \mathcal{V}_\mathcal{O} &\cup \{a\!:\!C \mid a \in \mathsf{Ind}(\mathcal{A}),\ C \in \mathsf{sub}(\mathcal{O})\} \\ &\cup \{(a,b)\!:\!r \mid a, b \in \mathsf{Ind}(\mathcal{A}),\ r \text{ occurs in } \mathcal{O}\} \\ &\cup \{(a,b)\!:\!\neg r \mid a, b \in \mathsf{Ind}(\mathcal{A}),\ r \text{ occurs in } \mathcal{O}\}\end{aligned}$$

with $\mathrm{inv}(a\!:\!C) := a\!:\!\neg C$ and $\mathrm{inv}((a,b)\!:\!r) := (a,b)\!:\!\neg r$.

Definition 14 (pre-completion). A *pre-completion* of $\mathcal{A}$ w.r.t. $\mathcal{T}$ is a total preorder $\lesssim_\mathcal{A} \in \mathrm{order}(\mathcal{W})$ such that:

a) for every $a \in \mathsf{Ind}(\mathcal{A})$ and all $C \in \mathsf{sub}(\mathcal{O})$,
   - if $C = \top$, then $a\!:\!C \equiv_\mathcal{A} 1$,
   - if $C = D_1 \sqcap D_2$, then $a\!:\!C \equiv_\mathcal{A} \min_\mathcal{A}(a\!:\!D_1, a\!:\!D_2)$,
   - if $C = D_1 \to D_2$, then $a\!:\!C \equiv_\mathcal{A} \mathrm{res}_\mathcal{A}(a\!:\!D_1, a\!:\!D_2)$;

b) for every $\exists r.C \in \mathsf{sub}(\mathcal{O})$ and $a, b \in \mathsf{Ind}(\mathcal{A})$, we have
$$a\!:\!\exists r.C \gtrsim_\mathcal{A} \min_\mathcal{A}((a,b)\!:\!r, b\!:\!C);$$

c) for every $\forall r.C \in \mathsf{sub}(\mathcal{O})$ and $a, b \in \mathsf{Ind}(\mathcal{A})$, we have
$$a\!:\!\forall r.C \lesssim_\mathcal{A} \mathrm{res}_\mathcal{A}((a,b)\!:\!r, b\!:\!C);$$

d) for all $a \in \mathsf{Ind}(\mathcal{A})$ and every GCI $\langle C \sqsubseteq D \geq q\rangle \in \mathcal{T}$, we have $\mathrm{res}_{\mathcal{A}}(a\!:\!C, a\!:\!D) \gtrsim_{\mathcal{A}} q$; and

e) for every assertion $\langle \alpha \bowtie \beta\rangle \in \mathcal{A}$, we have $\alpha \bowtie_{\mathcal{A}} \beta$.

This definition generalizes the local conditions of Definitions 8 and 9 to handle several named individuals simultaneously. One difference is that we do not create witnesses for the quantified concepts here. This will be taken care of by testing the following local ordered ABoxes for consistency. For a pre-completion $\lesssim_{\mathcal{A}}$ and $a \in \mathsf{Ind}(\mathcal{A})$, we define the local ordered ABox $\mathcal{A}_a$ as the set of all order assertions $\langle \alpha \bowtie \beta\rangle$ over a and $\mathsf{sub}(\mathcal{O})$ satisfying $\alpha \bowtie_{\mathcal{A}} \beta$.[2] That is,

$$\mathcal{A}_a := \{\langle a\!:\!C \bowtie q\rangle \mid C \in \mathsf{sub}(\mathcal{O}),\ q \in \mathcal{V}_{\mathcal{O}},\ a\!:\!C \bowtie_{\mathcal{A}} q\} \cup \{\langle a\!:\!C \bowtie a\!:\!D\rangle \mid C, D \in \mathsf{sub}(\mathcal{O}),\ a\!:\!C \bowtie_{\mathcal{A}} a\!:\!D\}.$$

Lemma 15. *An ontology $\mathcal{O} = (\mathcal{A}, \mathcal{T})$ is consistent iff there is a pre-completion $\lesssim_{\mathcal{A}}$ of $\mathcal{A}$ w.r.t. $\mathcal{T}$ such that, for every $a \in \mathsf{Ind}(\mathcal{A})$, the ontology $\mathcal{O}_a := (\mathcal{A}_a, \mathcal{T})$ is consistent.*

Proof. Let $\mathcal{I}$ be a model of $\mathcal{O}$. We define the total preorder $\lesssim_{\mathcal{A}}$ by setting $\alpha \lesssim_{\mathcal{A}} \beta$ if and only if $\alpha^{\mathcal{I}} \leq \beta^{\mathcal{I}}$, where we set $((a,b)\!:\!\neg r)^{\mathcal{I}} := 1 - r^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}})$. A straightforward argument that $\lesssim_{\mathcal{A}}$ is a pre-completion of $\mathcal{A}$ w.r.t. $\mathcal{T}$ and that $\mathcal{I}$ is a model of $(\mathcal{A}_a, \mathcal{T})$ for each $a \in \mathsf{Ind}(\mathcal{A})$ can be found in (Borgwardt, Distel, and Peñaloza 2013).

Conversely, let $\lesssim_{\mathcal{A}}$ be a pre-completion of $\mathcal{A}$ w.r.t. $\mathcal{T}$ and each $(\mathcal{A}_a, \mathcal{T})$ be consistent. By Proposition 11, there are Hintikka trees for $(\mathcal{A}_a, \mathcal{T})$ that consist of Hintikka orderings $\lesssim^a_u$ for all $u \in \{1, \dots, n\}^*$, where n is the number of existential and value restrictions in $\mathsf{sub}(\mathcal{O})$. Similar to the proof of Proposition 10, we first construct a function $v\colon \mathcal{W} \cup (\mathsf{Ind}(\mathcal{A}) \times \mathcal{U} \times \{1, \dots, n\}^*) \to [0,1]$ such that

- for all values $q \in \mathcal{V}_{\mathcal{O}}$, we have $v(q) = q$,
- for all $\alpha, \beta \in \mathcal{W}$, we have $v(\alpha) \leq v(\beta)$ iff $\alpha \lesssim_{\mathcal{A}} \beta$,
- for all $\alpha \in \mathcal{W}$, we have $v(\mathrm{inv}(\alpha)) = 1 - v(\alpha)$,
- for every $C \in \mathsf{sub}(\mathcal{O})$ and all $a \in \mathsf{Ind}(\mathcal{A})$, we have $v(a\!:\!C) = v(a, C, \varepsilon)$,
- for all $u \in \{1, \dots, n\}^*$ and all $a \in \mathsf{Ind}(\mathcal{A})$,
  - for all values $q \in \mathcal{V}_{\mathcal{O}}$, we have $v(a, q, u) = q$,
  - for all $\alpha, \beta \in \mathcal{U}$, we have $v(a, \alpha, u) \leq v(a, \beta, u)$ iff $\alpha \lesssim^a_u \beta$,
  - for all $\alpha \in \mathcal{U}$, we have $v(a, \mathrm{inv}(\alpha), u) = 1 - v(a, \alpha, u)$, and
  - for all concepts $C \in \mathsf{sub}(\mathcal{O})$ and all $i \in \{1, \dots, n\}$, we have that $v(a, C, u) = v(a, C_\uparrow, ui)$.

We will then use this function to define a model of $\mathcal{O}$.

Using the technique from the proof of Proposition 10, we first define v on $\mathcal{W}$. On the set $\mathcal{W}/\!\equiv_{\mathcal{A}}$ of all equivalence classes of $\equiv_{\mathcal{A}}$, we define an auxiliary function $\tilde{v}_{\mathcal{A}}\colon \mathcal{W}/\!\equiv_{\mathcal{A}} \to [0,1]$, by setting $\tilde{v}_{\mathcal{A}}([q]_{\mathcal{A}}) := q$ for each $q \in \mathcal{V}_{\mathcal{O}}$ and treating the remaining equivalence classes as in (3). We then define $v(\alpha) := \tilde{v}_{\mathcal{A}}([\alpha]_{\mathcal{A}})$ for all $\alpha \in \mathcal{W}$.

For each $a \in \mathsf{Ind}(\mathcal{A})$, $C \in \mathsf{sub}(\mathcal{O})$, and $q \in \mathcal{V}_{\mathcal{O}}$, we now set $v(a, C, \varepsilon) := v(a\!:\!C)$ and $v(a, q, \varepsilon) := q$. The values of $v(a, \alpha, \varepsilon)$ for elements $\alpha \in \mathsf{sub}_\uparrow(\mathcal{O}) \cup \{\lambda, \neg\lambda\}$ are irrelevant for the desired properties and can be fixed arbitrarily, as long as we have $v(a, \alpha, \varepsilon) \leq v(a, \beta, \varepsilon)$ iff $\alpha \lesssim^a_\varepsilon \beta$ and $v(a, \mathrm{inv}(\alpha), \varepsilon) = 1 - v(a, \alpha, u)$ for all $\alpha, \beta \in \mathcal{U}$, e.g. using the technique in (3). The definition of $v(a, \alpha, u)$ can now proceed as in the proof of Proposition 10 based on the Hintikka trees for $(\mathcal{A}_a, \mathcal{T})$. This construction ensures that v has the desired properties.

We now define the interpretation $\mathcal{I}$ as follows:

- $\Delta^{\mathcal{I}} := \mathsf{Ind}(\mathcal{A}) \times \{1, \dots, n\}^*$,
- $a^{\mathcal{I}} := (a, \varepsilon)$ for each $a \in \mathsf{Ind}(\mathcal{A})$,
- $A^{\mathcal{I}}(a, u) := v(a, A, u)$ for all $a \in \mathsf{Ind}(\mathcal{A})$, concept names $A \in \mathsf{sub}(\mathcal{O})$, and $u \in \{1, \dots, n\}^*$, and
- $r^{\mathcal{I}}((a,u),(b,u')) :=$
$$\begin{cases} v(a, \lambda, ui) & \text{if } a = b \text{ and } u' = ui \text{ with } i \in \Phi_r, \\ v((a,b)\!:\!r) & \text{if } u = u' = \varepsilon \text{ and } r \text{ occurs in } \mathcal{O}, \\ 0 & \text{otherwise.} \end{cases}$$

The interpretation of the remaining individual and concept names is irrelevant and can be fixed arbitrarily. As in Proposition 10, we can show by induction on the structure of C that $C^{\mathcal{I}}(a, u) = v(a, C, u)$ holds for all $C \in \mathsf{sub}(\mathcal{O})$, $a \in \mathsf{Ind}(\mathcal{A})$, and $u \in \{1, \dots, n\}^*$ by induction on the structure of C. The claim for $\top$, $\neg C$, $C \sqcap D$, and $C \to D$ follows as before from Condition a) of Definition 14 and the fact that each $\lesssim^a_u$ is a Hintikka ordering.

Consider now an existential restriction $\exists r.C \in \mathsf{sub}(\mathcal{O})$ and the domain element (a, ε) for some $a \in \mathsf{Ind}(\mathcal{A})$. By the Hintikka condition and the induction hypothesis, we have $v(a, \exists r.C, u) = \min\big(r^{\mathcal{I}}((a,\varepsilon),(a,i_0)), C^{\mathcal{I}}(a, i_0)\big)$, where $i_0 = \phi(\exists r.C)$, as in the proof of Proposition 10. Likewise, $v(a, \exists r.C, u) \geq \min(r^{\mathcal{I}}((a,\varepsilon),(a,i)), C^{\mathcal{I}}(a,i))$ holds for all $i \in \Phi_r$. Finally, for each $b \in \mathsf{Ind}(\mathcal{A})$, we have $v(a, \exists r.C, u) \geq \min(r^{\mathcal{I}}((a,\varepsilon),(b,\varepsilon)), C^{\mathcal{I}}(b,\varepsilon))$ by Condition b) of Definition 14. Since (a, ε) does not have any other relevant r-successors, this shows the claim for $\exists r.C$ at (a, ε). At the other domain elements, it can be shown as for Proposition 10. Similar arguments apply to all $\forall r.C \in \mathsf{sub}(\mathcal{O})$.

Finally, the fact that $\mathcal{I}$ is actually a model of $\mathcal{O}$ is ensured by compatibility of all Hintikka orderings with $\mathcal{T}$ and Conditions e) and d) of Definition 14. □

Note that the cardinality of $\mathrm{order}(\mathcal{W})$ is exponential in the size of $\mathcal{O}$, and all elements of $\mathrm{order}(\mathcal{W})$ are of polynomial size. We can thus enumerate $\mathrm{order}(\mathcal{W})$, check for each element whether it satisfies Definition 14 in polynomial time, and then execute the polynomially many local consistency tests as described by Lemma 15. This yields the following complexity result.

Corollary 16. *Consistency in* G-$\mathcal{IALC}$ *w.r.t. witnessed models is* ExpTime*-complete.*

6 Satisfiability and Subsumption

We have described an exponential-time algorithm for deciding consistency of G-$\mathcal{IALC}$ ontologies. We now direct our attention at other standard reasoning problems in fuzzy

[2] It actually suffices to consider only $\bowtie \in \{>, =, <\}$.

DLs; namely, deciding concept satisfiability and subsumption, and computing the best truth degrees to which these hold. Recall from Section 2 that for these reasoning problems we can restrict our attention to ontologies with an empty ABox.

Let now $\mathcal{O} = (\emptyset, \mathcal{T})$ be an ontology. It is easy to see that p-subsumption and p-satisfiability w.r.t. $\mathcal{O}$ can be reduced in polynomial time to consistency w.r.t. local ordered ABoxes. More precisely, for any two concepts C, D and $p \in [0, 1]$,

- C is p-satisfiable w.r.t. $\mathcal{O}$ iff $(\{\langle a : C \geq p\rangle\}, \mathcal{T})$ is consistent, and
- C is p-subsumed by D w.r.t. $\mathcal{O}$ iff $(\{\langle a : C \rightarrow D < p\rangle\}, \mathcal{T})$ is inconsistent,

where a is an arbitrary individual name. We thus obtain the following result from Theorem 13.

Theorem 17. *Satisfiability and subsumption in* G-$\mathcal{IALC}$ *w.r.t. witnessed models are* EXPTIME*-complete.*

We now shift our attention to the problems of computing the *best* satisfiability and subsumption degrees. We first show that the local consistency checks required for deciding p-satisfiability and p-subsumption only depend on the position of p relative to the values occurring in $\mathcal{T}$, but not on the precise value of p. To prove this, we again use the preorders of the previous sections, and in particular Hintikka trees.

Lemma 18. *Let $p, p' \in (q_i, q_{i+1})$ for two adjacent values $q_i, q_{i+1} \in \mathcal{V}_\mathcal{O}$, and C be a concept. Then $(\{\langle a : C \bowtie p\rangle\}, \mathcal{T})$ is consistent iff $(\{\langle a : C \bowtie p'\rangle\}, \mathcal{T})$ is consistent.*

Proof. By Propositions 10 and 11, both consistency conditions are equivalent to the existence of Hintikka trees, albeit over different order structures. We denote by $\mathcal{U}_p$ the order structure from Definition 7 over $\mathcal{V}_p := \mathcal{V}_\mathcal{O} \cup \{p, 1 - p\}$, and by $\mathcal{U}_{p'}$ the one over $\mathcal{V}_{p'} := \mathcal{V}_\mathcal{O} \cup \{p', 1 - p'\}$. Observe that the bijection $\iota \colon \mathcal{V}_p \to \mathcal{V}_{p'}$ that simply maps p to p' and $1 - p$ to $1 - p'$ and leaves the other values as they are, can be extended to a bijection between $\mathcal{U}_p$ and $\mathcal{U}_{p'}$ by defining it as the identity on all elements outside of $\mathcal{V}_p$. Furthermore, it is compatible with the involutive operator inv, i.e. we have $\iota(\mathrm{inv}(\alpha)) = \mathrm{inv}(\iota(\alpha))$ for all $\alpha \in \mathcal{U}_p$.

It is straightforward to extend this bijection to Hintikka orderings and Hintikka tress (see (Borgwardt, Distel, and Peñaloza 2013) for details). Then there is a Hintikka tree for $(\{\langle a : C \bowtie p\rangle\}, \mathcal{T})$ iff there is one for $(\{\langle a : C \bowtie p'\rangle\}, \mathcal{T})$, which concludes the proof. □

This shows that subsumption between C and D or satisfiability of C either holds for all values in an interval (q_i, q_{i+1}), or for none of them.

Corollary 19. *For any two concepts C and D, the best subsumption degree of C and D w.r.t. $\mathcal{O}$ and the best satisfiability degree of C w.r.t. $\mathcal{O}$ are always in $\mathcal{V}_\mathcal{O}$.*

Since the best subsumption degree p of C and D is always a subsumption degree, i.e. C is p-subsumed by D, it suffices to check subsumption w.r.t. the values from $\mathcal{V}_\mathcal{O}$ in order to determine the best subsumption degree. Thus, we only have to execute linearly many (in-)consistency checks to compute the best subsumption degree.

However, it is possible that C is p-satisfiable for every $p \in (q_i, q_{i+1})$, but not q_{i+1}-satisfiable. Therefore, we check satisfiability for all values $\frac{q_i + q_{i+1}}{2}$. The best satisfiability degree is then the largest q_{i+1} for which this check succeeds (or 0 if it never succeeds). Again, this means that we have to execute linearly many consistency checks to compute the best satisfiability degree.

By combining these reductions with Theorem 13, we obtain the following results.

Corollary 20. *In* G-$\mathcal{IALC}$ *w.r.t. witnessed models, best subsumption and satisfiability degrees can be computed in exponential time.*

7 Conclusions

We have studied the standard reasoning problems for the fuzzy DL G-$\mathcal{IALC}$ w.r.t. witnessed model semantics. The contributions of the paper are twofold. First, we have shown that, contrary to popular belief, reasoning in this logic cannot be restricted to reasoning over finitely-valued models without affecting its consequences. In particular, this implies that the algorithms based on maintaining only a finite set of truth degrees (Bobillo et al. 2009; 2012) are incomplete for the general semantics. Moreover, this also implies that the logic does not have the finite model property, and hence standard tableau-based approaches cannot terminate (Bobillo and Straccia 2007; Straccia and Bobillo 2007; Bobillo, Bou, and Straccia 2011).

As the second contribution of the paper, we showed that all standard reasoning problems can be solved in exponential time. To achieve this, we developed an automaton that decides the existence of a Hintikka tree, which is an abstract representation of a model of a given ontology. The main insight needed for this approach is that we can abstract from the precise truth degrees assigned by an interpretation, and focus only on their ordering.

As an added benefit, in our formalism we can express order assertions like $\langle \mathsf{ana} : \mathsf{Tall} > \mathsf{bob} : \mathsf{Tall}\rangle$, intuitively stating that Ana is taller than Bob, without needing to specify the precise degrees to which ana and bob belong to the concept Tall. This is similar to concrete domains (Lutz 2003), which can even compare values at unnamed domain elements. But concrete domains allow only for atomic attributes, whereas order assertions can also contain complex concepts.

As we have developed an automata-based algorithm, it is natural to ask whether previous automata-based approaches (Baader, Hladik, and Peñaloza 2008; Borgwardt and Peñaloza 2013a) can be adapted to this setting in order to handle the expressivity up to G-$\mathcal{ISCHI}$, or provide better upper-bounds for reasoning w.r.t. *acyclic* TBoxes. We will study this problem in future work. We also plan to adapt these ideas into a tableau-based algorithm which is more suitable for implementation.

Recall that we have restricted our framework to reasoning w.r.t. *witnessed* models only. Indeed, this restriction is fundamental for our proof of Proposition 11. One open question is whether consistency of G-$\mathcal{IALC}$ ontologies w.r.t. *general* models is still decidable. We conjecture that it is, and in fact remains in EXPTIME.

Acknowledgements The authors would like to thank F. Baader for fruitful discussions on the topics of this paper.

References

Baader, F., and Peñaloza, R. 2011a. Are fuzzy description logics with general concept inclusion axioms decidable? In *Proc. FUZZ-IEEE'11*, 1735–1742. IEEE Press.

Baader, F., and Peñaloza, R. 2011b. On the undecidability of fuzzy description logics with GCIs and product t-norm. In *Proc. FroCoS'11*, volume 6989 of *LNCS*, 55–70. Springer.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2nd edition.

Baader, F.; Hladik, J.; and Peñaloza, R. 2008. Automata can show PSPACE results for description logics. *Inform. Comput.* 206(9-10):1045–1056.

Bobillo, F., and Straccia, U. 2007. A fuzzy description logic with product t-norm. In *Proc. FUZZ-IEEE'07*, 1–6. IEEE Press.

Bobillo, F., and Straccia, U. 2011. Reasoning with the finitely many-valued Łukasiewicz fuzzy description logic $\mathcal{SROIQ}$. *Inform. Sci.* 181:758–778.

Bobillo, F., and Straccia, U. 2013. Finite fuzzy description logics and crisp representations. In *Uncertainty Reasoning for the Semantic Web II*, volume 7123 of *LNCS*. Springer. 102–121.

Bobillo, F.; Delgado, M.; Gómez-Romero, J.; and Straccia, U. 2009. Fuzzy description logics under Gödel semantics. *Int. J. Approx. Reas.* 50(3):494–514.

Bobillo, F.; Delgado, M.; Gómez-Romero, J.; and Straccia, U. 2012. Joining Gödel and Zadeh fuzzy logics in fuzzy description logics. *Int. J. Uncertain. Fuzz.* 20(4):475–508.

Bobillo, F.; Bou, F.; and Straccia, U. 2011. On the failure of the finite model property in some fuzzy description logics. *Fuzzy Set. Syst.* 172(1):1–12.

Bobillo, F.; Delgado, M.; and Gómez-Romero, J. 2008. A crisp representation for fuzzy $\mathcal{SHOIN}$ with fuzzy nominals and general concept inclusions. In *Uncertainty Reasoning for the Semantic Web I*, volume 5327 of *LNAI*, 174–188. Springer.

Borgwardt, S., and Peñaloza, R. 2012. Undecidability of fuzzy description logics. In *Proc. KR'12*, 232–242. AAAI Press.

Borgwardt, S., and Peñaloza, R. 2013a. The complexity of lattice-based fuzzy description logics. *J. Data Semant.* 2(1):1–19.

Borgwardt, S., and Peñaloza, R. 2013b. Consistency reasoning in lattice-based fuzzy description logics. *Int. J. Approx. Reas.* In press.

Borgwardt, S.; Distel, F.; and Peñaloza, R. 2012. How fuzzy is my fuzzy description logic? In *Proc. IJCAR'12*, volume 7364 of *LNAI*, 82–96. Springer.

Borgwardt, S.; Distel, F.; and Peñaloza, R. 2013. Gödel description logics: Decidability in the absence of the finitely-valued model property. LTCS-Report 13-09, Chair of Automata Theory, TU Dresden, Germany. See http://lat.inf.tu-dresden.de/research/reports.html.

Cerami, M., and Straccia, U. 2013. On the (un)decidability of fuzzy description logics under Łukasiewicz t-norm. *Inform. Sci.* 227:1–21.

Cintula, P.; Hájek, P.; and Noguera, C., eds. 2011. *Handbook of Mathematical Fuzzy Logic*, volume 37–38 of *Studies in Logic*. College Publications.

Guller, D. 2012. On the satisfiability and validity problems in the propositional Gödel logic. *Comput. Intell.* 399:211–227.

Hájek, P. 2001. *Metamathematics of Fuzzy Logic (Trends in Logic)*. Springer.

Hájek, P. 2005. Making fuzzy description logic more general. *Fuzzy Set. Syst.* 154(1):1–15.

Hájek, P. 2007. On witnessed models in fuzzy logic. *Math. Logic Quart.* 53(1):66–77.

Hollunder, B. 1996. Consistency checking reduced to satisfiability of concepts in terminological systems. *Ann. Math. Artif. Intell.* 18(2-4):133–157.

Klement, E. P.; Mesiar, R.; and Pap, E. 2000. *Triangular Norms*. Trends in Logic, Studia Logica Library. Springer.

Lutz, C. 2003. Description logics with concrete domains - a survey. In *Proc. AiML'02*, 265–296. King's College Publications.

Schild, K. 1991. A correspondence theory for terminological logics: Preliminary report. In *Proc. IJCAI'91*, 466–471. Morgan Kaufmann.

Stoilos, G.; Stamou, G. B.; Pan, J. Z.; Tzouvaras, V.; and Horrocks, I. 2007. Reasoning with very expressive fuzzy description logics. *J. Artif. Intell. Res.* 30:273–320.

Straccia, U., and Bobillo, F. 2007. Mixed integer programming, general concept inclusions and fuzzy description logics. *Mathware & Soft Computing* 14(3):247–259.

Straccia, U. 2001. Reasoning within fuzzy description logics. *J. Artif. Intell. Res.* 14:137–166.

Straccia, U. 2006. Description logics over lattices. *Int. J. Uncertain. Fuzz.* 14(1):1–16.

Vardi, M. Y., and Wolper, P. 1986. Automata-theoretic techniques for modal logics of programs. *J. Comput. Syst. Sci.* 32(2):183–221.

Query Inseparability for Description Logic Knowledge Bases

E. Botoeva,[1] R. Kontchakov,[2] V. Ryzhikov,[1] F. Wolter[3] and M. Zakharyaschev[2]

[1]Faculty of Computer Science
Free University of Bozen-Bolzano, Italy
{botoeva,ryzhikov}@inf.unibz.it

[3]Department of Computer Science
University of Liverpool, U.K.
wolter@liverpool.ac.uk

[2]Department of Computer Science
Birkbeck, University of London, U.K.
{roman,michael}@dcs.bbk.ac.uk

Abstract

We investigate conjunctive query inseparability of description logic (DL) knowledge bases (KBs) with respect to a given signature, a fundamental problem for KB versioning, module extraction, forgetting and knowledge exchange. We study the data and combined complexity of deciding KB query inseparability for fragments of *Horn-$\mathcal{ALCHI}$*, including the DLs underpinning *OWL 2 QL* and *OWL 2 EL*. While all of these DLs are P-complete for data complexity, the combined complexity ranges from P to ExpTime and 2ExpTime. We also resolve two major open problems for *OWL 2 QL* by showing that TBox query inseparability and the membership problem for universal UCQ-solutions in knowledge exchange are both ExpTime-complete for combined complexity.

Introduction

A description logic (DL) knowledge base (KB) consists of a terminological box (TBox), storing conceptual knowledge, and an assertion box (ABox), storing data. Typical applications of KBs involve answering queries over incomplete data sources (ABoxes) augmented by ontologies (TBoxes) that provide additional information about the domain of interest as well as a convenient vocabulary for user queries. The standard query language in such applications, which balances expressiveness and computational complexity, is the language of conjunctive queries (CQs).

With typically large data, often tangled ontologies, and the hard problem of answering CQs over ontologies, various transformation and comparison tasks are becoming indispensable for KB engineering and maintenance. For example, to make answering certain CQs more efficient, one may want to extract from a given KB a smaller module returning the same answers to those CQs as the original KB; to provide the user with a more convenient query vocabulary, one may want to reformulate the KB in a new language. These tasks are known as module extraction (Stuckenschmidt, Parent, and Spaccapietra 2009) and knowledge exchange (Arenas et al. 2012); other relevant tasks include versioning, revision and forgetting (Jiménez-Ruiz et al. 2011; Wang, Wang, and Topor 2010; Lin and Reiter 1994).

In this paper, we investigate the following relationship between KBs that is fundamental for all such tasks. Let Σ be a signature consisting of concept and role names. We call KBs $\mathcal{K}_1$ and $\mathcal{K}_2$ Σ-*query inseparable* and write $\mathcal{K}_1 \equiv_\Sigma \mathcal{K}_2$ if any CQ formulated in Σ has the same answers over $\mathcal{K}_1$ and $\mathcal{K}_2$. Note that even for Σ containing all concept and role names, Σ-query inseparability does not necessarily imply logical equivalence. The relativisation to (smaller) signatures is crucial to support the tasks mentioned above:

(versioning) When comparing two *versions* $\mathcal{K}_1$ and $\mathcal{K}_2$ of a KB with respect to their answers to CQs in a relevant signature Σ, the basic task is to check whether $\mathcal{K}_1 \equiv_\Sigma \mathcal{K}_2$.

(modularisation) A Σ-*module of* a KB $\mathcal{K}$ is a KB $\mathcal{K}' \subseteq \mathcal{K}$ such that $\mathcal{K}' \equiv_\Sigma \mathcal{K}$. If we are only interested in answering CQs in Σ over $\mathcal{K}$, then we can achieve our aim by querying any Σ-module of $\mathcal{K}$ instead of $\mathcal{K}$ itself.

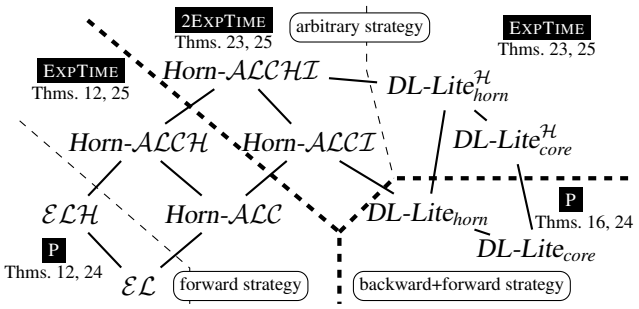
(knowledge exchange) In knowledge exchange, we want to transform a KB $\mathcal{K}_1$ in a signature Σ_1 to a new KB $\mathcal{K}_2$ in a disjoint signature Σ_2 connected to Σ_1 via a declarative mapping specification given by a TBox $\mathcal{T}_{12}$. Thus, the target KB $\mathcal{K}_2$ should satisfy the condition $\mathcal{K}_1 \cup \mathcal{T}_{12} \equiv_{\Sigma_2} \mathcal{K}_2$, in which case it is called a *universal UCQ-solution* (CQ and UCQ inseparabilities coincide for Horn DLs).

(forgetting) A KB $\mathcal{K}'$ results from *forgetting* a signature Σ in a KB $\mathcal{K}$ if $\mathcal{K}' \equiv_{\mathsf{sig}(\mathcal{K}) \setminus \Sigma} \mathcal{K}$ and $\mathsf{sig}(\mathcal{K}') \subseteq \mathsf{sig}(\mathcal{K}) \setminus \Sigma$. Thus, the result of forgetting Σ does not use Σ and gives the same answers to CQs without symbols in Σ as $\mathcal{K}$.

We investigate the data and combined complexity of deciding Σ-query inseparability for KBs given in various fragments of the DL *Horn-$\mathcal{ALCHI}$* (Krötzsch, Rudolph, and Hitzler 2013), which include *DL-Lite*$^{\mathcal{H}}_{core}$ (Calvanese et al. 2007) and $\mathcal{EL}$ (Baader, Brandt, and Lutz 2005) underlying the W3C profiles *OWL 2 QL* and *OWL 2 EL*. For all of these DLs, Σ-query inseparability turns out to be P-complete for data complexity, which matches the data complexity of CQ evaluation for all of our DLs lying outside the *DL-Lite* family. For combined complexity, the obtained tight complexity results are summarised in the diagram below. Most interesting are ExpTime-completeness of *DL-Lite*$^{\mathcal{H}}_{core}$ and 2ExpTime-completeness of *Horn-$\mathcal{ALCI}$*, which contrast with NP-completeness and ExpTime-completeness of CQ evaluation for those logics. For *DL-Lite* without role inclusions and $\mathcal{ELH}$, Σ-query inseparability is P-complete, while CQ evaluation is NP-complete. In general, it is the combined presence of inverse roles and qualified existential

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

restrictions (or role inclusions) that makes Σ-query inseparability hard. To establish the upper complexity bounds, we develop a uniform game-theoretic technique for checking finite Σ-homomorphic embeddability between (possibly infinite) materialisations of KBs.



Σ-query inseparability for KBs has not been investigated systematically before. The polynomial upper bound for $\mathcal{EL}$ was established as a preliminary step to study TBox inseparability (Lutz and Wolter 2010), and this notion was also used to study forgetting for *DL-Lite*$^{\mathcal{N}}_{bool}$ (Wang et al. 2010).

We apply our results to resolve two important open problems. First, we show that the membership problem for universal UCQ-solutions in knowledge exchange for KBs in *DL-Lite*$^{\mathcal{H}}_{core}$ is ExpTime-complete for combined complexity, which settles an open question of (Arenas et al. 2013), where only PSpace-hardness was established. We also show that Σ-query inseparability of *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes is ExpTime-complete, which closes the PSpace–ExpTime gap that was left open by Konev et al. (2011).

Recall that TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ are Σ-query inseparable if, for *all* Σ-ABoxes $\mathcal{A}$ (which only use concept and role names from Σ), the KBs $(\mathcal{T}_1, \mathcal{A})$ and $(\mathcal{T}_2, \mathcal{A})$ are Σ-query inseparable. TBox and KB inseparabilities have different applications. The former supports ontology engineering when data is not known or changes frequently: one can equivalently replace one TBox with another only if they return the same answers to queries for *every* Σ-ABox. In contrast, KB inseparability is useful in applications where data is stable—such as knowledge exchange or variants of module extraction and forgetting with fixed data—in order to use the KB in a new application or as a compilation step to make CQ answering more efficient. As we show below, TBox and KB Σ-query inseparabilities also have different computational properties.

TBox Σ-query inseparability has been extensively studied (Kontchakov, Wolter, and Zakharyaschev 2010; Lutz and Wolter 2010; Konev et al. 2012). For work on different notions of TBox inseparability and the corresponding notions of modules and forgetting, we refer the reader to (Cuenca Grau et al. 2008; Konev, Walther, and Wolter 2009; Del Vescovo et al. 2011; Nikitina and Rudolph 2012; Nikitina and Glimm 2012; Lutz, Seylan, and Wolter 2012).

Omitted proofs can be found in the full version available at www.dcs.bbk.ac.uk/~roman/KR2014.pdf.

Horn-$\mathcal{ALCHI}$ and its Fragments

All the DLs for which we investigate KB Σ-query inseparability are Horn fragments of $\mathcal{ALCHI}$. To define these DLs, we fix sequences of *individual names* a_i, *concept names* A_i, and *role names* P_i, where $i < \omega$. A *role* is either a role name P_i or an *inverse role* P_i^-; we assume that $(P_i^-)^- = P_i$. $\mathcal{ALCI}$*-concepts*, C, are defined by the grammar

$$C ::= A_i \mid \top \mid \bot \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C,$$

where R is a role. $\mathcal{ALC}$*-concepts* are $\mathcal{ALCI}$-concepts without inverse roles; $\mathcal{EL}$*-concepts* are $\mathcal{ALC}$-concepts without the constructs $\bot$, $\sqcup$, $\neg$ and $\forall R.C$. *DL-Lite*$_{horn}$*-concepts* are $\mathcal{ALCI}$-concepts without $\sqcup$, $\neg$ and $\forall R.C$, in which $C = \top$ in every occurrence of $\exists R.C$. Finally, *DL-Lite*$_{core}$*-concepts* are *DL-Lite*$_{horn}$-concepts without $\sqcap$; in other words, they are *basic concepts* of the form $\bot$, $\top$, A_i or $\exists R.\top$.

For a DL $\mathcal{L}$, an $\mathcal{L}$*-concept inclusion* (CI) takes the form $C \sqsubseteq D$, where C and D are $\mathcal{L}$-concepts. An $\mathcal{L}$*-TBox*, $\mathcal{T}$, contains a finite set of $\mathcal{L}$-CIs. An $\mathcal{ALCHI}$, *DL-Lite*$^{\mathcal{H}}_{horn}$ and *DL-Lite*$^{\mathcal{H}}_{core}$ TBox can also contain a finite set of *role inclusions* (RIs) $R_1 \sqsubseteq R_2$, where the R_i are roles. In $\mathcal{ELH}$ TBoxes, RIs do not have inverse roles. *DL-Lite* TBoxes may also contain *disjointness constraints* $B_1 \sqcap B_2 \sqsubseteq \bot$ and $R_1 \sqcap R_2 \sqsubseteq \bot$, for basic concepts B_i and roles R_i.

To introduce the Horn fragments of these DLs, we require the following (standard) recursive definition (Hustadt, Motik, and Sattler 2005; Kazakov 2009): a concept C occurs positively in C; if C occurs positively (respectively, negatively) in C' then C occurs positively (negatively) in $C' \sqcup D$, $C' \sqcap D$, $\exists R.C'$, $\forall R.C'$, $D \sqsubseteq C'$, and it occurs negatively (positively) in $\neg C'$ and $C' \sqsubseteq D$. Now, we call a TBox $\mathcal{T}$ *Horn* if no concept of the form $C \sqcup D$ occurs positively in $\mathcal{T}$, and no concept of the form $\neg C$ or $\forall R.C$ occurs negatively in $\mathcal{T}$. In the DL *Horn-*$\mathcal{L}$, where $\mathcal{L}$ is one of our DLs, only *Horn-*$\mathcal{L}$-TBoxes are allowed. Clearly, the $\mathcal{EL}$- and *DL-Lite*-TBoxes are Horn by definition.

An *ABox*, $\mathcal{A}$, is a finite set of *assertions* of the form $A_k(a_i)$ or $P_k(a_i, a_j)$. An $\mathcal{L}$-TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ together form an $\mathcal{L}$ *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The set of individual names in $\mathcal{K}$ is denoted by $\mathsf{ind}(\mathcal{K})$.

The semantics for the DLs is defined in the usual way based on interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that comply with the *unique name assumption*: $a_i^{\mathcal{I}} \neq a_j^{\mathcal{I}}$ for $i \neq j$ (Baader et al. 2003). We write $\mathcal{I} \models \alpha$ in case an inclusion or assertion α is true in $\mathcal{I}$. If $\mathcal{I} \models \alpha$, for all $\alpha \in \mathcal{T} \cup \mathcal{A}$, then $\mathcal{I}$ is a *model* of a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$; in symbols: $\mathcal{I} \models \mathcal{K}$. $\mathcal{K}$ is *consistent* if it has a model. $\mathcal{K} \models \alpha$ means that $\mathcal{I} \models \alpha$ for all $\mathcal{I} \models \mathcal{K}$.

A *conjunctive query* (CQ) $\boldsymbol{q}(\vec{x})$ is a formula $\exists \vec{y}\, \varphi(\vec{x}, \vec{y})$, where φ is a conjunction of atoms of the form $A_k(z_1)$ or $P_k(z_1, z_2)$ with $z_i \in \vec{x} \cup \vec{y}$. A tuple $\vec{a} \subseteq \mathsf{ind}(\mathcal{K})$ (of the same length as $\vec{x}$) is a *certain answer* to $\boldsymbol{q}(\vec{x})$ over $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \boldsymbol{q}(\vec{a})$ for all $\mathcal{I} \models \mathcal{K}$; in this case we write $\mathcal{K} \models \boldsymbol{q}(\vec{a})$. If $\vec{x} = \emptyset$, the answer to $\boldsymbol{q}$ is 'yes' if $\mathcal{K} \models \boldsymbol{q}$ and 'no' otherwise.

For combined complexity, the problem '$\mathcal{K} \models \boldsymbol{q}(\vec{a})$?' is NP-complete for the *DL-Lite* logics (Calvanese et al. 2007), $\mathcal{EL}$ and $\mathcal{ELH}$ (Rosati 2007), and ExpTime-complete for the remaining Horn DLs above (Eiter et al. 2008). For data complexity (with fixed $\mathcal{T}$ and $\boldsymbol{q}$), this problem is in AC^0 for the *DL-Lite* logics (Calvanese et al. 2007) and P-complete for the remaining DLs (Rosati 2007; Eiter et al. 2008).

A *signature*, Σ, is a set of concept and role names. By a Σ-concept, Σ-role, Σ-CQ, etc. we understand any concept, role, CQ, etc. constructed using the names from Σ.

Σ-Query Entailment and Inseparability

We define the central notions of this paper.

Definition 1 Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be KBs and Σ a signature.

- $\mathcal{K}_1$ Σ-*query entails* $\mathcal{K}_2$ if $\mathcal{K}_2 \models \boldsymbol{q}(\vec{a})$ implies $\mathcal{K}_1 \models \boldsymbol{q}(\vec{a})$ for *all* Σ-CQs $\boldsymbol{q}(\vec{x})$ and all $\vec{a} \subseteq \mathsf{ind}(\mathcal{K}_2)$.
- $\mathcal{K}_1$ and $\mathcal{K}_2$ are Σ-*query inseparable* if they Σ-query entail each other. In this case we write $\mathcal{K}_1 \equiv_\Sigma \mathcal{K}_2$.

Observe that Σ-query inseparability is weaker than logical equivalence even if $\Sigma = \mathsf{sig}(\mathcal{K}_1) \cup \mathsf{sig}(\mathcal{K}_2)$, where $\mathsf{sig}(\mathcal{K}_i)$ is the signature of $\mathcal{K}_i$. For example, $(\emptyset, \{A(a)\})$ is $\{A, B\}$-query inseparable from $(\{B \sqsubseteq A\}, \{A(a)\})$ but the two KBs are clearly not logically equivalent. Since checking Σ-query inseparability can be reduced to two Σ-query entailment checks, we can prove complexity upper bounds for entailment. Conversely, for most languages we have a semantically transparent reduction of Σ-query entailment to Σ-query inseparability:

Theorem 2 *Let $\mathcal{L}$ be any of our DLs containing $\mathcal{EL}$ or having role inclusions. Then Σ-query entailment for $\mathcal{L}$-KBs is* LogSpace*-reducible to Σ-query inseparability for $\mathcal{L}$-KBs.*

Proof sketch. Let $\mathcal{K}_i = (\mathcal{T}_i, \mathcal{A}_i)$, $i = 1, 2$, and Σ be given. We may assume that $\Sigma = \mathsf{sig}(\mathcal{K}_1) \cap \mathsf{sig}(\mathcal{K}_2)$. We also assume that $\mathcal{L}$ has role inclusions, $\mathcal{K}_1$ and $\mathcal{K}_2$ are consistent and the *trivial interpretation* $\mathcal{I}_\emptyset$ (with $|\Delta^{\mathcal{I}_\emptyset}| = 1$ and $S^{\mathcal{I}_\emptyset} = \emptyset$, for any S) is a model of the $\mathcal{T}_i$ (a proof without those assumptions is given in the full version). Let $\mathcal{K}_i'$ be a copy of $\mathcal{K}_i$ in which all symbols S are replaced by fresh S_i, and let $\mathcal{K}_i^\Sigma$ extend $\mathcal{K}_i'$ with $S_i \sqsubseteq S$, for $S \in \Sigma$. One can show that $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$ iff $\mathcal{K}_1 \equiv_\Sigma \mathcal{K}_1^\Sigma \cup \mathcal{K}_2^\Sigma$. ❑

That $\mathcal{I}_\emptyset \models \mathcal{K}_i$ is essential in the reduction above. Take $\mathcal{T}_1 = \{A \sqsubseteq B, A \sqsubseteq \exists R.C\}$, $\mathcal{T}_2 = \{\top \sqsubseteq B, C \sqcap B \sqsubseteq \bot\}$ and $\Sigma = \{A, B, R, C\}$. Then $\mathcal{K}_1 = (\mathcal{T}_1, \{A(a)\})$ Σ-query entails $\mathcal{K}_2 = (\mathcal{T}_2, \{A(a)\})$ but $\mathcal{K}_1 \not\equiv_\Sigma \mathcal{K}_1^\Sigma \cup \mathcal{K}_2^\Sigma$.

We now consider the relationship between inseparability and universal UCQ-solutions in knowledge exchange. Suppose $\mathcal{K}_1$ and $\mathcal{K}_2$ are KBs in disjoint signatures Σ_1 and Σ_2. Let $\mathcal{T}_{12}$ be a mapping consisting of inclusions of the form $S_1 \sqsubseteq S_2$, where the S_i are concept (or role) names in Σ_i. Then $\mathcal{K}_2$ is a universal UCQ-solution for $(\mathcal{K}_1, \mathcal{T}_{12}, \Sigma_2)$ if $\mathcal{K}_1 \cup \mathcal{T}_{12} \equiv_{\Sigma_2} \mathcal{K}_2$. Deciding the latter is called the *membership problem for universal UCQ-solutions*. For DLs $\mathcal{L}$ with role inclusions, the problem whether $\mathcal{K}_1 \cup \mathcal{T}_{12} \equiv_{\Sigma_2} \mathcal{K}_2$ is a Σ_2-query inseparability problem in $\mathcal{L}$. Conversely, we have:

Theorem 3 *Σ-query entailment for any of our DLs $\mathcal{L}$ is* LogSpace*-reducible to the membership problem for universal UCQ-solutions in $\mathcal{L}$.*

Proof sketch. We want to decide whether $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$. We again assume that $\mathcal{I}_\emptyset \models \mathcal{T}_i$ and use the proof of Theorem 2 (for the general case, see the full version). We may assume that $\Sigma = \mathsf{sig}(\mathcal{K}_1) \cap \mathsf{sig}(\mathcal{K}_2)$. Let $\Sigma_1 = \mathsf{sig}(\mathcal{K}_1)$. Then $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$ iff $\mathcal{K}_1$ Σ_1-query entails $\mathcal{K}_2$. By the proof of Theorem 2, the latter is the case iff $\mathcal{K}_1$ Σ_1-query entails $\mathcal{K}_1^{\Sigma_1} \cup \mathcal{K}_2^{\Sigma_1}$. Clearly, $\mathcal{K}_1^{\Sigma_1} \cup \mathcal{K}_2^{\Sigma_1}$ Σ_1-query entails $\mathcal{K}_1$, and so the two KBs are Σ_1-query inseparable. Then $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$ iff $\mathcal{K}_1$ is a universal UCQ-solution for $(\mathcal{K}_1' \cup \mathcal{K}_2', \mathcal{T}_{12}, \Sigma_1)$, where $\mathcal{T}_{12} = \{S_1 \sqsubseteq S, S_2 \sqsubseteq S \mid S \in \Sigma_1\}$. ❑

Semantic Characterisation

In this section, we give a semantic characterisation of KB Σ-query entailment based on an abstract notion of materialisation and finite homomorphisms between such structures.

Let $\mathcal{K}$ be a KB. An interpretation $\mathcal{I}$ is called a *materialisation* of $\mathcal{K}$ if, for all CQs $\boldsymbol{q}(\vec{x})$ and tuples $\vec{a} \subseteq \mathsf{ind}(\mathcal{K})$,

$$\mathcal{K} \models \boldsymbol{q}(\vec{a}) \quad \text{iff} \quad \mathcal{I} \models \boldsymbol{q}(\vec{a}).$$

We say that $\mathcal{K}$ is *materialisable* if it has a materialisation.

Materialisations can be used to characterise KB Σ-query entailment by means of Σ-homomorphisms. For an interpretation $\mathcal{I}$ and a signature Σ, the Σ-*types* $\boldsymbol{t}_\Sigma^\mathcal{I}(x)$ and $\boldsymbol{r}_\Sigma^\mathcal{I}(x, y)$ of $x, y \in \Delta^\mathcal{I}$ are defined by taking:

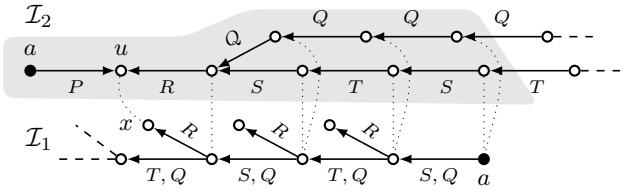
$$\boldsymbol{t}_\Sigma^\mathcal{I}(x) = \{\, \Sigma\text{-concept name } A \mid x \in A^\mathcal{I} \,\},$$
$$\boldsymbol{r}_\Sigma^\mathcal{I}(x, y) = \{\, \Sigma\text{-role } R \mid (x, y) \in R^\mathcal{I} \,\}.$$

Suppose $\mathcal{I}_i$ is a materialisation of $\mathcal{K}_i$, $i = 1, 2$. A function $h\colon \Delta^{\mathcal{I}_2} \to \Delta^{\mathcal{I}_1}$ is a Σ-*homomorphism* from $\mathcal{I}_2$ to $\mathcal{I}_1$ if, for any $a \in \mathsf{ind}(\mathcal{K}_2)$ and any $x, y \in \Delta^{\mathcal{I}_2}$,

- $h(a^{\mathcal{I}_2}) = a^{\mathcal{I}_1}$ whenever $\boldsymbol{t}_\Sigma^{\mathcal{I}_2}(a) \neq \emptyset$ or $\boldsymbol{r}_\Sigma^{\mathcal{I}_2}(a, y) \neq \emptyset$ for some $y \in \Delta^{\mathcal{I}_2}$, and
- $\boldsymbol{t}_\Sigma^{\mathcal{I}_2}(x) \subseteq \boldsymbol{t}_\Sigma^{\mathcal{I}_1}(h(x))$, $\boldsymbol{r}_\Sigma^{\mathcal{I}_2}(x, y) \subseteq \boldsymbol{r}_\Sigma^{\mathcal{I}_1}(h(x), h(y))$.

As answers to Σ-CQs are preserved under Σ-homomorphisms, $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$ if there is a Σ-homomorphism from $\mathcal{I}_2$ to $\mathcal{I}_1$. However, the converse does not hold:

Example 4 Suppose $\mathcal{I}_2$ and $\mathcal{I}_1$ below are materialisations of KBs $\mathcal{K}_2$ and $\mathcal{K}_1$, where a is the only ABox individual:



Let $\Sigma = \{Q, R, S, T\}$. Then there is no Σ-homomorphism from $\mathcal{I}_2$ to $\mathcal{I}_1$ (as $\boldsymbol{r}_\Sigma^{\mathcal{I}_2}(a, u) = \emptyset$, we can map u to, say, x but then only the shaded part of $\mathcal{I}_2$ can be mapped Σ-homomorphically to $\mathcal{I}_1$). However, for any Σ-query $\boldsymbol{q}(\vec{x})$, $\mathcal{I}_2 \models \boldsymbol{q}(\vec{a})$ implies $\mathcal{I}_1 \models \boldsymbol{q}(\vec{a})$ as any finite subinterpretation of $\mathcal{I}_2$ can be Σ-homomorphically mapped to $\mathcal{I}_1$.

We say that $\mathcal{I}_2$ is *finitely Σ-homomorphically embeddable into* $\mathcal{I}_1$ if, for every *finite* subinterpretation $\mathcal{I}_2'$ of $\mathcal{I}_2$, there exists a Σ-homomorphism from $\mathcal{I}_2'$ to $\mathcal{I}_1$.

To prove the following theorem, one can regard any finite subinterpretation of $\mathcal{I}_2$ as a CQ whose variables are elements of $\Delta^{\mathcal{I}_2}$, with the answer variables being in $\mathsf{ind}(\mathcal{K}_2)$.

Theorem 5 *Suppose $\mathcal{K}_i$ is a consistent KB with a materialisation $\mathcal{I}_i$, $i = 1, 2$. Then $\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$ iff $\mathcal{I}_2$ is finitely Σ-homomorphically embeddable into $\mathcal{I}_1$.*

One problem with applying Theorem 5 is that materialisations are in general infinite for any of the DLs considered in this paper. We address this problem by introducing finite representations of materialisations. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a KB and let $\mathcal{G} = (\Delta^\mathcal{G}, \cdot^\mathcal{G}, \rightsquigarrow)$ be a finite structure such that $\Delta^\mathcal{G} = \mathsf{ind}(\mathcal{K}) \cup \Omega$, for $\mathsf{ind}(\mathcal{K}) \cap \Omega = \emptyset$, $\cdot^\mathcal{G}$ is an interpretation

function on $\Delta^{\mathcal{G}}$ with $A_i^{\mathcal{G}} \subseteq \Delta^{\mathcal{G}}$, $P_i^{\mathcal{G}} \subseteq \mathsf{ind}(\mathcal{K}) \times \mathsf{ind}(\mathcal{K})$, and $(\Delta^{\mathcal{G}}, \rightsquigarrow)$ is a directed graph (containing loops) with nodes $\Delta^{\mathcal{G}}$ and edges $\rightsquigarrow \subseteq \Delta^{\mathcal{G}} \times \Omega$, in which every edge $u \rightsquigarrow v$ is labelled with a set $(u,v)^{\mathcal{G}} \neq \emptyset$ of roles satisfying the condition: if $u_1 \rightsquigarrow v$ and $u_2 \rightsquigarrow v$, then $(u_1,v)^{\mathcal{G}} = (u_2,v)^{\mathcal{G}}$. We call $\mathcal{G}$ a *generating structure for* $\mathcal{K}$ if the interpretation $\mathcal{M}$ defined below is a materialisation of $\mathcal{K}$.

A *path* in $\mathcal{G}$ is a sequence $\sigma = u_0 \dots u_n$ with $u_0 \in \mathsf{ind}(\mathcal{K})$ and $u_i \rightsquigarrow u_{i+1}$ for $i < n$. Let $\mathsf{tail}(\sigma) = u_n$ and let $\mathsf{path}(\mathcal{G})$ be the set of paths in $\mathcal{G}$. The materialisation $\mathcal{M}$ is given by:

$$\Delta^{\mathcal{M}} = \mathsf{path}(\mathcal{G}), \quad a^{\mathcal{M}} = a, \ \text{ for } a \in \mathsf{ind}(\mathcal{K}),$$
$$A^{\mathcal{M}} = \{\sigma \mid \mathsf{tail}(\sigma) \in A^{\mathcal{G}}\},$$
$$P^{\mathcal{M}} = P^{\mathcal{G}} \cup \{(\sigma, \sigma u) \mid \mathsf{tail}(\sigma) \rightsquigarrow u,\ P \in (\mathsf{tail}(\sigma), u)^{\mathcal{G}}\}$$
$$\cup \{(\sigma u, \sigma) \mid \mathsf{tail}(\sigma) \rightsquigarrow u,\ P^- \in (\mathsf{tail}(\sigma), u)^{\mathcal{G}}\}.$$

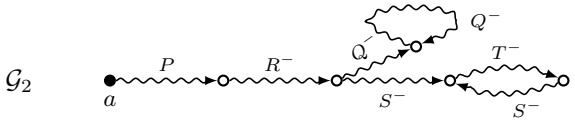
We say that a DL $\mathcal{L}$ has *finitely generated materialisations* if every $\mathcal{L}$-KB has a generating structure.

Theorem 6 *Horn-$\mathcal{ALCHI}$ and all of its fragments defined above have finitely generated materialisations. Moreover,*

- *for any $\mathcal{L} \in \{\mathcal{ALCHI}, \mathcal{ALCI}, \mathcal{ALCH}, \mathcal{ALC}\}$ and any Horn-$\mathcal{L}$ KB $(\mathcal{T}, \mathcal{A})$, a generating structure can be constructed in time $|\mathcal{A}| \cdot 2^{p(|\mathcal{T}|)}$, p a polynomial*;
- *for any $\mathcal{L}$ in the $\mathcal{EL}$ and DL-Lite families introduced above and any $\mathcal{L}$-KB $(\mathcal{T}, \mathcal{A})$, a generating structure can be constructed in time $|\mathcal{A}| \cdot p(|\mathcal{T}|)$, p a polynomial.*

Finite generating structures have been defined for $\mathcal{EL}$ (Lutz, Toman, and Wolter 2009), *DL-Lite* (Kontchakov et al. 2010) and more expressive Horn DLs (Eiter et al. 2008). With the exception of *DL-Lite*, however, the relation $\rightsquigarrow$ guiding the construction of materialisations was implicit. We show how the existing constructions can be converted to generating structures in the full version.

Example 7 The materialisation $\mathcal{I}_2$ from Example 4 can be generated by the structure $\mathcal{G}_2$ shown below:



For a generating structure $\mathcal{G}$ for $\mathcal{K}$ and a signature Σ, the Σ-*types* $\boldsymbol{t}_\Sigma^{\mathcal{G}}(u)$ and $\boldsymbol{r}_\Sigma^{\mathcal{G}}(u,v)$ of $u, v \in \Delta^{\mathcal{G}}$ are defined by:

$$\boldsymbol{t}_\Sigma^{\mathcal{G}}(u) = \{\, \Sigma\text{-concept name } A \mid u \in A^{\mathcal{G}} \,\},$$
$$\boldsymbol{r}_\Sigma^{\mathcal{G}}(u,v) = \begin{cases} \{\, \Sigma\text{-role } R \mid (u,v) \in R^{\mathcal{G}} \,\}, & \text{if } u, v \in \mathsf{ind}(\mathcal{K}), \\ \{\, \Sigma\text{-role } R \mid R \in (u,v)^{\mathcal{G}} \,\}, & \text{if } u \rightsquigarrow v, \\ \emptyset, & \text{otherwise,} \end{cases}$$

where $(P^-)^{\mathcal{G}}$ is the converse of $P^{\mathcal{G}}$. We also define $\bar{\boldsymbol{r}}_\Sigma^{\mathcal{G}}(u,v)$ to contain the inverses of the roles in $\boldsymbol{r}_\Sigma^{\mathcal{G}}(u,v)$; note that $\bar{\boldsymbol{r}}_\Sigma^{\mathcal{G}}(u,v)$ is not the same as $\boldsymbol{r}_\Sigma^{\mathcal{G}}(v,u)$; cf. the T^-, S^--cycle in Example 7. We write $u \rightsquigarrow^\Sigma v$ if $u \rightsquigarrow v$ and $\boldsymbol{r}_\Sigma^{\mathcal{G}}(u,v) \neq \emptyset$.

In the next section, we show that, for a DL $\mathcal{L}$ having finitely generated materialisations, the problem of checking Σ-query entailment between $\mathcal{L}$-KBs can be reduced to the problem of finding a winning strategy in a game played on the generating structures for these KBs.

Σ-Query Entailment by Games

Suppose a DL $\mathcal{L}$ has finitely generated materialisations, $\mathcal{K}_i$ is a consistent $\mathcal{L}$-KB, for $i = 1, 2$, and Σ a signature. Let $\mathcal{G}_i = (\Delta^{\mathcal{G}_i}, \cdot^{\mathcal{G}_i}, \rightsquigarrow_i)$ be a generating structure for $\mathcal{K}_i$ and let $\mathcal{M}_i$ be its materialisation; $\mathcal{G}_i^\Sigma$ and $\mathcal{M}_i^\Sigma$ denote the restrictions of $\mathcal{G}_i$ and $\mathcal{M}_i$ to Σ.

We begin with a very simple game on the finite generating structure $\mathcal{G}_2^\Sigma$ and the possibly infinite materialisation $\mathcal{M}_1^\Sigma$.

Infinite game $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$**.** This game is played by two players: player 2 and player 1. The *states* of the game are of the form $\mathfrak{s}_i = (u_i \mapsto \sigma_i)$, for $i \geq 0$, where $u_i \in \Delta^{\mathcal{G}_2}$ and $\sigma_i \in \Delta^{\mathcal{M}_1}$ satisfy the following condition:

(s$_1$) $\boldsymbol{t}_\Sigma^{\mathcal{G}_2}(u_i) \subseteq \boldsymbol{t}_\Sigma^{\mathcal{M}_1}(\sigma_i)$.

The game starts in a state $\mathfrak{s}_0 = (u_0 \mapsto \sigma_0)$ with $\sigma_0 = u_0$ in case $u_0 \in \mathsf{ind}(\mathcal{K}_2)$. In each round $i > 0$, player 2 challenges player 1 with some $u_i \in \Delta^{\mathcal{G}_2}$ such that $u_{i-1} \rightsquigarrow_2^\Sigma u_i$. Player 1 has to respond with a $\sigma_i \in \Delta^{\mathcal{M}_1}$ satisfying **(s$_1$)** and

(s$_2$) $\boldsymbol{r}_\Sigma^{\mathcal{G}_2}(u_{i-1}, u_i) \subseteq \boldsymbol{r}_\Sigma^{\mathcal{M}_1}(\sigma_{i-1}, \sigma_i)$.

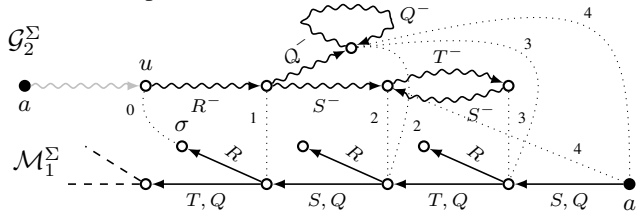
This gives the next state $\mathfrak{s}_i = (u_i \mapsto \sigma_i)$. Note that of all the u_i only u_0 may be an ABox individual; however, there is no such a restriction on the σ_i. A *play of length* $n \geq 0$ *starting from* $\mathfrak{s}_0$ is any sequence $\mathfrak{s}_0, \dots, \mathfrak{s}_n$ of states obtained as described above. For an ordinal $\lambda \leq \omega$, we say that player 1 has a λ-*winning strategy in the game* $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ *starting from a state* $\mathfrak{s}_0$ if, for any play of length $i < \lambda$, which starts from $\mathfrak{s}_0$ and conforms with this strategy, and any challenge of player 2 in round $i+1$, player 1 has a response.

The following theorem gives a game-theoretic flavour to the criterion of Theorem 5 (see the full paper for a proof).

Theorem 8 *$\mathcal{M}_2$ is finitely Σ-homomorphically embeddable into $\mathcal{M}_1$ iff the following conditions hold*:

(abox) $\boldsymbol{r}_\Sigma^{\mathcal{M}_2}(a,b) \subseteq \boldsymbol{r}_\Sigma^{\mathcal{M}_1}(a,b)$*, for any* $a, b \in \mathsf{ind}(\mathcal{K}_2)$;
(win) *for any $u_0 \in \Delta^{\mathcal{G}_2}$ and $n < \omega$, there exists $\sigma_0 \in \Delta^{\mathcal{M}_1}$ such that player 1 has an n-winning strategy in the game $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ starting from $(u_0 \mapsto \sigma_0)$.*

Example 9 Let $\Sigma = \{Q, R, S, T\}$. Consider $\mathcal{G}_2^\Sigma$ and $\mathcal{M}_1^\Sigma$ shown in the picture below:



For any $n < \omega$ and $u \in \Delta^{\mathcal{G}_2}$, player 1 has an n-winning strategy in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$. A 4-winning strategy starting from $(u \mapsto \sigma)$ is shown by dotted lines (in round 2, player 2 has two possible challenges). For a larger n, a suitable σ can be chosen further away from the root a of $\mathcal{M}_1$.

The criterion of Theorem 8 does not seem to be a big improvement on Theorem 5 as we still have to deal with an infinite materialisation. Our aim now is to show that condition **(win)** in the infinite game $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ can be checked

by analysing a more complex game on the *finite* generating structures $\mathcal{G}_2$ and $\mathcal{G}_1$. We consider four types of strategies in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$. For each type, τ, we define a game $G^\tau_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ such that, for any $u_0 \in \Delta^{\mathcal{G}_2}$, the following conditions are equivalent:

($< \omega$) for every $n < \omega$, player 1 has an n-winning strategy of type τ in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ starting from some $(u_0 \mapsto \sigma^n_0)$;

(ω) player 1 has an ω-winning strategy in $G^\tau_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ starting from some state depending on u_0 and τ.

We start by considering 'forward' winning strategies that are sufficient for the DLs without inverse roles.

Forward strategy and game $G^f_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$. We say that a λ-strategy ($\lambda \leq \omega$) for player 1 in the game $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ is *forward* if, for any play of length $i - 1 < \lambda$, which conforms with this strategy, and any challenge $u_{i-1} \rightsquigarrow^\Sigma_2 u_i$ by player 2, the response σ_i of player 1 is such that either $\sigma_{i-1}, \sigma_i \in \mathsf{ind}(\mathcal{K}_1)$ or $\sigma_i = \sigma_{i-1}v$, for some $v \in \Delta^{\mathcal{G}_1}$.

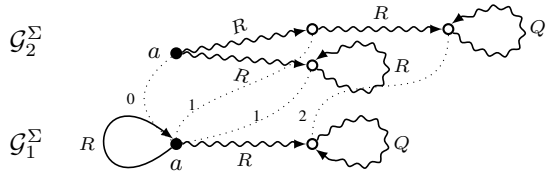
For example, if the $\mathcal{G}_i$, $i = 1, 2$, satisfy the condition

(f) the Σ-labels on $\rightsquigarrow_i$-edges contain no inverse roles,

then *every* strategy in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ is forward. This is clearly the case for *Horn-$\mathcal{ALCH}$*, *Horn-$\mathcal{ALC}$*, $\mathcal{ELH}$ and $\mathcal{EL}$, which by definition do not have inverse roles.

The existence of a forward λ-winning strategy for player 1 in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ is equivalent to the existence of such a strategy in the game $G^f_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$, which is defined similarly to $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ but with two modifications: (1) it is played on $\mathcal{G}_2$ and $\mathcal{G}_1$; and (2) the response $x_i \in \Delta^{\mathcal{G}_1}$ of player 1 to a challenge $u_{i-1} \rightsquigarrow^\Sigma_2 u_i$ must be such that either $x_{i-1}, x_i \in \mathsf{ind}(\mathcal{K}_1)$ or $x_{i-1} \rightsquigarrow_1 x_i$, and **($\mathbf{s}_1$)–($\mathbf{s}_2$)** hold (with $\mathcal{G}_1$ and x_i in place of $\mathcal{M}_1$ and σ_i).

Example 10 Let $\mathcal{G}_2$ and $\mathcal{G}_1$ be as shown below. Then, for any $u \in \Delta^{\mathcal{G}_2}$, there is $x \in \Delta^{\mathcal{G}_1}$ such that player 1 has an ω-winning strategy in $G^f_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ starting from $(u \mapsto x)$.



The next theorem follows from König's Lemma:

Lemma 11 *For $u_0 \in \Delta^{\mathcal{G}_2}$, condition ($< \omega$) holds for forward strategies in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ iff (ω) holds in $G^f_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ for some state $(u_0 \mapsto x_0)$.*

$G^f_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ is a standard simulation or reachability game on finite graphs, where the existence of ω-winning strategies for player 1 follows from the existence of n-winning strategies for $n = O(|\mathcal{G}_2| \times |\mathcal{G}_1|)$, which can be checked in polynomial time (Mazala 2001; Baier and Katoen 2007). By Theorem 6 and **(f)**, we obtain:

Theorem 12 *For combined complexity, checking Σ-query entailment is in* P *for $\mathcal{EL}$ and $\mathcal{ELH}$ KBs, and in* EXPTIME *for Horn-$\mathcal{ALC}$ and Horn-$\mathcal{ALCH}$ KBs. For data complexity, it is in* P *for all these DLs.*

In comparison to forward strategies, the winning strategies used in Example 9 can be described as 'backward.'

Backward strategy and game $G^b_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$. A λ-strategy for player 1 in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ is *backward* if, for any play of length $i - 1 < \lambda$, which conforms with this strategy, and any challenge $u_{i-1} \rightsquigarrow^\Sigma_2 u_i$ by player 2, the response σ_i of player 1 is the *immediate predecessor* of σ_{i-1} in $\mathcal{M}_1$ in the sense that $\sigma_{i-1} = \sigma_i w$, for some $w \in \Delta^{\mathcal{G}_1}$ (player 1 loses in case $\sigma_{i-1} \in \mathsf{ind}(\mathcal{K}_1)$). Note that, since $\mathcal{M}_1$ is tree-shaped, the response of player 1 to any different challenge $u_{i-1} \rightsquigarrow^\Sigma_2 u'_i$ must be the same σ_i; cf. Example 9.

That is why the states of the game $G^b_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ are of the form $\mathfrak{s}_i = (\Xi_i \mapsto x_i)$, where $\Xi_i \subseteq \Delta^{\mathcal{G}_2}$, $\Xi_i \neq \emptyset$, and $x_i \in \Delta^{\mathcal{G}_1}$ satisfy the following condition:

($\mathbf{s}'_1$) $\boldsymbol{t}^{\mathcal{G}_2}_\Sigma(u) \subseteq \boldsymbol{t}^{\mathcal{G}_1}_\Sigma(x_i)$, for all $u \in \Xi_i$.

The game starts in a state $\mathfrak{s}_0 = (\Xi_0 \mapsto x_0)$ such that

($\mathbf{s}'_0$) if $u \in \Xi_0 \cap \mathsf{ind}(\mathcal{K}_2)$, then $x_0 = u \in \mathsf{ind}(\mathcal{K}_1)$.

For each $i > 0$, player 2 always challenges player 1 with the set $\Xi_i = \Xi^{\rightsquigarrow}_{i-1}$, where

$$\Xi^{\rightsquigarrow} = \{v \in \Delta^{\mathcal{G}_2} \mid u \rightsquigarrow^\Sigma_2 v, \text{ for some } u \in \Xi\},$$

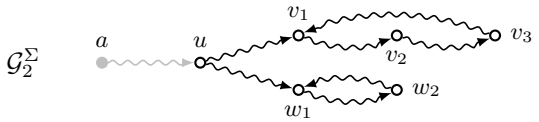
provided that it is not empty (otherwise, player 2 loses). Player 1 responds with $x_i \in \Delta^{\mathcal{G}_1}$ such that $x_i \rightsquigarrow_1 x_{i-1}$ and **($\mathbf{s}'_1$)** and the following condition hold:

($\mathbf{s}'_2$) $\boldsymbol{r}^{\mathcal{G}_2}_\Sigma(u, v) \subseteq \bar{\boldsymbol{r}}^{\mathcal{G}_1}_\Sigma(x_{i-1}, x_i)$, for all $u \in \Xi_{i-1}$, $v \in \Xi_i$.

Lemma 13 *For $u_0 \in \Delta^{\mathcal{G}_2}$, condition ($< \omega$) holds for backward strategies in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ iff (ω) holds in $G^b_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ for some state $(\{u_0\} \mapsto x_0)$.*

Although Lemmas 11 and 13 look similar, the game $G^b_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ turns out to be more complex than $G^f_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$.

Example 14 To illustrate, consider $\mathcal{G}^\Sigma_2$ shown below (with concepts and roles omitted) and an arbitrary $\mathcal{G}_1$:



A play in $G^b_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ may proceed as: $(\{u\} \mapsto x_0)$, $(\{v_1, w_1\} \mapsto x_1)$, $(\{v_2, w_2\} \mapsto x_2)$, $(\{v_3, w_1\} \mapsto x_3)$, etc. This gives at least 6 different sets Ξ_i. But if $\mathcal{G}_2$ contained k cycles of lengths $p_1, \dots, p_k$, where p_i is the ith prime number, then the number of states in $G^b_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ could be exponential ($p_1 \times \dots \times p_k$). In fact, we have the following:

Lemma 15 *Checking (ω) in Lemma 13 is* CONP*-hard.*

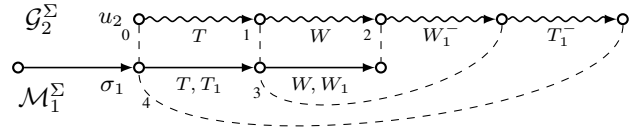
Observe that in the case of *DL-Lite$_{core}$* and *DL-Lite$_{horn}$* (which have inverse roles but no RIs), generating structures $\mathcal{G} = (\Delta^{\mathcal{G}}, \cdot^{\mathcal{G}}, \rightsquigarrow)$ can be defined so that, for any $u \in \Delta^{\mathcal{G}}$ and R, there is *at most one* v with $u \rightsquigarrow v$ and $R \in \boldsymbol{r}^{\mathcal{G}}(u, v)$ (Kontchakov et al. 2010). As a result, any n-winning strategy starting from $(u_0 \mapsto \sigma_0)$ in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ consists of a (possibly empty) backward part followed by a (possibly empty) forward part. Moreover, in the backward games for these DLs, the sets Ξ_i are always *singletons*. Thus, the number of states in the combined backward/forward games on the $\mathcal{G}_i$ is polynomial, and the existence of winning strategies can be checked in polynomial time.

Theorem 16 *Checking Σ-query entailment for DL-Lite$_{core}$ and DL-Lite$_{horn}$ KBs is in* P *for both combined and data complexity.*

An arbitrary strategy for player 1 in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ is a combination of a backward strategy and a number of start-bounded strategies to be defined next.

Start-bounded strategy and game $G^s_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$. A strategy for player 1 in the game $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ starting from a state $(u_0 \mapsto \sigma_0)$ is *start-bounded* if it never leads to $(u_i \mapsto \sigma_i)$ such that $\sigma_0 = \sigma_i v$, for some v and $i > 0$. In other words, player 1 cannot use those elements of $\mathcal{M}_1$ that are located closer to the ABox than σ_0; the ABox individuals in $\mathcal{M}_1$ can only be used if $\sigma_0 \in \mathsf{ind}(\mathcal{K}_1)$.

Example 17 The strategy starting from $(u_2 \mapsto \sigma_1)$ and shown below is start-bounded:



In the game $G^s_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$, player 1 will have to guess *all* the points of $\mathcal{G}_2$ that are mapped to the same point of $\mathcal{M}_1$.

The *states* of $G^s_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ are of the form $(\Gamma_i, \Xi_i \mapsto x_i)$, $i \geq 0$, where $\Gamma_i, \Xi_i \subseteq \Delta^{\mathcal{G}_2}$, $\Xi_i \neq \emptyset$, $x_i \in \Delta^{\mathcal{G}_1}$ and **($\mathbf{s}'_1$)** holds. The initial state is of the form $(\emptyset, \Xi_0 \mapsto x_0)$ such that **($\mathbf{s}'_0$)** holds. In each round $i > 0$, player 2 challenges player 1 with some $u \rightsquigarrow^\Sigma_2 v$ such that $u \in \Xi_{i-1}$ and

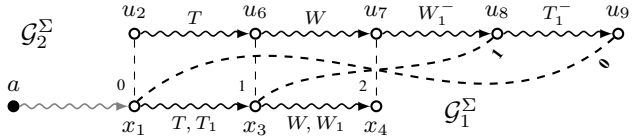
(nbk) if $v \in \Gamma_{i-1}$ then $\boldsymbol{r}^{\mathcal{G}_2}_\Sigma(u, v) \not\subseteq \bar{\boldsymbol{r}}^{\mathcal{G}_1}_\Sigma(x_{i-2}, x_{i-1})$.

Player 1 responds with either a state $(\Xi_{i-1}, \Xi_i \mapsto x_i)$ such that $x_{i-1} \rightsquigarrow_1 x_i$ (and so $x_i \notin \mathsf{ind}(\mathcal{K}_1)$) and **($\mathbf{s}''_2$)** holds, or a state $(\emptyset, \Xi_i \mapsto x_i)$ such that $x_{i-1}, x_i \in \mathsf{ind}(\mathcal{K}_1)$ and

($\mathbf{s}''_2$) $\boldsymbol{r}^{\mathcal{G}_2}_\Sigma(u, v) \subseteq \boldsymbol{r}^{\mathcal{G}_1}_\Sigma(x_{i-1}, x_i)$.

We make challenges $u \rightsquigarrow^\Sigma_2 v$, for which $u \in \Xi_{i-1}$ and **(nbk)** does not hold, 'illegitimate' because x_{i-2} can always be used as a response. Because of this, player 1 always moves 'forward' in $\mathcal{G}_1$, but has to guess appropriate sets Ξ_i in advance. Note that Γ_i is always uniquely determined by x_{i-1}, x_i and Ξ_{i-1} (and it is either Ξ_{i-1} or empty).

Example 18 Let $\mathcal{G}^\Sigma_2$ and $\mathcal{G}^\Sigma_1$ be as follows (cf. Example 17):



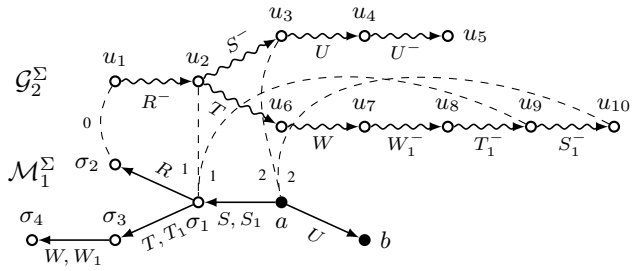
We show that player 1 has an ω-winning strategy in $G^s_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ starting from $(\emptyset, \{u_2, u_9\} \mapsto x_1)$. Player 2 challenges with $u_2 \rightsquigarrow^\Sigma_2 u_6$, and player 1 responds with $(\{u_2, u_9\}, \{u_6, u_8\} \mapsto x_3)$. Then player 2 picks $u_6 \rightsquigarrow^\Sigma_2 u_7$ and player 1 responds with $(\{u_6, u_8\}, \{u_7\} \mapsto x_4)$, where the game ends. Note the crucial guesses $\{u_2, u_9\} \mapsto x_1$ and $\{u_6, u_8\} \mapsto x_3$ made by player 1. If player 1 responded with $(\{u_2, u_9\}, \{u_6\} \mapsto x_3)$ (and failed to guess that u_8 must also be mapped to x_3), then after the challenge $u_6 \rightsquigarrow^\Sigma_2 u_7$ and response $(\{u_6\}, \{u_7\} \mapsto x_4)$), player 2 would pick $u_7 \rightsquigarrow^\Sigma_2 u_8$, to which player 1 could not respond.

Lemma 19 *For any $u_0 \in \Delta^{\mathcal{G}_2}$, condition $(< \omega)$ holds for start-bounded strategies in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ iff (ω) holds in $G^s_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ for some state $(\emptyset, \Xi_0 \mapsto x_0)$ with $u_0 \in \Xi_0$.*

As we shall see in the next section, the problem of checking the conditions of this lemma is ExpTime-hard.

Arbitrary strategies and game $G^a_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$. An arbitrary winning strategy in the game $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ can be composed of one backward and a number of start-bounded strategies.

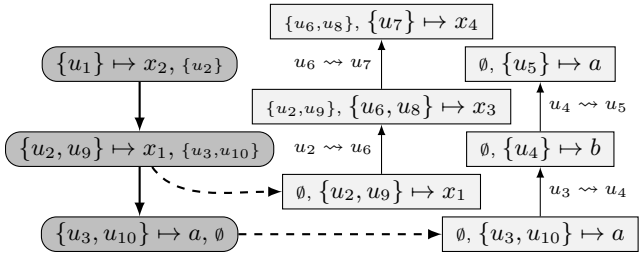
Example 20 Consider $\mathcal{G}^\Sigma_2$ and $\mathcal{M}^\Sigma_1$ shown below:



Starting from $(u_1 \mapsto \sigma_2)$, player 1 can respond to the challenges $u_1 \rightsquigarrow^\Sigma_2 u_2 \rightsquigarrow^\Sigma_2 u_3$ according to the backward strategy; the challenges $u_2 \rightsquigarrow^\Sigma_2 u_6 \rightsquigarrow^\Sigma_2 u_7 \rightsquigarrow^\Sigma_2 u_8 \rightsquigarrow^\Sigma_2 u_9$ according to the start-bounded strategy as in Example 17; the challenges $u_3 \rightsquigarrow^\Sigma_2 u_4 \rightsquigarrow^\Sigma_2 u_5$ also according to the obvious start-bounded strategy; finally, the challenge $u_9 \rightsquigarrow^\Sigma_2 u_{10}$ needs a response according to the backward strategy. We will combine the two backward strategies into a single one, but keep the start-bounded ones separate.

The game $G^a_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ begins as $G^b_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$, but with states of the form $(\Xi_i \mapsto x_i, \Psi_i)$, $i \geq 0$, where $\Xi_i \subseteq \Delta^{\mathcal{G}_2}$ and $x_i \in \Delta^{\mathcal{G}_1}$ satisfy **($\mathbf{s}'_1$)** and Ψ_i is a (possibly empty) subset of $\Xi_i^{\rightsquigarrow}$, which indicates initial challenges in start-bounded games. The initial state satisfies **($\mathbf{s}'_0$)**. In each round $i > 0$, if $x_{i-1} \in \mathsf{ind}(\mathcal{K}_1)$ then player 2 launches the start-bounded game $G^s_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ with the initial state $(\emptyset, \Xi_{i-1} \mapsto x_{i-1})$. Otherwise, if $x_{i-1} \notin \mathsf{ind}(\mathcal{K}_1)$, player 2 has two options. First, he can challenge player 1 with the set Ψ_{i-1} (that is, similar to the backward game but with a possibly smaller Ψ_{i-1} in place of $\Xi_{i-1}^{\rightsquigarrow}$); player 1 responds to this challenge with a state $(\Xi_i \mapsto x_i, \Psi_i)$ such that $\Psi_{i-1} \subseteq \Xi_i$, $x_i \rightsquigarrow_1 x_{i-1}$ and **($\mathbf{s}'_2$)** holds. Second, player 2 can launch the start-bounded game $G^s_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ with the initial state $(\emptyset, \Xi_{i-1} \mapsto x_{i-1})$, where the first challenge of player 2 must be picked from $\Phi_{i-1} = \Xi_{i-1}^{\rightsquigarrow} \setminus \Psi_{i-1}$.

Example 21 We illustrate the ω-winning strategy for player 1 in $G^a_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ starting from $(\{u_1\} \mapsto x_2, \{u_2\})$, where $\mathcal{G}^\Sigma_2$ is from Example 20 and $\mathcal{G}^\Sigma_1$ looks like $\mathcal{M}^\Sigma_1$ from Example 20 (but with x_i in place of σ_i):



Lemma 22 *For any $u_0 \in \Delta^{\mathcal{G}_2}$, condition $(<\omega)$ holds for arbitrary strategies in $G_\Sigma(\mathcal{G}_2, \mathcal{M}_1)$ iff (ω) holds in $G^a_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ for some state $(\Xi_0 \mapsto x_0, \Psi_0)$ with $u_0 \in \Xi_0$.*

Condition (ω) in the lemma above is checked in time $O(|\mathsf{ind}(\mathcal{K}_2)| \times 2^{|\Delta^{\mathcal{G}_2} \setminus \mathsf{ind}(\mathcal{K}_2)|} \times |\Delta^{\mathcal{G}_1}|)$, which can be readily seen by analysing the full game graph for $G^a_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$ (similar to that in Example 21). By Theorem 6, we then obtain:

Theorem 23 *For combined complexity, Σ-query entailment is in* 2ExpTime *for Horn-$\mathcal{ALCHI}$ and Horn-$\mathcal{ALCI}$ KBs, and in* ExpTime *for DL-Lite$^{\mathcal{H}}_{horn}$ and DL-Lite$^{\mathcal{H}}_{core}$ KBs. For data complexity, these problems are all in* P.

Lower Bounds

We have shown that, for all of our DLs, Σ-query entailment and inseparability are in P for data complexity. The next theorem establishes a matching lower bound:

Theorem 24 *For data complexity, Σ-query entailment and inseparability are* P*-hard for DL-Lite$_{core}$ and $\mathcal{EL}$ KBs.*

Proof. The proof is by reduction of the P-complete entailment problem for *acyclic* Horn ternary clauses: given a conjunction φ of clauses of the form a_i and $a_i \wedge a_{i'} \to a_j$, $i, i' < j$, decide whether a_n is true in every model of φ. Consider the $\mathcal{EL}$ TBox $\mathcal{T} = \{V \sqsubseteq \exists P.(\exists R_1.V \sqcap \exists R_2.V)\}$ and an ABox $\mathcal{A}$ comprised of $F(a_n)$ and

$$P(a_i, a_i),\ R_1(a_i, a_i),\ R_2(a_i, a_i), \text{ for each clause } a_i \text{ in } \varphi,$$
$$P(a_j, c), R_1(c, a_i), R_2(c, a_{i'}), \text{ for } c = a_i \wedge a_{i'} \to a_j \text{ in } \varphi.$$

Set $\Sigma = \{F, P, R_1, R_2\}$, $\mathcal{K}_2 = (\mathcal{T}, \mathcal{A} \cup \{V(a_n)\})$ and $\mathcal{K}_1 = (\emptyset, \mathcal{A})$. Obviously, $\mathcal{K}_2$ Σ-query entails $\mathcal{K}_1$. On the other hand, the materialisation of $\mathcal{K}_2$ is (finitely) Σ-homomorphically embeddable in the materialisation of $\mathcal{K}_1$ iff φ derives a_n (see the full version for details). For *DL-Lite$_{core}$*, we take $\mathcal{T}$ to contain $V \sqsubseteq \exists P$, $\exists P^- \sqsubseteq \exists R_i$ and $\exists R_i^- \sqsubseteq V$, for $i = 1, 2$. ❑

For combined complexity, ExpTime-hardness of Σ-query inseparability for *Horn-$\mathcal{ALC}$* can be proved by reduction of the subsumption problem: we have $\mathcal{T} \models A \sqsubseteq B$ iff $(\mathcal{T}, \{A(a)\})$ and $(\mathcal{T} \cup \{A \sqsubseteq B\}, \{A(a)\})$ are $\{B\}$-query inseparable. We now establish matching lower bounds in the technically challenging cases.

Theorem 25 *For combined complexity, Σ-query entailment and inseparability are (i) 2*ExpTime*-hard for Horn-$\mathcal{ALCI}$ KBs and (ii)* ExpTime*-hard for DL-Lite$^{\mathcal{H}}_{core}$ KBs.*

Proof. The proof of (*i*) is by encoding alternating Turing machines (ATMs) with exponential tape and using the fact that AExpSpace = 2ExpTime; see, e.g. (Kozen 2006).

Let $M = (\Gamma, Q, q_0, q_1, \delta)$ be an ATM with a tape alphabet Γ, a set of states Q partitioned into existential $Q_\exists$ and universal $Q_\forall$ states, an initial state $q_0 \in Q_\exists$, an accepting state $q_1 \in Q$, and a transition function

$$\delta \colon (Q \setminus \{q_1\}) \times \Gamma \times \{1, 2\} \to Q \times \Gamma \times \{-1, 0, +1\},$$

which, for a state q and symbol a, gives two instructions, $\delta(q, a, 1)$ and $\delta(q, a, 2)$. We assume that existential and universal states strictly alternate: any transition from an existential state results in a universal state, and vice versa. We extend δ with the instructions $\delta(q_1, a, k) = (q_1, a, 0)$, for $a \in \Gamma$ and $k = 1, 2$, which go into an infinite loop if M reaches the accepting state q_1. Thus, assuming that M terminates on every input, it accepts $\vec{w}$ iff the modified ATM M' has a run on $\vec{w}$, all branches of which are infinite.

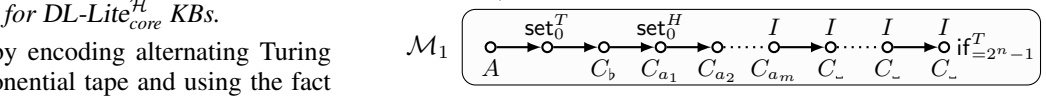
Our aim is to construct, given M and $\vec{w}$, TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ and a signature Σ such that M' has a run with only infinite branches iff the materialisation $\mathcal{M}_2$ of $(\mathcal{T}_2, \{A(c)\})$ is finitely Σ-homomorphically embeddable into the materialisation $\mathcal{M}_1$ of $(\mathcal{T}_1, \{A(c)\})$. Let f be a polynomial such that, on any input of length m, M uses at most $2^n - 2$ tape cells, with $n = f(m)$, which are numbered from 1 to $2^n - 2$, and the head stays to the right of cell 0, which contains the marker $\flat \in \Gamma$. The construction proceeds in five steps.

Step 0. We use tuples of $2n$ concepts to represent distances of up to 2^n between the cells on the tape in consecutive configurations. We refer to a tuple $Y_{n-1}, \overline{Y}_{n-1}, \dots, Y_0, \overline{Y}_0$ of concept names as Y and assume that the TBox contains the following CIs to encode an n-bit *R-counter on Y*:

$$\overline{Y}_k \sqcap Y_{k-1} \sqcap \dots \sqcap Y_0 \sqsubseteq \forall R.(Y_k \sqcap \overline{Y}_{k-1} \sqcap \dots \sqcap \overline{Y}_0), \quad n > k \geq 0,$$
$$\overline{Y}_i \sqcap \overline{Y}_k \sqsubseteq \forall R.\overline{Y}_i \text{ and } Y_i \sqcap \overline{Y}_k \sqsubseteq \forall R.Y_i, \quad n > i > k.$$

We use the expression $\mathsf{if}^Y_{=2^n-1}$ on the left-hand side of CIs to say that the Y-value is $2^n - 1$ (which is a shortcut for $Y_{n-1} \sqcap \dots \sqcap Y_0$); we also use $\mathsf{if}^Y_{<2^n-1}$ on the left-hand side of CIs for the complementary statement (which is a shortcut for n CIs with $\mathsf{if}^Y_{<2^n-1}$ replaced by each of $\overline{Y}_{n-1}, \dots, \overline{Y}_0$). Finally, we use set^Y_0 on the right-hand side of CIs for the reset command (which is equivalent to $\overline{Y}_{n-1} \sqcap \dots \sqcap \overline{Y}_0$). Note that the counter stops at $2^n - 1$: the R-successors of a domain element in $\mathsf{if}^Y_{=2^n-1}$ do not have to encode any value.

Step 1. First we encode configurations and transitions of M' using $\mathcal{T}_1$. We represent a configuration by a *block*, which is a sequence of $2^n + 1$ domain elements connected by a role P. The first element distinguishes the blocks for the two alternative transitions; using a P-counter on a tuple T, we assign indices from 0 to $2^n - 1$ to all other elements in each block. The element with index 0 is needed for padding. Each of the remaining $2^n - 1$ elements belongs to a concept C_a, for some $a \in \Gamma$: if the element with index $i + 1$ is in C_a, then the cell i is assumed to contain a in the configuration represented by the block (in particular, the element with index 1 contains $\flat$ for cell 0) as shown below:



The first block represents the initial configuration: the input $\vec{w} = a_1 \dots a_m$ is followed by $2^n - m - 2$ blank symbols $\sqcup$ and the head is positioned over cell 1, which is indicated by the 0 value of the P-counter on a tuple H. This is achieved by the following CIs in the TBox $\mathcal{T}_1$:

$$A \sqsubseteq \exists P.(\mathsf{set}^T_0 \sqcap \exists P.(C_\flat \sqcap \exists P.(C_{a_1} \sqcap \mathsf{set}^H_0 \sqcap \exists P.(C_{a_2} \sqcap \exists P.(\dots \exists P.(C_{a_m} \sqcap I) \dots))))), \quad (\mathcal{T}_1\text{-1})$$
$$\mathsf{if}^T_{<2^n-1} \sqcap I \sqsubseteq \exists P.(I \sqcap C_\sqcup), \quad (\mathcal{T}_1\text{-2})$$
$$\mathsf{if}^T_{=2^n-1} \sqcap I \sqsubseteq Z^0_{q_0 a_1}. \quad (\mathcal{T}_1\text{-3})$$

Step 2. The contents of the tape and the head position in each configuration is encoded in a block of length $2^n + 1$; the current state $q \in Q$ is recorded in the concept Z^0_{qa} that contains the last element of the block ($a \in \Gamma$ specifies the contents of the active cell scanned by the head). At the end of the block, when the T-value reaches $2^n - 1$, we branch out one block for each of the two transitions, reset the P-counter on T, and propagate via Z^1_{qa} and Z^2_{qa} the current state and symbol in the active cell: for $q \in Q$ and $a \in \Gamma$, we add to $\mathcal{T}_1$ the CI

$$\mathsf{if}^T_{=2^n-1} \sqcap Z^0_{qa} \sqsubseteq \prod_{k=1,2} \exists P.(X_k \sqcap \exists P.(\mathsf{set}^T_0 \sqcap Z^k_{qa})), \quad (\mathcal{T}_1\text{-4})$$

where X_1 and X_2 are two fresh concept names.

The acceptance condition for M' is enforced by means of $\mathcal{T}_2$, which uses a P-counter on a tuple T^0 for a block representing the initial configuration (a T^0-block):

$$A \sqsubseteq \exists P.\mathsf{set}^{T^0}_0, \quad (\mathcal{T}_2\text{-1})$$

$$\mathsf{if}^{T^0}_{<2^n-1} \sqsubseteq \exists P. \quad (\mathcal{T}_2\text{-2})$$

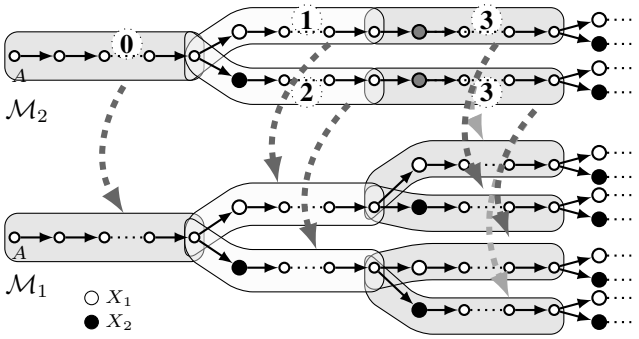
Two P-counters, on T^1 and T^2, are used for blocks representing configurations with universal states (T^1- and T^2-blocks respectively) and one P-counter, on a tuple T^3, suffices for blocks representing configurations with existential states (T^3-blocks). These blocks are arranged into an infinite tree-like structure: the T^0-block is the root, from which a T^1- and a T^2-blocks branch out (successors of the initial state q_0 are universal). Each of them is followed by a T^3-block, which branches out a T^1- and a T^2-block, and so on. This is achieved by adding to $\mathcal{T}_2$ the following CIs:

$$\mathsf{if}^{T^k}_{=2^n-1} \sqsubseteq \prod_{j=1,2} \exists P.(X_j \sqcap \exists P.\mathsf{set}^{T^j}_0), \text{ for } k = 0, 3, \quad (\mathcal{T}_2\text{-3})$$

$$\mathsf{if}^{T^k}_{<2^n-1} \sqsubseteq \exists P.G, \quad \text{for } k = 1, 2, 3, \quad (\mathcal{T}_2\text{-4})$$

$$\mathsf{if}^{T^k}_{=2^n-1} \sqsubseteq \exists P.\exists P.\mathsf{set}^{T^3}_0, \quad \text{for } k = 1, 2, \quad (\mathcal{T}_2\text{-5})$$

where G is a concept name. If $\Sigma = \{A, X_1, X_2, P\}$ then there is a unique Σ-homomorphism from the T^0-block in $\mathcal{M}_2$ to the block of the initial configuration in $\mathcal{M}_1$. Next, concepts X_1 and X_2 ensure that the T^1- and T^2-blocks are Σ-homomorphically mapped (in a unique way) into the respective blocks in $\mathcal{M}_1$, which reflects the acceptance condition of universal states. The following T^3-block, however, contains neither X_1 nor X_2 and can be mapped to either of the blocks in $\mathcal{M}_1$, which reflects the choice in existential states; see the picture below, where possible Σ-homomorphisms are shown by thick dashed arrows:



Step 3. Recall that the P-counter on H measures the distance from the head: if the active cell in the current configuration is k, then its H-value is 0 and the H-value of the cell $k-2$ in a *successor* configuration is $2^n - 1$. So, until the H-counter reaches $2^n - 1$, the following CIs in $\mathcal{T}_1$ propagate the state and symbol in the active cell along the blocks: for $q \in Q$, $a \in \Gamma$ and $k = 0, 1, 2$,

$$\mathsf{if}^T_{<2^n-1} \sqcap \mathsf{if}^H_{<2^n-1} \sqcap Z^k_{qa} \sqsubseteq \prod_{b\in\Gamma} \exists P.(C_b \sqcap Z^k_{qa}) \quad (\mathcal{T}_1\text{-5})$$

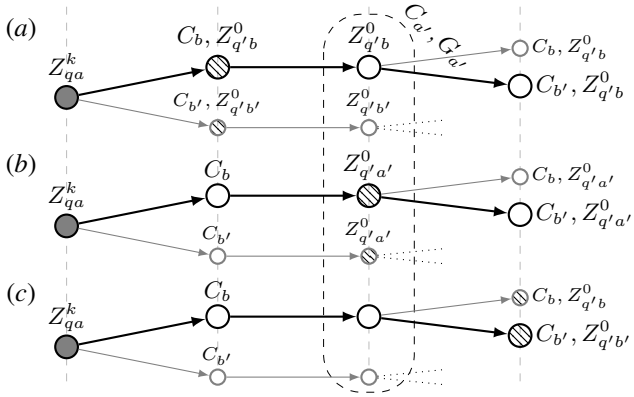
(for each $b \in \Gamma$, these CIs generate a branch in $\mathcal{M}_1$ to represent the same cell but with a different symbol, b, tentatively assigned to the cell—Step 4 will ensure that the correct branch and symbol are selected to match the cell contents in the preceding configuration). When the distance from the last head position is 2^n, the contents of the cell and the current state are changed according to δ:

$$\mathsf{if}^T_{<2^n-1} \sqcap \mathsf{if}^H_{=2^n-1} \sqcap Z^k_{qa} \sqsubseteq \prod_{b\in\Gamma} \exists P.(C_b \sqcap \Delta^k_{qa,b}), \quad (\mathcal{T}_1\text{-6})$$

where $\delta(q, a, k) = (q', a', \sigma)$ and $\Delta^k_{qa,b}$ is the concept

$$\begin{aligned}
&\mathsf{set}^H_0 \sqcap Z^0_{q'b} \sqcap \exists P.(C_{a'} \sqcap G_{a'}), \quad \text{if } \sigma = -1,\\
&\exists P.(C_{a'} \sqcap G_{a'} \sqcap \mathsf{set}^H_0 \sqcap Z^0_{q'a'}), \quad \text{if } \sigma = 0,\\
&\exists P.(C_{a'} \sqcap G_{a'} \sqcap \prod_{b'\in\Gamma} \exists P.(C_{b'} \sqcap \mathsf{set}^H_0 \sqcap Z^0_{q'b'})), \text{ if } \sigma = +1
\end{aligned}$$

(the symbol in the active cell is changed according to the instruction, and the current state and symbol in the next active cell are then recorded in Z^0_{qa}). Since the head never visits cell 0, this happens over cells 0 to $2^n - 1$, that is, at least one element after the P-counter on T is reset to 0. These three situations are shown below, where grey and hatched nodes denote domain elements with H-values $2^n - 1$ and 0, respectively, and the domain elements in the dashed oval represent the active cell of the preceding configuration:



(Note that there is only one branch for the modified cell, which corresponds to the new symbol, a', in that cell; see explanations below.) Then, the current state and the symbol in the active cell are propagated along the tape using ($\mathcal{T}_1$-5).

Step 4. The CIs ($\mathcal{T}_1$-5)–($\mathcal{T}_1$-6) generate a separate P-successor for each $b \in \Gamma$. The correct one is chosen by a

finite Σ-homomorphism, h, from $\mathcal{M}_2$ to $\mathcal{M}_1$. To exclude wrong choices, we take

$$\Sigma \;=\; \{A, P, X_1, X_2\} \;\cup\; \{D_a \mid a \in \Gamma\}.$$

Recall that if $d_1 \in C_a^{\mathcal{M}_1}$, for some $a \in \Gamma$, then it represents a cell containing a. The following CIs in $\mathcal{T}_1$ ensure that, for each $b \in \Gamma$ different from a, there is a block of $(2^n + 1)$-many P^--connected elements that ends in the concept D_b (called a D_b-*block* in the sequel):

$$C_a \sqsubseteq D_a \sqcap \textstyle\prod_{b \in \Gamma \setminus \{a\}} G_b, \qquad (\mathcal{T}_1\text{-}7)$$

$$G_b \sqsubseteq \exists P^-.(S_b \sqcap \mathsf{set}_0^B), \qquad (\mathcal{T}\text{-}1)$$

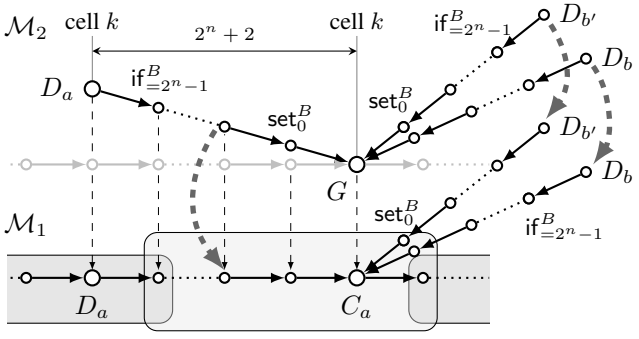
$$\mathsf{if}^B_{<2^n-1} \sqcap S_b \sqsubseteq \exists P^-.S_b, \qquad (\mathcal{T}\text{-}2)$$

$$\mathsf{if}^B_{=2^n-1} \sqcap S_b \sqsubseteq \exists P^-.D_b, \qquad (\mathcal{T}\text{-}3)$$

where we use a P^--counter on a tuple B (unlike P-counters in all other cases) and a concept S_b to propagate b along the whole block. Suppose $h(d_2) = d_1$ and d_2 belongs to G in $\mathcal{M}_2$ (it represents a cell in a non-initial configuration). Then the following CI and ($\mathcal{T}$-1)–($\mathcal{T}$-3), added to $\mathcal{T}_2$, generate a D_b-block, for *each* $b \in \Gamma$ (including a):

$$G \sqsubseteq \textstyle\prod_{b \in \Gamma} G_b. \qquad (\mathcal{T}_2\text{-}6)$$

Each of the D_b-blocks in $\mathcal{M}_2$, for $b \in \Gamma$ with $b \neq a$, can be mapped by h to the respective D_b-block in $\mathcal{M}_1$. By the choice of Σ, the only remaining D_a-block, in case a is tentatively contained in this cell, could be mapped (in the reverse order) along the branch in $\mathcal{M}_1$ *but only* if the cell contains a in the preceding configuration (that is, the element which is $2^n + 1$ steps closer to the root of $\mathcal{M}_1$ belongs to D_a):



Note (see $\Delta^k_{qa,b}$) that the cell whose content is changed generates the additional D_a-block in $\mathcal{M}_1$ to allow the respective D_a-block from $\mathcal{M}_2$ to be mapped there.

One can show that M' has a run with only infinite branches iff $(\mathcal{T}_1, \{A(c)\})$ Σ-query entails $(\mathcal{T}_2, \{A(c)\})$. It follows, by Theorem 2, that deciding Σ-query inseparability is 2ExpTime-hard.

(*ii*) A proof of ExpTime-hardness of Σ-query inseparability for *DL-Lite*$^{\mathcal{H}}_{core}$ KBs is given in the full paper. It uses the same idea of encoding computations of ATMs. One essential difference is that the expressive power of *DL-Lite*$^{\mathcal{H}}_{core}$ is not enough to represent n-bit counters in Step 0, and so we can only encode computations on polynomial tape. ❑

As a consequence of Theorems 3, 23 and 25 we obtain:

Theorem 26 *For combined complexity, the membership problem for universal UCQ-solutions is* 2ExpTime*-complete for Horn-$\mathcal{ALCHI}$ and Horn-$\mathcal{ALCI}$;* ExpTime*-complete for Horn-$\mathcal{ALCH}$, Horn-$\mathcal{ALC}$, DL-Lite*$^{\mathcal{H}}_{horn}$ *and DL-Lite*$^{\mathcal{H}}_{core}$*; and* P*-complete for $\mathcal{EL}$ and $\mathcal{ELH}$. For data complexity, all these problems are* P*-complete.*

In the case of *DL-Lite*$^{\mathcal{H}}_{core}$, we also obtain an ExpTime algorithm for checking the existence and computing universal UCQ-solutions. Indeed, given a KB $\mathcal{K}_1$, a target signature Σ_2 and a mapping $\mathcal{T}_{12}$, we first compute the Σ_2-ABox over $\mathsf{ind}(\mathcal{K}_1)$ that is implied by $\mathcal{K}_1$ and $\mathcal{T}_{12}$, and then check whether at least one KB $\mathcal{K}_2$ in Σ_2 with this ABox is a universal UCQ-solution (there are $\leq O(2^{|\Sigma_2|})$ such KBs). This gives an ExpTime upper bound for the non-emptiness problem for universal UCQ-solutions in *DL-Lite*$^{\mathcal{H}}_{core}$ (Arenas et al. 2013). Similarly, we can check in ExpTime whether the result of forgetting a signature in a *DL-Lite*$^{\mathcal{H}}_{core}$ KB exists.

Σ-query inseparability of *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes was known to sit between PSpace and ExpTime (Konev et al. 2011). Using the fact that witness ABoxes for *DL-Lite*$^{\mathcal{H}}_{core}$ TBox separability can always be chosen among the singleton ABoxes (Konev et al. 2011, Theorem 8), we can modify the proof of Theorem 25 to improve the PSpace lower bound:

Theorem 27 *Σ-query inseparability of DL-Lite*$^{\mathcal{H}}_{core}$ *TBoxes is* ExpTime*-complete.*

For more expressive DLs, TBox Σ-query inseparability is often harder than KB inseparability: for *DL-Lite*$_{horn}$, the space of relevant witness ABoxes for TBox separability is of exponential size and, in fact, TBox inseparability is NP-hard, while KB inseparability is in P. Similarly, Σ-query inseparability of $\mathcal{EL}$ KBs is tractable, while Σ-query inseparability of TBoxes is ExpTime-complete (Lutz and Wolter 2010). The complexity of TBox inseparability for Horn-DLs extending *Horn-$\mathcal{ALC}$* is not known.

Future Work

From a theoretical point of view, it would be of interest to investigate the complexity of Σ-query inseparability for KBs in more expressive Horn DLs (e.g., *Horn-$\mathcal{SHIQ}$*) and non-Horn DLs extending $\mathcal{ALC}$. We conjecture that the game technique developed in this paper can be extended to those DLs as well. Our games can also be used to define *efficient approximations* of Σ-query entailment and inseparability for KBs. The existence of a forward strategy, for example, provides a sufficient condition for Σ-query entailment for all of our DLs. Thus, one can extract a Σ-query module of a given KB $\mathcal{K}$ by exhaustively removing from $\mathcal{K}$ those inclusions and assertions α for which player 1 has a winning strategy in the game $G^f_\Sigma(\mathcal{G}_1, \mathcal{G}_2)$, where $\mathcal{G}_1$ is a generating structure for $\mathcal{K}\setminus\{\alpha\}$ and $\mathcal{G}_2$ for $\mathcal{K}$. The resulting modules are minimal for our DLs without inverse roles, and we conjecture that in practice they are often minimal for DLs with inverse roles as well; see (Konev et al. 2011) for experiments testing similar ideas for module extraction from TBoxes.

Finally, we plan to use the developed technique to investigate the complexity of the non-emptiness problem for universal UCQ-solutions in data exchange as well as algorithms for computing universal UCQ-solutions in various DLs.

Acknowledgements. The paper was supported by the UK EPSRC grants EP/H043594 and EP/H05099X and the EU IP Optique, n. FP7-318338.

References

Arenas, M.; Botoeva, E.; Calvanese, D.; Ryzhikov, V.; and Sherkhonov, E. 2012. Exchanging description logic knowledge bases. In *Proc. of 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*. AAAI Press.

Arenas, M.; Botoeva, E.; Calvanese, D.; and Ryzhikov, V. 2013. Exchanging OWL 2 QL knowledge bases. In *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI)*.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press. (2nd edition, 2007).

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the EL envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, 364–369.

Baier, C., and Katoen, J.-P. 2007. *Principles of Model Checking*. MIT Press.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *Journal of Automated Reasoning* 39(3):385–429.

Cuenca Grau, B.; Horrocks, I.; Kazakov, Y.; and Sattler, U. 2008. Modular reuse of ontologies: theory and practice. *J. of Artificial Intelligence Research (JAIR)* 31:273–318.

Del Vescovo, C.; Parsia, B.; Sattler, U.; and Schneider, T. 2011. The modular structure of an ontology: Atomic decomposition. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011)*, 2232–2237. AAAI Press.

Eiter, T.; Gottlob, G.; Ortiz, M.; and Simkus, M. 2008. Query answering in the description logic Horn-$\mathcal{SHIQ}$. In *Proc. of the 11th European Conf. on Logics in Artificial Intelligence (JELIA 2008)*, volume 5293 of *Lecture Notes in Computer Science*, 166–179. Springer.

Hustadt, U.; Motik, B.; and Sattler, U. 2005. Data complexity of reasoning in very expressive description logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, 466–471.

Jiménez-Ruiz, E.; Cuenca Grau, B.; Horrocks, I.; and Berlanga Llavori, R. 2011. Supporting concurrent ontology development: Framework, algorithms and tool. *Data Knowl. Eng.* 70(1):146–164.

Kazakov, Y. 2009. Consequence-driven reasoning for Horn SHIQ ontologies. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009)*, 2040–2045.

Konev, B.; Kontchakov, R.; Ludwig, M.; Schneider, T.; Wolter, F.; and Zakharyaschev, M. 2011. Conjunctive query inseparability of OWL 2 QL TBoxes. In *Proc. of the 25th AAAI Conf. on Artificial Intelligence*. AAAI Press.

Konev, B.; Ludwig, M.; Walther, D.; and Wolter, F. 2012. The logical difference for the lightweight description logic EL. *Journal of Artificial Intelligence Research (JAIR)* 44:633–708.

Konev, B.; Walther, D.; and Wolter, F. 2009. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009)*, 830–835. AAAI Press.

Kontchakov, R.; Lutz, C.; Toman, D.; Wolter, F.; and Zakharyaschev, M. 2010. The combined approach to query answering in DL-Lite. In *Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2010)*. AAAI Press.

Kontchakov, R.; Wolter, F.; and Zakharyaschev, M. 2010. Logic-based ontology comparison and module extraction, with an application to DL-Lite. *Artificial Intelligence* 174:1093–1141.

Kozen, D. 2006. *Theory of Computation*. Springer.

Krötzsch, M.; Rudolph, S.; and Hitzler, P. 2013. Complexities of horn description logics. *ACM Trans. Comput. Log.* 14(1):2.

Lin, F., and Reiter, R. 1994. Forget it! In *In Proc. of the AAAI Fall Symposium on Relevance*, 154–159.

Lutz, C., and Wolter, F. 2010. Deciding inseparability and conservative extensions in the description logic EL. *Journal of Symbolic Computation* 45(2):194–228.

Lutz, C.; Seylan, I.; and Wolter, F. 2012. An automata-theoretic approach to uniform interpolation and approximation in the description logic EL. In *Proc. of the 13th Int. Conf. on Principles of Knowledge Representation (KR 2012)*. AAAI Press.

Lutz, C.; Toman, D.; and Wolter, F. 2009. Conjunctive query answering in the description logic EL using a relational database system. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI-09)*, 2070–2075.

Mazala, R. 2001. Infinite games. In *Automata, Logics, and Infinite Games*, 23–42.

Nikitina, N., and Glimm, B. 2012. Hitting the sweetspot: Economic rewriting of knowledge bases. In *Proc. of the 11th Int. Semantic Web Conf. (ISWC 2012), Part I*, volume 7649 of *Lecture Notes in Computer Science*, 394–409. Springer.

Nikitina, N., and Rudolph, S. 2012. ExpExpExplosion: Uniform interpolation in general EL terminologies. In *Proc. of the 20th European Conf. on Artificial Intelligence (ECAI 2012)*, 618–623. IOS Press.

Rosati, R. 2007. On conjunctive query answering in EL. In *Proc. of the 2007 Int. Workshop on Description Logics (DL)*.

Stuckenschmidt, H.; Parent, C.; and Spaccapietra, S., eds. 2009. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*. Springer.

Wang, Z.; Wang, K.; Topor, R. W.; and Pan, J. Z. 2010. Forgetting for knowledge bases in DL-Lite. *Ann. Math. Artif. Intell.* 58(1-2):117–151.

Wang, Z.; Wang, K.; and Topor, R. W. 2010. Revising general knowledge bases in description logics. In *Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2010)*. AAAI Press.

Answering Instance Queries Relaxed by Concept Similarity

Andreas Ecke
Theoretical Computer Science,
TU Dresden, Germany

Rafael Peñaloza
Theoretical Computer Science,
TU Dresden, Germany
Center for Advancing Electronics Dresden

Anni-Yasmin Turhan
Theoretical Computer Science,
TU Dresden, Germany

{ecke,penaloza,turhan}@tcs.inf.tu-dresden.de

Abstract

In Description Logic (DL) knowledge bases (KBs) information is typically captured by crisp concepts. For many applications, querying the KB by crisp query concepts is too restrictive. A controlled way of gradually relaxing a query concept can be achieved by the use of concept similarity measures. In this paper we formalize the task of instance query answering for crisp DL KBs using concepts relaxed by concept similarity measures. We investigate computation algorithms for this task in the DL $\mathcal{EL}$, their complexity and properties for the employed similarity measure regarding whether unfoldable or general TBoxes are used.

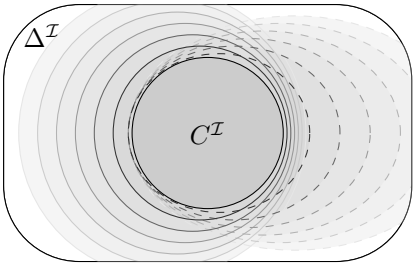


Figure 1: Relaxed instances w.r.t. two different similarity measures. Darker colors represent the relaxed instances of C w.r.t. higher degrees t.

1 Introduction

Description Logic (DL) knowledge bases are formal vocabularies that describe categories or specific subjects from application domains—such as for service matching, the biomedical or geo-spatial field. The concepts in the knowledge base are characterized by relationships to other concepts using constructors available in the DL in which the knowledge base is formulated.

The use of DLs became increasingly popular in recent years, due to the W3C standard OWL 2 for ontology languages which are based on DLs, and the development of powerful DL reasoner systems for those languages. These systems support various reasoning services, such as subsumption and instance queries. *Subsumption* determines for a pair of concepts whether elements belonging to the first concept necessarily also belong to the second one. Given a concept, *instance queries* retrieve from a knowledge base all those individuals that belong to the concept. For some applications exact matches to an instance query are sometimes too restrictive. In service matching, OWL TBoxes are employed to describe types of services and a user request for a service specifies several conditions for the desired service by a concept. For such a query concept instance query answering is performed over the OWL knowledge base that contains the individual services. If an exact match with the provided requirements is not available, a 'feasible' alternative needs to be retrieved. More precisely, the system should retrieve those individuals from the knowledge base that satisfy the main conditions, while all other conditions can be relaxed. A natural idea on how to relax the notion of instance query answering is to simply employ fuzzy DLs and perform query answering on a fuzzy variant of the initial query concept. However, on the one hand reasoning in fuzzy DLs easily becomes undecidable (Borgwardt and Peñaloza 2012) and on the other hand depending on the user and on the request, different ways of relaxing the query concept may be appropriate.

In this paper we investigate a form of answering relaxed instance queries, where the query concept can be relaxed by using concept similarity measures. A *concept similarity measure* (CSM) is a function from pairs of concepts to the unit interval, where the value 1 indicates total similarity and the value 0 total dissimilarity of the concepts. Answering *relaxed instance queries* is to compute, given concept C, a concept similarity measure $\sim$, and a degree $t \in [0,1]$, the set of individuals that instances of concepts similar to C by a degree of at least t, if measured by $\sim$. This approach, recently proposed in (Ecke, Peñaloza, and Turhan 2013), does not need to extend the DL in which the knowledge base is written, thus the complexity for standard reasoning tasks remain the same. The choice of a CSM allows to encode the application-specific notion of similarity. Furthermore the choice of t allows for a flexible degree of similarity, as depicted in Figure 1.

For DLs there is whole range of CSMs defined (see for example (Borgida, Walsh, and Hirsh 2005; d'Amato, Fanizzi, and Esposito 2005; Janowicz and Wilkes 2009;

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Lehmann and Turhan 2012)), which could be employed for this task. Computing concept similarity is an ontology service that is investigated in its own right. For instance, for the Gene Ontology (Gene Ontology Consortium 2000), which is written in the DL $\mathcal{EL}$ and is used, among others, to solve the task of finding genes that realize similar functionality (Lord et al. 2003), a proliferation of different similarity measures has been defined (Lord et al. 2003; Schlicker et al. 2006; Mistry and Pavlidis 2008; Alvarez and Yan 2011).

Since concept similarity is not a formalized notion, CSMs are often defined in an ad-hoc manner or simply tuned to test data. This makes their behavior hard to predict when applied to new ontologies. In (Lehmann and Turhan 2012) a set of properties for CSMs was described and the framework given there allows users to generate CSMs for $\mathcal{EL}$-concepts with those properties. Furthermore, users can specify which parts of the vocabulary used in their knowledge base are important in regard of assessing similarity of concepts. These measures naturally allow to select which aspect of a query concept to relax.

We devise a computation algorithm for answering relaxed instance queries for concepts defined w.r.t. unfoldable $\mathcal{EL}$ TBoxes and any CSM that meets certain requirements. The core reasoning problem in our algorithm is to compute, for an individual a and the query concept C, a concept C' that *mimics* C, i.e. a concept that is 'sufficiently similar' to C w.r.t. the used similarity measure $\sim$ and the degree t, while preserving a as an instance.

To the best of our knowledge there is no CSM defined in the literature that takes *all* of the information from *general* $\mathcal{EL}$-TBoxes into account. Existing CSMs for general TBoxes consider only the concept hierarchy (e.g. (d'Amato, Staab, and Fanizzi 2008; Alvarez and Yan 2011)). In this paper we devise a family of CSMs that handle all the knowledge encoded in general $\mathcal{EL}$-TBoxes. We derive such CSMs by employing the canonical models of TBoxes and applying a similarity measure for interpretations. We show that this interpretation similarity measure (ISM) has properties analogous to those from (Lehmann and Turhan 2012). We define corresponding CSMs, which also preserve these properties. These CSMs for general TBoxes are the foundation to lift the computation algorithm for answering relaxed instance queries to the case of general $\mathcal{EL}$-TBoxes. It shows that the stronger CMSs obtained for this case allow for computation of relaxed instances in polynomial time, if applied to unfoldable TBoxes.

This paper is structured as follows. The next section recalls some basic notions for DLs, in particular $\mathcal{EL}$, and CSMs. Section 3 introduces relaxed instance query answering and our approach to compute it by the auxiliary task of computing a mimic. Section 4 presents a computation algorithm for relaxed instances w.r.t. unfoldable $\mathcal{EL}$-TBoxes and gives an upper bound on the complexity. Afterwards, we introduce a family of similarity measures on pointed canonical interpretations $\sim_i$ which have well-defined formal properties. From this measure we derive a CSM $\sim_c$ with the same set of properties which can be used to compute relaxed instances w.r.t. general $\mathcal{EL}$-TBoxes. We give a computation algorithm in that section and end the paper with directions for future work. Due to space constraints the proofs can be found in the technical report (Ecke and Turhan 2013).

Table 1: Concept constructors, TBox axioms and ABox assertions for $\mathcal{EL}$.

	Syntax	Semantics
top concept	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

2 Preliminaries

As we want to investigate the notion of similarity in the context of the DL $\mathcal{EL}$, we briefly introduce this logic and CSMs.

The Description Logic $\mathcal{EL}$

Starting from the fixed, countably infinite, disjoint sets of concept names N_C and role names N_R, $\mathcal{EL}$*-concepts* can be constructed by the following syntactic rule

$$C, D ::= \top \mid A \mid C \sqcap D \mid \exists r.D,$$

where A is a concept name and r is a role name. The set of all $\mathcal{EL}$-concepts is denoted by $\mathfrak{C}(\mathcal{EL})$.

The semantics of concepts is defined by means of *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names and subsets of $\Delta^{\mathcal{I}}$ to concept names. The interpretation function is recursively extended to $\mathcal{EL}$-concepts as shown in the upper part of Table 1. We denote the class of all interpretations as $\mathfrak{I}$. A *pointed interpretation* $p = (\mathcal{I}, d)$ consists of an interpretation $\mathcal{I}$, and an element $d \in \Delta^{\mathcal{I}}$. $\mathfrak{P}$ is the class of all pointed interpretations, i.e., $\mathfrak{P} = \{(\mathcal{I}, d) \mid \mathcal{I} \in \mathfrak{I}, d \in \Delta^{\mathcal{I}}\}$. Given a pointed interpretation $p = (\mathcal{I}, d)$, the set of all $\mathcal{EL}$-concepts that have d as an instance in $\mathcal{I}$ is $\mathfrak{C}(p) = \{C \in \mathfrak{C}(\mathcal{EL}) \mid d \in C^{\mathcal{I}}\}$.

The middle part of Table 1 displays two kinds of concept axioms. An *$\mathcal{EL}$-TBox* or *terminology* $\mathcal{T}$ is a set of such concept axioms. An *unfoldable TBox*, is a set of concept definitions such that each concept name occurs at most once on the left-hand side of a concept definition and there are no cyclic dependencies between defined concepts. Concept names occurring on the left-hand side of a concept definition are called *defined concepts*, the other concept names are *atomic concepts*.

Consider an additional set N_I of *individual names*, which is disjoint with N_C and N_R. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ additionally maps each individual name to an element of

$\Delta^{\mathcal{I}}$. An ABox $\mathcal{A}$ is a set of concept or role assertions, as displayed in the lower part of Table 1. An *$\mathcal{EL}$-knowledge base* (KB) is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is an *$\mathcal{EL}$-TBox* and $\mathcal{A}$ an *$\mathcal{EL}$-ABox*.

The semantics of interpretations is extended to TBox axioms and ABox assertions as shown in Table 1. An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ (ABox $\mathcal{A}$), if it satisfies all axioms in $\mathcal{T}$ (assertions in $\mathcal{A}$). $\mathcal{I}$ is a model of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it is a model for both $\mathcal{T}$ and $\mathcal{A}$.

The following commonly used reasoning tasks are implemented in most DL reasoning systems. *Concept subsumption* asks, given a TBox $\mathcal{T}$ and two concepts C and D, whether C is subsumed by D w.r.t. $\mathcal{T}$ (denoted $C \sqsubseteq_{\mathcal{T}} D$), i.e. $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. *Concept equivalence* holds for two concepts C and D w.r.t. $\mathcal{T}$ (denoted $C \equiv_{\mathcal{T}} D$), iff $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$. Given an individual a, a concept C, and a KB $\mathcal{K}$, a is called an *instance of* C w.r.t. $\mathcal{K}$, denoted $\mathcal{K} \models C(a)$, iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$. Given a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a concept C, *instance retrieval* returns all individuals from $\mathcal{A}$ that are instances of C. These reasoning tasks can all be characterized by means of simulations between interpretations.

Definition 1 (simulation). Let $\mathcal{I}$ and $\mathcal{J}$ be interpretations. A relation $S \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ is a *simulation* between $\mathcal{I}$ and $\mathcal{J}$, if the following two conditions hold:

1. For all $(d, e) \in S$ and $A \in N_C$, if $d \in A^{\mathcal{I}}$ then $e \in A^{\mathcal{J}}$.
2. For all $(d, e) \in S$, $r \in N_R$ and $(d, d') \in r^{\mathcal{I}}$, there is an $(e, e') \in r^{\mathcal{J}}$ with $(d', e') \in S$.

The pointed interpretation $p = (\mathcal{I}, d)$ *simulates* the pointed interpretation $q = (\mathcal{J}, e)$ ($p \lesssim q$), if there exists a simulation $S \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ between $\mathcal{I}$ and $\mathcal{J}$ with $(d, e) \in S$. If $p \lesssim q$ and $q \lesssim p$, then p and q are *equisimilar* ($p \simeq q$). There is a strong connection between simulations between pointed interpretations and their concept sets.

Theorem 2 ((Lutz and Wolter 2010)). *Let $p, q \in \mathfrak{P}$. Then:*
1. $p \lesssim q$ iff $\mathfrak{C}(p) \subseteq \mathfrak{C}(q)$, and 2. $p \simeq q$ iff $\mathfrak{C}(p) = \mathfrak{C}(q)$.

For $\mathcal{EL}$ most reasoning procedures rely on the fact that canonical models can be built, from which entailments can be read-off directly. By $\mathrm{Sig}(X)$ we denote the set of names and by $\mathsf{sub}(X)$ the sub-concepts appearing in X.

Definition 3. Let $C \in \mathfrak{C}(\mathcal{EL})$ and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-KB. The *canonical model $\mathcal{I}_{C,\mathcal{T}} = (\Delta^{\mathcal{I}_{C,\mathcal{T}}}, \cdot^{\mathcal{I}_{C,\mathcal{T}}})$ of C* w.r.t. the TBox $\mathcal{T}$ is defined as follows:

- $\Delta^{\mathcal{I}_{C,\mathcal{T}}} = \{d_C\} \cup \{d_D \mid \exists r.D \in \mathsf{sub}(C) \cup \mathsf{sub}(\mathcal{T})\}$
- $A^{\mathcal{I}_{C,\mathcal{T}}} = \{d_D \mid D \sqsubseteq_{\mathcal{T}} A\}$, and
- $r^{\mathcal{I}_{C,\mathcal{T}}} = \{(d_D, d_E) \mid D \sqsubseteq_{\mathcal{T}} \exists r.E\}$.

The *canonical model $\mathcal{I}_{\mathcal{K}} = (\Delta^{\mathcal{I}_{\mathcal{K}}}, \cdot^{\mathcal{I}_{\mathcal{K}}})$ of the KB $\mathcal{K}$* is defined as follows:

- $\Delta^{\mathcal{I}_{\mathcal{K}}} = \{d_a \mid a \in \mathrm{Sig}(\mathcal{A}) \cap N_I\} \cup \{d_C \mid \exists r.C \in \mathsf{sub}(\mathcal{A}) \cup \mathsf{sub}(\mathcal{T})\}$,
- $A^{\mathcal{I}_{\mathcal{K}}} = \{d_D \mid D \sqsubseteq_{\mathcal{T}} A\} \cup \{d_a \mid \mathcal{K} \models A(a)$,
- $r^{\mathcal{I}_{\mathcal{K}}} = \{(d_D, d_E) \mid D \sqsubseteq_{\mathcal{T}} \exists r.E\} \cup \{(d_a, d_D) \mid \mathcal{K} \models \exists r.D(a)\} \cup \{(d_a, d_b) \mid r(a, b) \in \mathcal{A}\}$.

Note that canonical models for $\mathcal{EL}$ are always finite. The canonical model $\mathcal{I}_{C,\mathcal{T}}$ can be seen as the most general model for C and $\mathcal{T}$, as any other model $\mathcal{J}$ of $\mathcal{T}$ with a $d \in C^{\mathcal{J}}$ can be simulated by $(\mathcal{I}_{C,\mathcal{T}}, d_C)$. Similarly, for any other model $\mathcal{J}$ of $\mathcal{K}$ with $d = a^{\mathcal{J}}$ for an individual a, $(\mathcal{J}, d)$ is simulated by $(\mathcal{I}_{\mathcal{K}}, d_a)$.

Theorem 4 ((Lutz and Wolter 2010)). *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox, C and D be $\mathcal{EL}$-concepts. Then:*

1. *for all models $\mathcal{I}$ of $\mathcal{T}$ and all elements $d \in \Delta^{\mathcal{I}}$ it is the case that $d \in C^{\mathcal{I}}$ iff $(\mathcal{I}_{C,\mathcal{T}}, d_C) \lesssim (\mathcal{I}, d)$; and*
2. *$C \sqsubseteq_{\mathcal{T}} D$ iff $d_C \in D^{\mathcal{I}_{C,\mathcal{T}}}$ (i.e., $D \in \mathfrak{C}((\mathcal{I}_{C,\mathcal{T}}, d_C))$) iff $(\mathcal{I}_{D,\mathcal{T}}, d_D) \lesssim (\mathcal{I}_{C,\mathcal{T}}, d_C)$.*

When testing whether an individual is a relaxed instance of some concept C, we will see the need to compute 'the best-fitting $\mathcal{EL}$-concept description' that has the individual as an instance. This can be realized by the task of computing the most specific concept.

Definition 5 (most specific concept). Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a KB and a an individual from $\mathcal{A}$. A concept C is the *most specific concept (msc) of a w.r.t. $\mathcal{K}$* (denoted $\mathrm{msc}_{\mathcal{K}}(a)$) if it satisfies:

1. $\mathcal{K} \models C(a)$, and
2. $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_{\mathcal{T}} D$ for all concepts D.

Since the msc may not exists due to an infinite nesting of existential restrictions, one can compute an approximation by bounding the depth of the quantifiers. Formally, given a concept C, we define the role-depth $\mathrm{rd}(C)$ of C by

$$\mathrm{rd}(C) := \begin{cases} 0 & \text{if } C \in N_C \cup \{\top\} \\ 1 + \mathrm{rd}(D) & \text{if } C = \exists r.D \\ max\{\mathrm{rd}(C_1), \mathrm{rd}(C_2)\} & \text{if } C = C_1 \sqcap C_2 \end{cases}$$

If in Definition 5 C and D have a role-depth limited to $k \in \mathbb{N}$, then C is the *role-depth bounded msc* of a w.r.t. $\mathcal{K}$ (k-$\mathrm{msc}_{\mathcal{K}}(a)$). The msc and the k-msc are unique up to equivalence in $\mathcal{EL}$, if they exist (Peñaloza and Turhan 2011).

Concept Similarity Measures

Given a DL $\mathcal{L}$ and a TBox $\mathcal{T}$, a concept similarity measure (CSM) is a function $\sim_{\mathcal{T}} : \mathfrak{C}(\mathcal{L}) \times \mathfrak{C}(\mathcal{L}) \to [0, 1]$ such that $C \sim_{\mathcal{T}} C = 1$ for all concepts C. Intuitively, $\sim_{\mathcal{T}}$ expresses how close two concepts are. A value $C \sim_{\mathcal{T}} D = 0$ means that the concepts C and D are totally dissimilar w.r.t. $\mathcal{T}$, while a value of 1 indicates total similarity. We often simply write $\sim$ instead of $\sim_{\mathcal{T}}$ if the TBox $\mathcal{T}$ is clear from the context.

In (Lehmann and Turhan 2012) a set of properties for CSMs is defined and a framework is devised that allows to construct CSMs for $\mathcal{EL}$-concepts, w.r.t. unfoldable TBoxes, that have these formal properties. In this paper we investigate CSMs for $\mathcal{EL}$-concepts defined w.r.t. general TBoxes. We extend the definition of the properties of CSMs from (Lehmann and Turhan 2012) to the case of general TBoxes.

Definition 6. A CSM $\sim : \mathfrak{C}(\mathcal{EL}) \times \mathfrak{C}(\mathcal{EL}) \to [0, 1]$ is:

symmetric iff $C \sim_{\mathcal{T}} D = D \sim_{\mathcal{T}} C$;

equivalence invariant iff for all $C \equiv_\mathcal{T} D$ it holds that $C \sim_\mathcal{T} E = D \sim_\mathcal{T} E$;

equivalence closed iff $C \equiv_\mathcal{T} D \iff C \sim_\mathcal{T} D = 1$;

bounded iff the existence of $E \neq \top$ with $C \sqsubseteq_\mathcal{T} E$ and $D \sqsubseteq_\mathcal{T} E$ implies $C \sim_\mathcal{T} D > 0$;

dissimilar closed iff $C, D \neq \top$ and there is no $E \neq \top$ with $C \sqsubseteq_\mathcal{T} E$ and $D \sqsubseteq_\mathcal{T} E$ implies $C \sim_\mathcal{T} D = 0$;

subsumption preserving iff $C \sqsubseteq_\mathcal{T} D \sqsubseteq_\mathcal{T} E$ implies $C \sim_\mathcal{T} D \geq C \sim_\mathcal{T} E$;

reverse subsumption preserving iff $C \sqsubseteq_\mathcal{T} D \sqsubseteq_\mathcal{T} E$ implies $D \sim_\mathcal{T} E \geq C \sim_\mathcal{T} E$, and

These formally defined properties make CSMs more predictable for users. The measures in (Suntisrivaraporn 2013; Lehmann and Turhan 2012) fulfill most of these properties, in particular, they are symmetric and equivalence invariant, two properties that we will require later to compute relaxed instances. The parameterizable similarity measures from (Lehmann and Turhan 2012) additionally allow users to calibrate the measure to fit their expectations. In our setting these parametrizable CSMs enable users to specify which features of query concepts should be relaxed.

3 Relaxed Instances

As discussed before, our goal is to generalize query answering to allow for relaxing the solution set. This means, when a query concept Q is given, we are not only interested in the instances of Q, but also those individuals that are close enough to being instances; those individuals are called *relaxed instances*.

There are many ways to formalized the notion of relaxed instances, but in this paper we make use of CSMs for this relaxation. We say that relaxed instances of the query concept are those individuals that are instance of a (possibly different) concept that is sufficiently similar to the original query concept. The condition of 'sufficiently similar' is expressed via a threshold, or minimal similarity, t. One can then control how inclusive the relaxed instance solutions should be, by adjusting t, and guide the direction, in which solutions are relaxed by choosing an appropriate CSM.

Definition 7 (relaxed instance). Let $\mathcal{L}$ be some DL, $\sim$ be a CSM, and $t \in [0, 1)$. The individual $a \in N_I$ is a *relaxed instance* of the query concept Q w.r.t. the $\mathcal{L}$-knowledge base $\mathcal{K}$, $\sim$ and the threshold t iff there exists a $\mathcal{L}$-concept description $X \in \mathfrak{C}(\mathcal{L})$ such that $Q \sim X > t$ and $\mathcal{K} \models X(a)$. $\text{Relax}_t^\sim(Q)$ denotes the set of all individuals occurring in $\mathcal{K}$ that are relaxed instances of Q w.r.t. $\mathcal{K}$, $\sim$ and t.

The task we are mostly interested in is not *relaxed instance checking*, i.e. testing whether a given individual a is a relaxed instance of the query concept Q, but to compute all relaxed instances of Q, i.e. the set $\text{Relax}_t^\sim(Q)$.

One naive idea to compute this set would be to first computing all concepts X that are similar to C with degree greater than t, and then obtaining all the instances of these concepts X; in symbols,

$$\text{Relax}_t^\sim(C) = \bigcup_{C \sim X > t} \{a \mid a \text{ is an instance of } X\}.$$

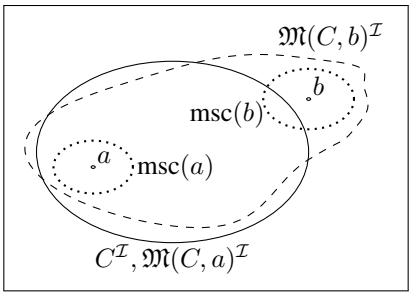


Figure 2: Two individuals, their most specific concepts (dotted), and the mimics of a concept C w.r.t. the individuals (dashed).

However, this approach is not feasible, since there might be an infinite number of concepts X sufficiently similar to C. Furthermore, given an CSM, it is not apparent how to even obtain all concepts that are similar to C with degree $> t$, since in general CSMs only allow to evaluate the similarity of two given concepts.

To avoid these issues, we consider a different reasoning problem, that considers the computation of a concept that has a given individual a as an instance and resembles C best w.r.t. the given CSM. We call this the *mimic* of C w.r.t. a.

Definition 8 (mimic). Let $\mathcal{L}$ be a DL, $\mathcal{K}$ be an $\mathcal{L}$-knowledge base, $a \in N_I$ be an individual name, C be an $\mathcal{L}$-concept description, and $\sim$ be a CSM. An $\mathcal{L}$-concept D is called a *mimic* of C w.r.t. a, denoted $D \in \mathfrak{M}(C, a)$, iff the following two conditions hold:

- a is an instance of D, i.e., $a^\mathcal{I} \in D^\mathcal{I}$ for all models $\mathcal{I}$ of $\mathcal{K}$, and
- for all $\mathcal{L}$-concept descriptions E holds, if a is an instance of E, then $C \sim D \geq C \sim E$.

Intuitively, a mimic of C w.r.t. a is the most similar concept to C that has a as an instance. In general, the mimic of C w.r.t. an individual a may not be unique, even modulo concept equivalence, however, for our purposes it suffices to compute only one of them.

Figure 2 depicts the idea of mimics. In the figure, a and b are two different individuals. Since a is an instance of C, C is also a mimic of C w.r.t. a because $C \sim C = 1$. The dashed line depicts a mimic of C w.r.t. b. Notice that this mimic must contain b and thus subsume $\text{msc}(b)$, but need not be a subsumer of C, it just tries to resemble C in a way that results in maximal similarity.

There is an easy reduction from the computation of relaxed instances to the computation of mimics. The idea is that, for each individual a appearing in the knowledge base $\mathcal{K}$, the mimic of Q w.r.t. a must have a similarity greater than t to the query concept Q for a to be a relaxed instance of Q; if not, a cannot be a relaxed instance, as no concept can have a greater similarity to Q while containing a. This is formalized in the following proposition.

Proposition 9. *Let $\mathcal{K}$ be a knowledge base, a be an individual occurring in $\mathcal{K}$, Q be a concept description, $\sim$ be a*

CSM and $t \in [0,1]$. Then $a \in \text{Relax}_t^{\sim}(Q)$ iff there is a mimic $D \in \mathfrak{M}(Q,a)$ of Q w.r.t. individual a such that $Q \sim D > t$.

This connection allows to study the problem of computing mimics in order to find relaxed instances. In the next section, we will show how this can be solved for CSMs on unfoldable terminologies.

4 Computing Relaxed Instances w.r.t. Unfoldable $\mathcal{EL}$-Terminologies

Structural concept similarity measures with nice properties are often only defined for unfoldable TBoxes, as in (Lehmann and Turhan 2012) and (Suntisrivaraporn 2013). The reason is that in those terminologies, any concept can be expanded by exhaustively replacing all defined concept names by their definitions, until only atomic concept names remain; the concept is then called *fully expanded*. After expanding all concepts, the TBox can be completely disregarded for computing similarities or other inferences like subsumption. In this section, we show how to compute mimics, and therefore relaxed instances w.r.t. unfoldable $\mathcal{EL}$-terminologies.

In general there may exist infinitely many concept descriptions which have the individual a as an instance, and thus enumerating them and finding the maximal similarity to the query concept Q to compute the mimic is not a feasible option. However, under some circumstances we can limit the number of concepts that need to be tested.

Recall that any mimic $D \in \mathfrak{M}(Q,a)$ must have a as an instance, and thus, by definition of the msc, $\text{msc}(a) \sqsubseteq_{\mathcal{T}} D$ holds. For equivalence invariant similarity measures one can use the $\text{msc}(a)$ as a lower bound for the mimic, and only consider concept descriptions that can be obtained from syntactic manipulations of $\text{msc}(a)$ that result in a generalized concept, i.e., by removing some concept names or existential restrictions.

Definition 10 (generalized concept). Let C be a concept description of the form

$$C = \textstyle\bigsqcap_{i\in I} A_i \sqcap \bigsqcap_{j\in J} \exists r_j.E_j,$$

with $A_i \in N_C$ for all $i \in I$, and $r_j \in N_R$, E_j is a concept description for all $j \in J$. Then a concept description D is a *generalized concept* of C iff it has the form

$$D = \textstyle\bigsqcap_{i\in I'} A_i \sqcap \bigsqcap_{j\in J'} \exists r_j.E'_j$$

with $I' \subseteq I$, $J' \subseteq J$ and E'_j is a generalized concept of E_j for $j \in J'$.

Clearly, if D is a generalized concept of C, then we always have $C \sqsubseteq D$, since generalizing only removes parts of C, but never adds new features. Indeed, in our setting the other direction holds as well for unfoldable TBoxes $\mathcal{T}$: If $\text{msc}(a)$ is fully expanded w.r.t. $\mathcal{T}$, then any concept description D with $\text{msc}(a) \sqsubseteq D$ is equivalent to a generalized concept of $\text{msc}(a)$. If we now restrict to equivalence invariant CSMs, we get the following result for the mimic.

Lemma 11. *Let $\mathcal{K} = (\mathcal{T},\mathcal{A})$ be an unfoldable $\mathcal{EL}$-KB, a be an individual from $\mathcal{A}$, C be an $\mathcal{EL}$-concept, and $\sim$ be an equivalence invariant similarity measure. If $E = \text{msc}(a)$ is the fully expanded most specific concept of a, then there is a mimic $D \in \mathfrak{M}(C,a)$ of C w.r.t. a and $\mathcal{K}$ that is a generalized concept of E.*

Proof sketch. We show that any concept having a as an instance is equivalent to a generalized concept of $\text{msc}(a)$. Let F be a concept description with $\mathcal{K} \models F(a)$. Since $\sim$ is equivalence invariant, we can assume F to be fully expanded as well. $E \sqsubseteq_{\mathcal{K}} F$ holds by definition of the msc. As both E and F are fully expanded and thus do not contain defined concept names from the TBox, any subconcept of F must also be a subconcept E, but F may contain redundancies that can simply be removed to get an equivalent concept. Thus F is equivalent to a generalized concept of E. □

However, in general the msc may result in infinitely nested existential restrictions, and thus could only be described by concept descriptions of infinite size. In this case there are infinitely many generalized concepts of finite size that would need to be checked to find a mimic. Lemma 11 therefore does not always provide a solution to the problem. However, the query concept Q has always a finite role-depth, even if fully expanded. Since most structural similarity measures used in practice, like those presented in (Suntisrivaraporn 2013; Lehmann and Turhan 2012), compute the similarity recursively between sub-concepts at the same role-depth, it is possible to also limit the role-depth of the most specific concept and still get the same result:

Lemma 12. *Let $\mathcal{K} = (\mathcal{T},\mathcal{A})$ be an unfoldable $\mathcal{EL}$-KB, a be an individual from $\mathcal{A}$, Q be a fully expanded $\mathcal{EL}$-concept, and $\sim$ be a equivalence invariant similarity measure, which is additionally structural and depth-bounded, i.e.:*

$$C \sim D \geq C \sim E \Rightarrow \exists r.C \sim \exists r.D \geq \exists r.C \sim \exists r.E, \quad (1)$$

$$C \sim \textstyle\bigsqcap_{i\in I} A_i \;\geq\; C \sqcap \exists r.D \sim \bigsqcap_{i\in I} A_i, \quad (2)$$

for all fully expanded concepts C, D, and E, and atomic concept names A_i. If $k = \text{rd}(Q)$ and $E = k\text{-msc}(a)$ is the fully expanded most specific concept of a bounded to the role-depth k, then there is a mimic $D = \mathfrak{M}(Q,a)$ of Q w.r.t. a that is a generalized concept of E.

Proof sketch. Using the properties (1) and (2) above, we can show the following: Trimming the role-depth of two concepts C and D to the minimal role-depth $\min(\text{rd}(C),\text{rd}(D))$ never reduces the similarity value, i.e., the similarity value between the trimmed concepts must be greater than or equal to the similarity value of C and D.

Using this, we can prove the lemma as follows: Let M be any mimic of Q w.r.t. a. Since the mimic is always equivalent to a generalized concept M' of the (possibly infinite) $\text{msc}(a)$, and since $\sim$ is equivalence invariant, M' is also a mimic. Using the previous property, we know that M' trimmed to role-depth k must still have (at least) the same similarity to Q, and hence also be a mimic. But M' trimmed to role-depth k is also a generalized concept of the $k\text{-msc}(a)$. □

This shows that we can always find the mimic of Q w.r.t. a from a finite set of concept descriptions: the generalized

Procedure: relaxed-instance? $(a, Q, \mathcal{K}, \sim, t)$
Input: a: individual in $\mathcal{K}$; Q: $\mathcal{EL}$-concept; $\mathcal{K}$: unfoldable $\mathcal{EL}$-KB; $\sim$: CSM; $t \in (0, 1]$: threshold
Output: whether $a \in \text{Relax}_t^{\sim}(Q)$ w.r.t. $\mathcal{K}$

1: $k := \text{rd}(Q)$
2: $E := k\text{-msc}(a)$ w.r.t. $\mathcal{K}$
3: guess a generalized concept F of E
4: **return** $F \sim Q > t$

Figure 3: Non-deterministic algorithm for relaxed instances checking w.r.t. unfoldable $\mathcal{EL}$ terminologies.

concepts of the fully expanded k-msc(a). However, instead of computing the mimic $D \in \mathfrak{M}(Q, a)$ and testing whether the similarity between the Q and D is greater than t, it is enough to find *any* concept D' with a as an instance and $Q \sim D' > t$ to show that a is a relaxed instance of Q; this gives rise to the non-deterministic algorithm in Figure 3, that checks whether a is a relaxed instance of Q.

Corollary 13. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an unfoldable $\mathcal{EL}$-KB, Q be a fully expanded $\mathcal{EL}$-concept, a be an individual in $\mathcal{K}$, $\sim$ be an equivalence invariant similarity measure fulfilling properties 1 and 2 from Lemma 12 and $t \in [0, 1]$. Then* relaxed-instance?$(a, Q, \mathcal{K}, \sim, t)$ *computes whether $a \in \text{Relax}_t^{\sim}(Q)$ w.r.t. $\mathcal{K}$.*

Guessing a generalized concept F of a concept description E can be done in time linear on the size $\|E\|$ of E by recursively guessing for each concept name and each existential restriction in E whether they should occur in F or not. However, the size of $E = k$-msc(a) can be exponential in k and polynomial in $\|\mathcal{K}\|$ (Peñaloza and Turhan 2011). Since $k = \text{rd}(C)$ is bounded linearly by $\|C\|$, the algorithm runs in NEXP-time (provided that $\sim$ can be computed in NEXP-time). However, the algorithm runs in NP-time in $\|\mathcal{K}\|$ (provided that $\sim$ can be computed in NP), and since C is an input concept, its role-depth can be assumed to be rather low. Hence, we conjecture that the exponential blow-up of the msc usually plays only a minor role in practical applications.

5 Relaxed Instances w.r.t. General $\mathcal{EL}$-TBoxes

The approach presented in the previous section does not work for general $\mathcal{EL}$-TBoxes since those structural similarity measures first fully expand the concepts w.r.t. the background knowledge. As soon as the background knowledge contains cyclic concept inclusions, this expansion is no longer possible. In fact, to the best of our knowledge no equivalence invariant, structural concept similarity measure that can handle general $\mathcal{EL}$-TBoxes has been introduced in the literature so far. In this section, we therefore define such a concept similarity measure $\sim_c$; this measure can then be used to find all relaxed instances for such TBoxes.

The approach to compute the similarity $C \sim_c D$ w.r.t. $\mathcal{T}$ is to translate the concepts C and D into their canonical models $\mathcal{I}_{C,\mathcal{T}}$ and $\mathcal{I}_{D,\mathcal{T}}$ and then structurally compute the similarities for the elements d_C and d_D in these interpretations. As observed before, $\mathcal{I}_{C,\mathcal{T}}$ and $\mathcal{I}_{D,\mathcal{T}}$ are always finite. The TBox $\mathcal{T}$ itself can be disregarded when computing the interpretation similarity from the canonical models, since all its axioms are already used to construct the models.

We need to define similarity measures on interpretations and their properties, which are derived from the properties of concept similarity measures. All proofs for this section can be found in the technical report (Ecke and Turhan 2013).

Interpretation Similarity Measures

An *interpretation similarity measure* (ISM) is defined as a similarity measure on pointed interpretations, i.e., a function $\sim_{\mathfrak{P}}: \mathfrak{P} \times \mathfrak{P} \to [0, 1]$ such that $p \sim_{\mathfrak{P}} p = 1$ for all $p \in \mathfrak{P}$. It maps any pair of pointed interpretations to a similarity value between 0 and 1.

There are various desirable properties that ISMs can have. We concentrate here on those that directly transfer from analogous properties of CSMs introduced by (Lehmann and Turhan 2012). Given suitable simulation relations $\lesssim$ and $\simeq$ (like e.g., those defined in Definition 1 for $\mathcal{EL}$), we call an interpretation similarity measure:

- *symmetric* iff $p \sim_{\mathfrak{P}} q = q \sim_{\mathfrak{P}} p$ for all $p, q \in \mathfrak{P}$;
- *bounded* iff $\mathfrak{C}(p) \cap \mathfrak{C}(q) \supsetneq \{\top\}$ implies $p \sim_{\mathfrak{P}} q > 0$ for all $p, q \in \mathfrak{P}$;
- *dissimilar closed* iff $\mathfrak{C}(p) \cap \mathfrak{C}(q) = \{\top\}$ implies $p \sim_{\mathfrak{P}} q = 0$ for all $p, q \in \mathfrak{P}$ with $\mathfrak{C}(p) \supsetneq \{\top\}$ and $\mathfrak{C}(q) \supsetneq \{\top\}$;
- *equisimulation invariant* iff for all $p, q, u \in \mathfrak{P}$, $p \simeq q$ implies $p \sim_{\mathfrak{P}} u = q \sim_{\mathfrak{P}} u$;
- *equisimulation closed* iff $p \simeq q \iff p \sim_{\mathfrak{P}} q = 1$ for all $p, q \in \mathfrak{P}$;
- *simulation preserving* iff for all $p, q, r \in \mathfrak{P}$, $r \lesssim q \lesssim p$ implies $p \sim_{\mathfrak{P}} q \geq p \sim_{\mathfrak{P}} r$;
- *reverse simulation preserving* iff $r \lesssim q \lesssim p$ implies $q \sim_{\mathfrak{P}} r \geq p \sim_{\mathfrak{P}} r$ for all $p, q, r \in \mathfrak{P}$.

We now define a parameterizable ISM $\sim_i$, using the simulation relations defined in Definition 1, which correspond to subsumption and equivalence in $\mathcal{EL}$. This is important when lifting those properties to the CSMs $\sim_c$. Given a pointed interpretation $p = (\mathcal{I}, d)$, we denote with

- $\text{CN}(p) = \{A \in N_C \mid d \in A^{\mathcal{I}}\}$ the set of concept names that have d as instance in $\mathcal{I}$, and
- $\text{SC}(p) = \{(r, (\mathcal{I}, e)) \in N_R \times \mathfrak{P} \mid (d, e) \in r^{\mathcal{I}}\}$ the set of direct successors of d in $\mathcal{I}$.

For two pointed interpretations to be perfectly similar, they need to have the same set of concept names and edges labeled with the same roles going to perfectly similar successor elements. Otherwise, the most similar concept names and the most similar direct successors are compared and a similarity value is computed from this pair. For both cases, we need the notion of pairings:

A *pairing* $P \subseteq X \times Y$ is a total binary relation, where totality means that all elements of X and all elements of Y appear in some tuple of P as the first component or second

component, respectively. For two pointed interpretations p and q, we are interested in two types of pairings:

- $P_C(p,q) \subseteq \mathcal{P}(\text{CN}(p) \times \text{CN}(q))$ is the set of all *concept name pairing* on the concept names that p and q are instances of; and
- $P_S(p,q) \subseteq \mathcal{P}(\text{SC}(p) \times \text{SC}(q))$ is the set of all *successor pairings* on the direct successors of p and q.

Note that when one of $\text{CN}(p)$ or $\text{CN}(q)$ is empty, the pairings can not be total. In this case, we use $\{\top\}$ instead of the empty concept name set. Similarly, when one of $\text{SC}(p)$ or $\text{SC}(q)$ is empty, we instead use the set $\{(r_\top, p)\}$ or $\{(r_\top, q)\}$, respectively, where $r_\top \in N_R$ is a new role name. This way, the pairings are always well-defined.

The ISM $\sim_i$ will extend a primitive measure. A *primitive measure* $\sim_{\text{prim}} : N_C \cup \{\top\} \times N_C \cup \{\top\} \cup N_R \times N_R \to [0,1]$ assigns similarity values to each pair of basic concepts (i.e., concept names or $\top$) and each pair of role names. Any primitive measure has to satisfy the following properties: $x \sim_{\text{prim}} x = 1$ for any role name or basic concept x, $\top \sim_{\text{prim}} A = A \sim_{\text{prim}} \top = 0$ for all $A \in N_C$, and similarly $r_\top \sim_{\text{prim}} s = s \sim_{\text{prim}} r_\top = 0$ for all $s \in N_R \setminus \{r_\top\}$. Additionally, for the similarity measure $\sim_i$ to be symmetric, $\sim_{\text{prim}}$ needs to be symmetric as well.

We give a default primitive measure, that simply assigns similarity 0 to pairs of different basic concepts or role names:

$$x \sim_{\text{default}} y = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

However, other primitive measures are imaginable and useful. For example, one might want to express that two colours Red and Orange are similar to *some* degree even if they have different concept names. Additionally, one can assign weights to different concept and role names through a weighting function $g : N_C \cup N_R \to \mathbb{R}_{>0}$ that prioritizes different features of the similarity measure. This function g can be extended to pairs of concept names as $g(A,B) = \max(g(A), g(B))$ and pairs of role names as $g(r,s) = \max(g(r), g(s))$.

Any primitive measure $\sim_{\text{prim}}$ and weighting function g can then be extended to a similarity measure on pointed interpretations by recursively traversing the interpretation graphs, computing the primitive measure for each pair of concepts in the best concept name pairing and combining it with the discounted similarity between all pairs of successors in the best successor pairing at each element.

Definition 14. Given a primitive measure $\sim_{\text{prim}}$, a weighting function g, and a discounting factor $w \in (0,1)$, the ISM $\sim_i$: $\mathfrak{P} \times \mathfrak{P} \to [0,1]$ is defined as follows:

$$p \sim_i q = \max_{\substack{P_C \in P_C(p,q) \\ P_S \in P_S(p,q)}} \left(\frac{\text{sim}(P_C) + \text{sim}(P_S)}{\sum_{(A,B)\in P_C} g(A,B) + \sum_{((r,p'),(s,q'))\in P_S} g(r,s)} \right)$$

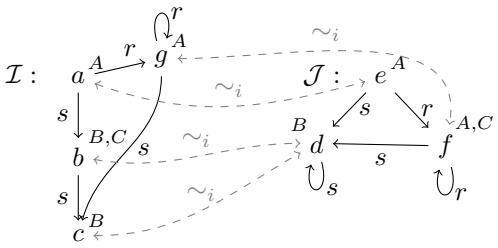


Figure 4: Computation of $(\mathcal{I}, a) \sim_i (\mathcal{J}, e)$ by pairing their successors and recursively computing $\sim_i$ between those.

where

$$\text{sim}(P_C) = \sum_{(A,B)\in P_C} g(A,B)(A \sim_{\text{prim}} B), \text{ and}$$

$$\text{sim}(P_S) = \sum_{\substack{((r,p'),(s,q'))\\ \in P_S}} g(r,s)(r \sim_{\text{prim}} s)((1-w) + w(p' \sim_i q')).$$

The constant w allows for discounting of successors, and should have a value $0 < w < 1$.

Example 15. Consider the pointed interpretations $(\mathcal{I}, a)$ and $(\mathcal{J}, e)$ depicted in Figure 4, the default primitive measure, a weighting function g that assigns 1 to all concept and role names, and a discounting factor $w = 0.8$. To compute the similarity $(\mathcal{I}, a) \sim_i (\mathcal{J}, e)$, we proceed as follows.

1. To compute the similarity between c and d, we find the only concept name pairing is $\{(B,B)\}$ and the only successor pairing is $\{((r_\top, c), (s,d))\}$. Thus we have that $(\mathcal{I}, a) \sim_i (\mathcal{J}, b)$ is $\frac{B \sim_{\text{default}} B + (r_\top \sim_{\text{default}} s) \cdot \dots}{2} = 0.5$.
2. The similarity between b and d, for the best concept name pairing $\{(B,B),(C,B)\}$ and the best successor pairing $\{((s,c),(s,d))\}$, is $\frac{1+0+(0.2+0.8\cdot 0.5)}{3} = 0.533$.
3. The similarity between g and f, for the best concept name pairing $\{(A,A),(A,C)\}$ and the best successor pairing $\{((s,c),(s,d)),((r,g),(r,f))\}$, is the solution to the equation $x = \frac{1+0+(0.2+0.8\cdot 0.5)+(0.2+0.8\cdot x)}{4}$, which is $x = 0.563$.
4. Finally, the similarity between a and e, for the best concept name pairing $\{(A,A)\}$ and the best successor pairing $\{((s,b),(s,d)),((r,g),(r,f))\}$, is $\frac{1+(0.2+0.8\cdot 0.533)+(0.2+0.8\cdot 0.563)}{3} = 0.759$.

If role s is very important for the similarity, one can adapt the similarity measure by increase the weight $g(s)$. Similarly, if concept name A is less important than the other concept names, since it only describes a broad category, one can decrease its weight. One can also change the primitive measure to evaluate $r \sim_{\text{prim}} s = 0.5$ if roles r and s actually denote similar relations, even though they use different role names. In general, this might imply that different best pairings will be found. Finally, if one wants to put more emphasize on the similarity of the role names itself and less on the similarity of the actual elements of the successors, one can do so by reducing the discounting factor. These three degrees of freedom allow for the adaptation of $\sim_i$ to many different use cases.

Note that by defining the ISM $\sim_i$ this way, it is not equisimulation closed. The reason is that the successor pairing always connects successors symmetrically, which gives rise to problems for the case where one successor of an element simulates a second successor. To regain equivalence invariance, one can first normalize the interpretations $\mathcal{I}$ and $\mathcal{J}$ before applying the similarity measure.

Definition 16 (normal form for interpretations). An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is in *normal form* if there are no elements $a, b, c \in \Delta^{\mathcal{I}}$ with $\{(a,b),(a,c)\} \in r^{\mathcal{I}}$ and $(\mathcal{I}, b) \lesssim (\mathcal{I}, c)$, i.e., no node has two successor nodes for the same role name that are in a simulation relation.

Any interpretation $\mathcal{I}$ can be transformed into normal form as follows:

1. remove all edges $(a,b) \in r^{\mathcal{I}}$ in the interpretation graph, for which there exists an edge $(a,c) \in r^{\mathcal{I}}$ such that $(\mathcal{I}, b) \lesssim (\mathcal{I}, c)$ but not $(\mathcal{I}, b) \simeq (\mathcal{I}, c)$
2. for all edges $(a, b_0) \in r^{\mathcal{I}}$, check if there are other edges $(a, b_i) \in r^{\mathcal{I}}$, $i > 0$, with $(\mathcal{I}, b_0) \simeq (\mathcal{I}, b_i)$ and choose one representative b_j; then remove all other edges (a, b_i), $i \neq j$, from $r^{\mathcal{I}}$.

Note that this normalization is well-defined, since the pointed interpretations are always finite, and simulations can be computed in polynomial time in the size of the interpretation. In the following, whenever we write $(\mathcal{I}, a) \sim_i (\mathcal{J}, b)$, we implicitly assume that $\mathcal{I}$ and $\mathcal{J}$ have been normalized first. Using this, we can finally show that $\sim_i$ is a well-defined ISM with formal properties:

Theorem 17. *The similarity measure $\sim_i$ is well-defined, i.e., $p \sim_i q$ has a unique solution for all pointed interpretations $p, q \in \mathfrak{P}$. Furthermore $\sim_i$ is symmetric, bounded, dissimilar closed, equisimulation invariant, and equisimulation closed for the simulation relations defined in Definition 1 and normalized pointed interpretations.*

Although $\sim_i$ as given in Definition 14 is well-defined, it cannot be used directly to compute the similarity value, since cycles in the interpretation would lead to infinite recursion. Instead, one can view the defining equation as an iterative algorithm: When starting with a similarity value of 0 between all elements of two interpretations $\mathcal{I}$ and $\mathcal{J}$, and iteratively applying the equation to update those similarity values, they will converge to the solution in the limit. This follows from the Banach fixed-point theorem (Banach 1922), as $\sim_i$ can be seen as a contraction mapping on the similarity values between all elements of $\mathcal{I}$ and $\mathcal{J}$.

Using this canonical model, we define a concept similarity measure $\sim_c$ on $\mathcal{EL}$-concept descriptions w.r.t. a general $\mathcal{EL}$-TBox $\mathcal{T}$ as follows:

$$C \sim_c D = (\mathcal{I}'_{C,\mathcal{T}}, d_C) \sim_i (\mathcal{I}'_{D,\mathcal{T}}, d_D),$$

where $\mathcal{I}'_{C,T}$ and $\mathcal{I}'_{D,\mathcal{T}}$ are the normalized canonical models of C and D w.r.t. $\mathcal{T}$.

The concept similarity measure $\sim_c$ inherits the useful properties of $\sim_i$, since the properties for interpretation similarity measures were defined to correspond exactly to the concept similarity properties given in the preliminaries.

Theorem 18 (Properties of $\sim_c$). *The concept similarity measure $\sim_c$ is symmetric, bounded, dissimilar closed, equivalence invariant, and equivalence closed.*

The proof for this theorem is given in (Ecke and Turhan 2013).

Computing Relaxed Instances w.r.t. $\sim_c$

First we define the notion of *fully expanded concepts* also for the general $\mathcal{EL}$ case:

Definition 19 (fully expanded concept). Let $\mathcal{T}$ be a general $\mathcal{EL}$-TBox. A concept description C is *fully expanded* w.r.t. $\mathcal{T}$ iff for all GCIs $D \sqsubseteq E \in \mathcal{T}$ with $C \sqsubseteq_{\mathcal{T}} \exists r_1 \ldots \exists r_n.D$ we have that $\exists r_1 \ldots \exists r_n.E$ is a generalized concept of C.

For the computation of relaxed instances for $\sim_c$, recall that $a \in Relax_t^{\sim}(Q)$ can be computed for unfoldable terminologies by checking all generalized concepts of the k-msc(a) for $k = \text{rd}(Q)$. Of course, as soon as we have a general TBox, expanding Q may result in an infinite role-depth by expanding cyclic definitions, so this approach does not work directly here. However, given a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, one can express the fully expanded msc(a) in $\mathcal{EL}$ by unraveling the canonical model $\mathcal{I}_{\mathcal{K}}$ starting from d_a, and thus for any concept C we have

$$C \sim_c \text{msc}(a) = (\mathcal{I}_{C,\mathcal{T}}{}', d_C) \sim_i (\mathcal{I}_{\mathcal{K}}{}', d_a),$$

where $\mathcal{I}_{C,\mathcal{T}}{}'$ and $\mathcal{I}_{\mathcal{K}}{}'$ are the normalized canonical models of C and $\mathcal{A}$ w.r.t. the TBox $\mathcal{T}$. The canonical model $\mathcal{I}_{\mathcal{K}}$, in contrast to the fully expanded msc, is always finite.

However, we do not need to compute the similarity between the query concept Q and the msc(a) directly, but find the maximal similarity between Q and generalized concepts of msc(a). Generalizing a concept C is possible by removing concept names or existential restriction, which corresponds on the interpretation side to only taking subsets of the concept names $S_{\text{CN}} \subseteq \text{CN}(q)$ and successors $S_{\text{SC}} \subseteq \text{SC}(q)$ of the pointed interpretations $q = (\mathcal{I}_{C,\mathcal{T}}, d_C)$ and all its successors. This gives the algorithm in Figure 5 to iteratively compute the maximal similarity between a pointed interpretation p and all generalizations of the pointed interpretation q.

Using this, the algorithm to actually compute all relaxed instances of a query concept Q w.r.t. $\sim_c$ is conceptually quite easy, as it only needs to compute the maximal similarities between Q and all individuals a and check whether they are larger than t. The algorithm is depicted in Figure 6.

Example 20. Consider the the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consisting of the following TBox $\mathcal{T}$ and knowledge base $\mathcal{A}$:

$$\begin{aligned}\mathcal{T} &= \{\, A \sqsubseteq \exists r.A \sqcap \exists s.B,\ \ C \sqsubseteq \exists s.B \,\},\\ \mathcal{A} &= \{\, A(e),\ \ A \sqcap C(f),\ \ B(d),\\ &\qquad r(e,f),\ \ r(f,f),\ \ s(f,d),\\ &\qquad s(e,d),\ \ s(d,d) \,\}.\end{aligned}$$

The similarity measure used is the same as in Example 15, with the default weighting function, the discounting factor $w = 0.8$ and the default primitive measure. Given the query concept $Q = A \sqcap \exists s.(B \sqcap C)$, the normalized canonical

Procedure: maxsim $(\mathcal{I}, \mathcal{J}, \sim_{\text{prim}}, g, w)$
Input: $\mathcal{I}, \mathcal{J}$: finite interpretations; $\sim_{\text{prim}}$: primitive measure; g: weighting function; $w \in (0,1)$: discount factor
Output: maximal similarities between $p = (\mathcal{I}, a)$ and all generalizations of $q = (\mathcal{J}, b)$

1: $\text{msim}_0(d, e) \leftarrow 0$ for all $d \in \Delta^{\mathcal{I}}$ and $e \in \Delta^{\mathcal{J}}$
2: **for** $i \leftarrow 1$ **to** n **do**
3: **for all** $d \in \Delta^{\mathcal{I}}$ and $e \in \Delta^{\mathcal{J}}$ **do**
4: $\text{msim}_i(d, e) \leftarrow \max_{\substack{S_{\text{CN}} \subseteq \text{CN}(e) \\ S_{\text{SC}} \subseteq \text{SC}(e)}} \Big(\max_{\substack{P_C \subseteq \text{CN}(d) \times S_{\text{CN}} \\ P_S \subseteq \text{SC}(d) \times S_{\text{SC}}}} \text{similarity}(P_C, P_S, \sim_{\text{prim}}, g, w, i) \Big)$
5: **end for**
6: **end for**

Procedure: similarity $(P_C, P_S, \sim_{\text{prim}}, g, w, i)$

1: $\text{sim}(P_C) \leftarrow \sum_{(A,B) \in P_C} g(A, B)(A \sim_{\text{prim}} B)$
2: $\text{sim}(P_S) \leftarrow \sum_{((r,p'),(s,q')) \in P_S} g(r, s)(r \sim_{\text{prim}} s)\big((1-w) + w \cdot \text{msim}_{i-1}(p', q)\big)$
3: **return** $\dfrac{\text{sim}(P_C) + \text{sim}(P_S)}{\sum_{(A,B) \in P_C} g(A, B) + \sum_{((r,p'),(s,q')) \in P_S} g(r, s)}$

Figure 5: Algorithm to compute the maximal similarities between all elements of two finite interpretations $\mathcal{I}$ and $\mathcal{J}$.

models $\mathcal{I}'_{Q,\mathcal{T}}$ and $\mathcal{I}'_{\mathcal{K}}$ are exactly the interpretations $\mathcal{I}$ and $\mathcal{J}$ found in Example 15, with $d_Q = a$. The algorithm maxsim will compute the following similarity values msim_i between d_Q and d, e, f in each iteration i:

i	$\text{msim}_i(d_Q, d)$	$\text{msim}_i(d_Q, e)$	$\text{msim}_i(d_Q, f)$
0	0	0	0
1	0.067	0.467	0.35
2	0.173	0.667	0.5
3	0.209	0.748	0.561
4	0.209	0.757	0.567
5	0.209	0.758	0.569

Since the both canonical models are cyclic, the similarity values will change slightly in further iterations, but already after the 5th, the maximal error is less than 0.05%. Using these values, we see that for the threshold $t = 0.5$, both e and f are relaxed instance of Q, while d is not.

The maxsim_i values computed in the algorithm monotonically converge from below to the maximal similarities between generalized concepts of the most specific concept of an individual and the query concept. Thus, for any individual a, which is a relaxed instance of Q with a threshold strictly larger than t, there exists $i \in \mathbb{N}$ such that for all $j > i$ we have $\text{maxsim}_j(Q, a) > t$. Thus, the algorithm is sound and complete in the following sense:

Theorem 21. *Let $\sim_c$ be the CSM derived from $\sim_i$ with the primitive measure $\sim_{\text{prim}}$, the weighting function g and the discounting factor w. Then the algorithm* relaxed-instances *is sound and complete:*

1. Soundness*: If $a \in$ relaxed-instances$(Q, \mathcal{K}, t, \sim_{\text{prim}}, g, w)$ for a number n of iterations, then $a \in Relax_t^{\sim_c}(Q)$.*
2. Completeness*: If $a \in Relax_t^{\sim_c}(Q)$, then there exists an $i \in \mathbb{N}$ such that for all $n \geq i$ iterations it holds that $a \in$ relaxed-instances$(Q, \mathcal{K}, t, \sim_{\text{prim}}, g, w)$.*

Furthermore, the algorithm converges quite fast: For any iteration, the difference between the actual similarity and the computed value reduces by a factor of w. This is again a direct consequence of the Banach fixed-point theorem, as w is an upper bound for the Lipschitz constant of the contraction mapping. This means that, to reduce the error tolerance of the solutions by a constant factor, e.g. one tenth, only a constant number of iterations need to be done additionally. However, one cannot compute how many iterations are needed beforehand and cannot be sure if, at any given point, the algorithm already found all relaxed instances, or if some relaxed instances with a maximal similarity very close to the threshold t are still missing.

If applying the algorithm relaxed-instances w.r.t. unfoldable TBoxes $\mathcal{T}$, then maxsim will however return the exact answer after exactly k iterations, where $k = \text{rd}(Q) + 1$ is the role-depth of the query concept Q expanded w.r.t. $\mathcal{T}$. In this case, the algorithm can be made deterministic and, since each iteration of maxsim only takes polynomial time in the size of $\mathcal{K}$ and Q, runs in PTIME.

Procedure: relaxed-instances $(Q, \mathcal{K}, t, \sim_{\text{prim}}, g, w)$
Input: Q: $\mathcal{EL}$-concept; $\mathcal{K} = (\mathcal{T}, \mathcal{A})$: $\mathcal{EL}$-KB; $t \in [0,1]$: threshold; $\sim_{\text{prim}}$: primitive measure; g: weighting function; $w \in (0,1)$: discounting factor
Output: individuals $a \in Relax_t^{\sim_c}(Q)$

1: compute canonical models $\mathcal{I}_{Q,\mathcal{T}}$ and $\mathcal{I}_{\mathcal{K}}$
2: $\text{maxsim}(d, e) \leftarrow \text{maxsim}(\mathcal{I}_{Q,\mathcal{T}}, \mathcal{I}_{\mathcal{K}}, \sim_{\text{prim}}, g, w)$
3: **return** $\{a \in N_I \cap \text{Sig}(\mathcal{A}) \mid \text{maxsim}(d_Q, d_a) > t\}$

Figure 6: Algorithm to compute all relaxed instances of a query concept Q w.r.t. a knowledge base $\mathcal{K}$ and threshold t.

6 Conclusions

We have proposed a new reasoning service that allows relaxed instance query answering for application-specific notions of similarity by the appropriate choice of a CSM. The inference has two main degrees of freedom: in the choice of the CSM, and in the degree of relaxation of the concept. Intuitively, different similarity measures yield different weights on specific criteria. For example, one could require that small changes inside existential restrictions produce a high level of dissimilarity.

Further we investigated necessary requirements for the CSMs to be employed. We devised computation algorithms for relaxed instances in the setting with unfoldable and with general TBoxes. For the latter setting we needed to introduce a new family of CSMs that take the whole information from

general TBoxes into account. The $\sim_c$ CSMs are, to the best of our knowledge, the first CSMs for general TBoxes. Based on these we gave an computation algorithm for relaxed instances w.r.t. general TBoxes.

There are many options for future work. On the theoretical side it would be interesting to explore how this approach can be extended to expressive DLs. We conjecture that our approach extends to Horn-DLs, since they induce canonical models as well. On the practical side there is plenty of room for optimizations. For instance, the use of a concept that states necessary conditions in combination with the query concept can considerably reduce the number of individuals to be checked in practice. Furthermore, while the complexity of each iteration in the general case is polynomial, the need to check every subset and every pairing is certainly inefficient. Methods to reduce the subsets and pairings that need to be considered are expedient to make this work in practice.

Acknowledgements

A. Ecke is supported by the German Research Foundation (DFG) in the Graduiertenkolleg 1763 (QuantLA). R. Peñaloza is partially supported by DFG within the Cluster of Excellence 'Center for Advancing Electronics Dresden' (cfAED). A.-Y. Turhan is partially supported by DFG in the Collaborative Research Center 912 'Highly Adaptive Energy-Efficient Computing' (HAEC).

References

Alvarez, M. A., and Yan, C. 2011. A graph-based semantic similarity measure for the gene ontology. *J. Bioinformatics and Computational Biology* 9(6):681–695.

Banach, S. 1922. Sur les oprations dans les ensembles abstraits et leur application aux quations intgrales. *Fundamenta Mathematicae* 3(1):133–181.

Borgida, A.; Walsh, T.; and Hirsh, H. 2005. Towards measuring similarity in description logics. volume 147 of *CEUR Workshop Proceedings*.

Borgwardt, S., and Peñaloza, R. 2012. Undecidability of fuzzy description logics. In Brewka, G.; Eiter, T.; and McIlraith, S. A., eds., *KR-12*, 232–242. AAAI Press.

d'Amato, C.; Fanizzi, N.; and Esposito, F. 2005. A semantic similarity measure for expressive description logics. In *Proc. of Convegno Italiano di Logica Computazionale, CILC05*.

d'Amato, C.; Staab, S.; and Fanizzi, N. 2008. On the influence of description logics ontologies on conceptual similarity. In Gangemi, A., and Euzenat, J., eds., *Proceedings of Knowledge Engineering: Practice and Patterns, 16th Int. Conf. (EKAW 2008)*, volume 5268, 48–63.

Ecke, A., and Turhan, A.-Y. 2013. Similarity measures for computing relaxed instances w.r.t. general $\mathcal{EL}$-TBoxes. LTCS-Report 13-12, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany. See http://lat.inf.tu-dresden.de/research/reports.html.

Ecke, A.; Peñaloza, R.; and Turhan, A.-Y. 2013. Towards instance query answering for concepts relaxed by similarity measures. In Godo, L.; Prade, H.; and Qi, G., eds., *Workshop on Weighted Logics for AI (in conjunction with IJCAI'13)*.

Gene Ontology Consortium, T. 2000. Gene Ontology: Tool for the unification of biology. *Nature Genetics* 25:25–29.

Janowicz, K., and Wilkes, M. 2009. Sim-dla: A novel semantic similarity measure for description logics reducing inter-concept to inter-instance similarity. volume 5554, 353–367.

Lehmann, K., and Turhan, A.-Y. 2012. A framework for semantic-based similarity measures for $\mathcal{ELH}$-concepts. 307–319.

Lord, P. W.; Stevens, R. D.; Brass, A.; and Goble, C. A. 2003. Investigating semantic similarity measures across the gene ontology: The relationship between sequence and annotation. *Bioinformatics* 19(10):1275–1283.

Lutz, C., and Wolter, F. 2010. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *Journal of Symbolic Computation* 45(2):194–228.

Mistry, M., and Pavlidis, P. 2008. Gene ontology term overlap as a measure of gene functional similarity. *BMC Bioinformatics* 9.

Peñaloza, R., and Turhan, A.-Y. 2011. A practical approach for computing generalization inferences in $\mathcal{EL}$. In Grobelnik, M., and Simperl, E., eds., *Proceedings of the 8th European Semantic Web Conference (ESWC'11)*, 410–423.

Schlicker, A.; Domingues, F. S.; Rahnenführer, J.; and Lengauer, T. 2006. A new measure for functional similarity of gene products based on gene ontology. *BMC Bioinformatics* 7:302.

Suntisrivaraporn, B. 2013. A similarity measure for the description logic $\mathcal{EL}$ with unfoldable terminologies. In *5th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, 408–413.

Stable Model Semantics for Guarded Existential Rules and Description Logics

Georg Gottlob[1] **André Hernich**[2] **Clemens Kupke**[3] **Thomas Lukasiewicz**[1]

[1]Department of Computer Science, University of Oxford, UK
firstname.lastname@cs.ox.ac.uk
[2]Department of Computer Science, University of Liverpool, UK
andre.hernich@liverpool.ac.uk
[3]Department of Computer and Information Sciences, University of Strathclyde, Glasgow, UK
clemens.kupke@strath.ac.uk

Abstract

We tackle a long-standing open research problem and prove the decidability of query answering under the stable model semantics for guarded existential rules, where rule bodies may contain negated atoms, and provide complexity results. The results extend to guarded Datalog$^{\pm}$ with negation, and thus provide a natural and decidable stable model semantics to description logics such as $\mathcal{ELHI}$ and *DL-Lite*$_{\mathcal{R}}$.

1 Introduction

1.1 Existential Rules, Datalog$^{\pm}$, and DLs

Existential rules have attracted much recent interest for knowledge representation and reasoning (see (Calì et al. 2010) for an overview). An example for an existential rule is

$$\textit{EmpProj(x,y), BaseLevel(x)} \rightarrow \exists z\ \textit{IsManagedBy(x,z)}, \quad (1)$$

which states that every base-level employee working in a project must be managed by somebody. Such rules are also well-known as *tuple-generating dependencies (TGDs)*, see (Beeri and Vardi 1984). In the Datalog$^{\pm}$ languages (Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012), TGDs are complemented by a few other types of rules, notably by *negative constraints (NCs)*, such as *Ceo(x), BaseLevel(x)* $\rightarrow \bot$, where $\bot$ stands for *false*.[1]

Query answering based on rule sets is the following decision problem: For a database D, a set of rules Σ, and a Boolean query Q, decide whether $D \cup \Sigma \models Q$. The queries considered in this paper are unions of Boolean conjunctive queries (UBCQs). Query answering with TGDs is undecidable (Beeri and Vardi 1981), even in the case of fixed sets Σ of TGDs (Calì, Gottlob, and Kifer 2013) and singleton sets of TGDs (Baget, Leclère, and Mugnier 2010). Therefore, to obtain decidability, restrictions need to be imposed on TGDs. Semantic conditions that lead to decidability were described in (Baget, Leclère, and Mugnier 2010; Baget et al. 2009; 2011; Thomazo 2011; Calì, Gottlob, and Pieris 2012; Gottlob, Manna, and Pieris 2013). Related syntactic conditions led to the Datalog$^{\pm}$ family (Calì, Gottlob, and Lukasiewicz 2012). A class of major relevance are *guarded* TGDs (GTGDs). A TGD is guarded if its body contains an atom, called *guard*, that covers all body variables. NCs may be safely used together with guarded TGDs; their bodies are not required to contain a guard atom.

A nice and important aspect of guarded rules is that they generalize well-known description logics (DLs). All the DLs of the *DL-Lite* family in (Calvanese et al. 2007; Poggi et al. 2008) and the DL $\mathcal{ELHI}$ (Baader, Brandt, and Lutz 2005) can be embedded into guarded Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012). In particular, this holds for *DL-Lite*$_{\mathcal{R}}$, the theoretical basis of the QL profile of the Web ontology language OWL 2. To appreciate how easily DL axioms can be transformed into Datalog$^{\pm}$, note that rule (1) can be written as $\exists\text{EmpProj} \sqcap \text{BaseLevel} \sqsubseteq \exists\text{IsManagedBy}$.

1.2 The Main Problem: Stable Negation

The main goal of this paper is to add non-monotonic negation under the *stable model semantics (SMS)* (Gelfond and Lifschitz 1988) (see also (Subrahmanian 1999; Dantsin et al. 2001) for an overview of the semantic and computational properties of the SMS) to guarded rules (possibly with NCs) where body atoms may be negated. This is considered a hot open research problem, and has, to our knowledge, not been solved previously. Rules that may contain negated atoms of the form $notR(\mathbf{x})$ in their bodies, are called *normal TGDs (NTGDs)*. Guarded such rules are *guarded NTGDs* (*GNTGDs*). The standard way of defining the SMS for them is inherited from the well-known stable model semantics of normal logic programming with function symbols (Gelfond and Lifschitz 1988). In fact, we can replace existentially quantified variables in program heads by Skolem terms.

Example 1 Let $D_a = \{Person(mary)\}$ be a database and let Σ_a be the following rule set expressing that each person has at least one parent, and that each person belongs to either an odd generation or to an even generation, and that odd and even alternate between one generation and the next.

$$\begin{aligned} \textit{Person(x)} &\rightarrow \exists y\ \textit{Parent(x,y)}, \\ \textit{Parent(x,y)} &\rightarrow \textit{Person(y)}, \\ \textit{Person(x), not Even(x)} &\rightarrow \textit{Odd(x)}, \\ \textit{Person(x), not Odd(x)} &\rightarrow \textit{Even(x)}, \end{aligned}$$

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]Often also *equality-generating dependencies (EGDs)* are used, which may express key constraints (keys) or functional dependencies. Here, we do not consider EGDs. But, as will be explained in the full paper, all results of this paper carry over to the case where rule sets are complemented with a restricted class of EGDs, called *non-conflicting EGDs* (Calì, Gottlob, and Lukasiewicz 2012).

$$\textit{Parent(x,y),Odd(x)} \rightarrow \textit{Even(y)},$$
$$\textit{Parent(x,y), Even(x)} \rightarrow \textit{Odd(y)}.$$

After skolemization, the first rule becomes:

$$\textit{Person(x)} \rightarrow \textit{Parent(x,f(x))},$$

where f is a Skolem function. This program has exactly two stable models, each consisting of an infinite chain rooted in the constant *mary*, one containing (among others) the facts *Odd(mary), Even(f(mary)), Odd(f(f(mary)))*, and so on, the other containing the fact *Even(mary), Odd(f(mary)), Even(f(f(mary)))*. Let Q_a be the conjunctive query $\exists x, y, z$ *(Parent(x,y), Parent(y,z), Odd(x), Odd(z))*, and let Q'_a be the query $\exists x, y$ *(Parent(x,y), Odd(x), Odd(y))*. Then, under the SMS, $(D_a \cup \Sigma_a) \models Q_a$ but $(D_a \cup \Sigma_a) \not\models Q'_a$. ∎

Note that further examples are given in Section 7. It is easy to exhibit rule sets Σ that generate infinitely many infinite stable models, as illustrated by the following example.

Example 2 Let $D_b = D_a = \{Person(mary)\}$, and let Σ_b be obtained by taking just the first four rules of the rule set Σ_a of Example 1. Then, there is an uncountably infinite number of stable models of $D_b \cup \Sigma_b$, one for each possibility of having for each Skolem term $f^i(mary)$ either *Even(f^i(mary))* or *Odd(f^i(mary))*, where $1 \leqslant i \leqslant \infty$. For the query Q_a of Example 1, note that $(D_b \cup \Sigma_b) \not\models Q_a$. ∎

It seems that the interplay of nonmonotonic negation with these two types of infinity made it hard to understand how the SMS for GNTGDs could be brought under control. The main questions that we pose and tackle are thus: Is query answering under the stable model semantics for GNTGDs decidable? If so, what is the complexity of this problem and what kind of algorithms are appropriate for it?

1.3 Results

An insightful initial observation while addressing the above decidability problem was that if a guarded rule set with negated atoms admits a stable model, then it admits a tree-shaped one. In fact, each such stable model is a model of some Gelfond-Lifschitz transform of the ground version *ground*(Σ) of the original rule set. For the latter (which are guarded positive existential rule sets), we know that there are universal tree-shaped models (Calì, Gottlob, and Kifer 2013) that arise via the well-known chase procedure (Maier, Mendelzon, and Sagiv 1979; Beeri and Vardi 1984). We could thus encode query answering under the SMS for existential Datalog$^{\pm}$ in monadic second-order logic (MSO), making stable models correspond to binary trees. Decidability of this problem then follows from Rabin's Theorem (1969), by which satisfiability of MSO is decidable over the class of binary trees. This is very briefly described in Section 3.3 and fully proven in the extended paper (Gottlob et al. 2014). Related ideas for deciding other guarded logics or DLs via MSO can be found in (Eiter and Simkus 2010) and in (Motik, Horrocks, and Sattler 2009). We thus have shown:

Theorem. There is an algorithm that, given a database D, a finite set Σ of GNTGDs, and a BCQ Q, decides whether Q is true in all stable models for D and Σ.

Unfortunately, the MSO formula constructed for proving the above result does not give us a tight complexity upper bound for query evaluation with GNTGDs. To arrive at a matching bound, we pursued the following different ideas. Recall that finding a stable model essentially involves first guessing a model and then checking that this model is indeed the least fixpoint of the Gelfond-Lifschitz transform Σ^M of the original rule set Σ relative to the guessed model M. Given that query answering under the classical first-order (FO) semantics for guarded disjunctive TGDs (GDTGDs) has been recently defined and studied (Alviano et al. 2012; Gottlob et al. 2012b; Bourhis, Morak, and Pieris 2013), why not simulating the stable model semantics of GNTGDs by GDTGDs under the FO semantics? For the guessing phase, this turned out to be possible, as guessing the truth or falsehood of an atom can be encoded as a classical disjunction. But the checking phase requires computing the complement of Σ^M. This is an inherently nonmonotonic task not feasible in FO. But only one level of nonmotonic inference is needed. It would thus be sufficient to extend DGTDs under FO semantics by *stratified negation*, yielding the formalism of stratified sets of guarded disjunctive normal TGDs (GDNTGDs). In this setting, atom entailment within each stratum is the classical monotonic FO-entailment, but each stratum may access the lower strata nonmonotonically via the *not* operator, i.e., for each atom $\mathbf{a}$ of the Herbrand base, each stratum may assume that *not* $\mathbf{a}$ is true if the predicate of $\mathbf{a}$ belongs to a lower stratum, but $\mathbf{a}$ cannot be derived there.

We thus aimed at proving the decidability and determining the complexity of query answering with stratified GDNTGDs. This turned out to be much harder a challenge than the same problem for GTGDs (Calì, Gottlob, and Lukasiewicz 2012) or stratified GNTGDs. With GTGDs (or stratified GNTGDs), there is only a single, possibly infinite canonical model, and each query UBCQ Q is true in this model iff it is true in a finite part of this model. This finite part consists, for each stratum, of the atoms generated by the chase in that stratum that do not exceed a certain derivation depth. Unfortunately, this is no longer true for models generated by GDTGDs, and thus for stratified GDNTGDs. For a fixed rule set Σ and a fixed query Q, and any arbitrary positive integer d, in general, there may exist "tardive" models M such that $M \models Q$ but $M_d \not\models Q$, where M_d contains all atoms of M whose chase derivation depth is at most d. Not too surprisingly, tardive models may also arise with rules sets Σ of original formalism, namely, GNTGDs under the SMS. To illustrate this, consider the setting of Example 2, and let Q be the query $\exists x$*Even(x)*. Let d be an arbitrary positive integer. There are stable models M such that, for all $1 \leqslant i \leqslant d$, *Odd*$(f^i(mary))$ and *Even*(f^{d+1}). For such models M, it does not suffice to consider the initial part M_d containing all facts of M of chase derivation depth $\leqslant d$, for Q would be false in this initial part, while true in M.

Fortunately, we can cope with tardive models for a query Q due to the following observation. Whenever for a stratified GDNTGD set Σ and a query Q, a tardive model exists in which the query is not answered within a certain (doubly exponential) derivation depth, then there must exist *another model* in which Q is generally false, and thus Q must be

overall answered *false*, because Q is false in some model. It is thus still possible to reconduct query answering to a finitary setting. This insight led us to develop alternating algorithms for query answering under stratified sets of GDNTDGs, which give rise to the following results, where $\models_{\mathsf{strat}}$ is entailment under the described stratified semantics:

Theorem. Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a stratified set of GDNTGDs on $\mathcal{R}$, and Q a union of CQs over $\mathcal{R}$. Deciding $D \cup \Sigma \models_{\mathsf{strat}} Q$ has the following complexity:

1. 2-EXPTIME-complete in general;
2. EXPTIME-complete in case the maximum arity w of the relation symbols in $\mathcal{R}$ is fixed;
3. coNP-complete in case $|\mathcal{R}|$, w, and the number of atoms in Q are fixed.

In Section 6, we present the desired polynomial-time translation from GNTGDs under the SMS to GDNTGDs under FO semantics with stratified negation. This solves the main problem attacked in this paper. From this translation, and from easily obtained lower bounds, for query-answering for GNTGDs, we get exactly the same complexity bounds as those in the above theorem. This solves our main complexity problem. Moreover, all the above decidability and complexity results also hold in case negative constraints are added to the rule sets. (This is trivial: To simulate NCs, it suffices to add the body of each NC as a disjunct to the UCQ.)

In Section 7, we finally apply our results to DLs. We immediately obtain a stable model semantics for these DLs, as well as decidability results and complexity upper bounds. In particular, this provides a stable model semantics for the well-known DLs $\mathcal{ELHI}$ and *DL-Lite*$_{\mathcal{R}}$.

In the sequel, we use the letters "G", "D", and "N" in front of TGDs to abbreviate that the TGDs are "guarded", "disjunctive", and "normal", respectively. Other properties of TGDs are written out in textual form whenever necessary. Note that proofs of all results are given in the extended version of this paper (Gottlob et al. 2014).

1.4 Related Work on Rules with Negation

We now discuss a number of previous approaches to extend guarded TGDs with negation.

Stratified Negation. Stratified negation for guarded Datalog$^{\pm}$, and thus automatically for a number of important DLs was introduced in (Calì, Gottlob, and Lukasiewicz 2012) and further extended to the more expressive formalism of *weakly guarded Datalog*$^{\pm}$ in (Arenas, Gottlob, and Pieris 2014). These papers show that stratified negation is very well-behaved in the sense that its addition does not endanger decidability, and actually does not augment the complexity of reasoning and query answering. But this semantics does not apply to the (unstratified) rules in Examples 1 and 2.

Well-Founded Negation. Negation under the well-founded semantics (WFS) for guarded Datalog$^{\pm}$ and covered DLs was studied in (Hernich et al. 2013; Gottlob et al. 2012a) for two variants of the WFS. The version of (Hernich et al. 2013) studies the so-called *standard WFS*, which extends the well-known WFS for logic programming with function symbols, where it is assumed (as here) that different Skolem terms are not unifiable, which means that the *unique name assumption (UNA)* is applied to Skolem terms. The second variant, called *equality-friendly WFS* (Gottlob et al. 2012a) does not use the UNA. Note that both variants would be unsatisfactory for Examples 1 and 2, as they would simply leave the predicates *Odd* and *Even* undefined.

Stable Model Negation. In (Magka, Krötzsch, and Horrocks 2013), new acyclicity and stratification conditions are presented for existential rules with negation in rule bodies, which identify classes of rule sets that have finite and/or unique stable models, and constraints are added on the input facts to further extend these classes. Our work here, in contrast, is not restricted to finite and/or unique stable models, and thus much more general. The only syntactic restriction necessary here is guardedness (or the weaker weakly guardedness). The $\mathbb{FDNC}$ programs in (Eiter and Simkus 2010) combine nonmonotonic negation and rules, allowing also the use of function symbols. Decidability is obtained by restricting the structure of rules to one of seven predefined forms.

Hybrid Approaches. Less closely related approaches are loosely and tightly coupled dl-programs (Eiter et al. 2008; Lukasiewicz 2010), as well as the hybrid MKNF knowledge bases (Motik and Rosati 2010). More precisely, the former loosely and tightly, respectively, combine a description logic knowledge base L and a logic program P. Rule bodies in P may contain queries to L, which may also contain facts as additional input to L, in the loosely coupled case, while concepts and roles from L are used as predicates in P in the tightly coupled case. Decidability is based on the finiteness of stable models. The hybrid MKNF knowledge bases allow for querying a description logic knowledge base L via the operators $\mathbf{K}$ and $\mathbf{not}$. Decidability is obtained via the so-called DL-safety condition, which makes the rules applicable only to explicitly known individuals. In our approach here, in contrast, stable models may be infinite, and no restrictive DL-safety of rules is assumed.

In summary, none of the above approaches allows one to use the stable model semantics for the general class of guarded normal TGDs.

2 Preliminaries

We now briefly recall some basics on Datalog$^{\pm}$ (Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012), namely, on relational databases, (Boolean) conjunctive queries ((B)CQs), tuple-generating dependencies (TGDs), and negative constraints. For equality-generating dependencies (EGDs), in particular, non-conflicting keys, we refer to (Calì, Gottlob, and Lukasiewicz 2012).

Databases and Queries. We assume (i) an infinite universe of *(data) constants* Δ (which constitute the "normal" domain of a database), (ii) an infinite set of *(labeled) nulls* Δ_N (used as "fresh" Skolem terms, which are placeholders for unknown values), and (iii) an infinite set of variables $\mathcal{V}$ (used in queries and constraints). We denote by $\mathbf{X}$ sequences of variables $X_1, \ldots, X_k$ with $k \geqslant 0$. We assume a *relational schema* $\mathcal{R}$, which is a finite set of *relation names* (or *predicate symbols*, or simply *predicates*).

A *position* $P[i]$ identifies the i-th argument of a predicate P. A *term* t is a constant or variable. An *atomic formula* (or *atom*) $\mathbf{a}$ has the form $P(t_1, ..., t_n)$, where P is an n-ary predicate, and $t_1, ..., t_n$ are terms. A conjunction of atoms is often identified with the set of all its atoms.

A *database (instance)* D for a relational schema $\mathcal{R}$ is a (possibly infinite) set of atoms with predicates from $\mathcal{R}$ and arguments from Δ. We denote by $dom(D)$ the set of all elements of Δ that occur in D. A *conjunctive query (CQ)* over $\mathcal{R}$ has the form $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms with the variables $\mathbf{X}$ and $\mathbf{Y}$ (and eventually constants). Note that $\Phi(\mathbf{X}, \mathbf{Y})$ may also contain equalities but no inequalities. A *Boolean CQ (BCQ)* over $\mathcal{R}$ is a CQ of the form $Q()$. We often write a BCQ as the set of all its atoms, having constants and variables as arguments, and omitting the quantifiers. Answers to CQs and BCQs are defined via *homomorphisms*, which are mappings $\mu\colon \Delta \cup \mathcal{V} \to \Delta \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, and (ii) μ is naturally extended to atoms, sets of atoms, and conjunctions of atoms. The set of all *answers* to a CQ $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$ over a database D, denoted $Q(D)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism $\mu\colon \mathbf{X} \cup \mathbf{Y} \to \Delta$ such that $\mu(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The *answer* to a BCQ $Q() = \exists \mathbf{Y}\, \Phi(\mathbf{Y})$ over a database D is *Yes*, denoted $D \models Q$, iff $Q(D) \neq \varnothing$, i.e., there is a homomorphism $\mu\colon \mathbf{Y} \to \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{Y})) \subseteq D$.

Tuple-Generating Dependencies (TGDs). Tuple-generating dependencies (TGDs) describe constraints on databases in the form of generalized Datalog rules with existentially quantified conjunctions of atoms in rule heads; their syntax and semantics are as follows. Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ is a first-order formula $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$, called the *body* and the *head* of σ, denoted $body(\sigma)$ and $head(\sigma)$, respectively. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there is a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D, then h is extendable to a homomorphism h' that maps the atoms of $\Psi(\mathbf{X}, \mathbf{Z})$ to atoms of D. As TGDs can be reduced to TGDs with only single atoms in their heads, in the sequel, every TGD has w.l.o.g. a single atom in its head. A TGD σ is *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ. The leftmost such atom is the *guard* of σ.

Query answering under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs is defined as follows. For a database D for $\mathcal{R}$, and a set of TGDs Σ on $\mathcal{R}$, the set of *models* of D and Σ, denoted $mods(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $ans(Q, D, \Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in mods(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $ans(Q, D, \Sigma) \neq \varnothing$.

Negative Constraints. Another crucial ingredient of Datalog$^{\pm}$ for ontological modeling are *negative constraints* (or simply *constraints*), which are first-order formulas of the form $\forall \mathbf{X}\ \Phi(\mathbf{X}) \to \bot$, where $\Phi(\mathbf{X})$ is a conjunction of atoms (not necessarily guarded), called its *body*.

We usually omit the universal quantifiers and implicitly assume all sets of dependencies/constraints to be finite here.

3 Stable Models for Normal Datalog$^{\pm}$

In this section, we recall normal TGDs and BCQs (Calì, Gottlob, and Lukasiewicz 2012), as well as stable models of normal (logic) programs (Gelfond and Lifschitz 1988), and we define the stable model semantics of guarded normal TGDs.

3.1 Normal TGDs and BCQs

Normal TGDs may also contain (default-)negated atoms in their bodies: Given a relational schema $\mathcal{R}$, a *normal TGD (NTGD)* σ has the form $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms and negated atoms over $\mathcal{R}$, and $\Psi(\mathbf{X}, \mathbf{Z})$ is a conjunction of atoms over $\mathcal{R}$. It is also abbreviated as $\Phi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$. We denote by $head(\sigma)$ the atom in the head of σ, and by $body^+(\sigma)$ and $body^-(\sigma)$ the sets of all positive and negative ("$\neg$"-free) atoms in the body of σ, respectively. We say σ is *guarded* iff it contains a positive body atom, denoted $guard(\sigma)$, that contains all universally quantified variables of σ. W.l.o.g., every NTGD has a single atom in its head, and constants in the body of guarded σ occur only in guards. We say σ is *linear* iff σ is guarded and has exactly one positive atom in its body. As for the semantics, an NTGD σ is *satisfied* in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h for all the variables and constants in the body of σ that maps (i) all atoms of $body^+(\sigma)$ to atoms of D and (ii) no atom of $body^-(\sigma)$ to atoms of D, then there exists an extension h' of h that maps all atoms of $head(\sigma)$ to atoms of D.

We next add negation to BCQs. A *normal Boolean conjunctive query (NBCQ)* Q is an existentially closed conjunction of atoms and negated atoms of the form

$$\exists \mathbf{X}\, p_1(\mathbf{X}) \wedge \cdots \wedge p_m(\mathbf{X}) \wedge \neg p_{m+1}(\mathbf{X}) \wedge \cdots \wedge \neg p_{m+n}(\mathbf{X}),$$

where $m \geqslant 1$, $n \geqslant 0$, and the variables of the p_i's are among $\mathbf{X}$. Let $Q^+ = \{p_1(\mathbf{X}), \ldots, p_m(\mathbf{X})\}$ and $Q^- = \{p_{m+1}(\mathbf{X}), \ldots, p_{m+n}(\mathbf{X})\}$. That is, Q^+ (resp., Q^-) is the set of all positive (resp., negative) ("$\neg$"-free) atoms of Q. We say Q is *safe* iff every variable in a negative atom in Q also occurs in a positive atom in Q. We say Q is *covered* iff for every negative atom α in Q, there is a positive atom β in Q such that every argument in α occurs in β. Note that the coveredness of Q implies also the safeness of Q, but not vice versa. In the sequel, w.l.o.g., NBCQs contain no constants. Note that disjunctive normal non-Boolean CQs over finite databases can be reduced to constant-free NBCQs, by first reducing them to NBCQs with constants, and then moving their constants into the TGDs by introducing new predicate symbols.

3.2 Stable Models of Normal Programs

A *normal rule* (or simply *rule*) r is of the form

$$\alpha \leftarrow \beta_1, \ldots, \beta_n, not\ \beta_{n+1}, \ldots, not\ \beta_{n+m}\,, \qquad (1)$$

where $\alpha, \beta_1, \ldots, \beta_{n+m}$ are atoms and $k, m, n \geqslant 0$. We call α the *head* of r, denoted *head*(r), while the conjunction

$\beta_1, \ldots, \beta_n, not\ \beta_{n+1}, \ldots, not\ \beta_{n+m}$ is its *body*. We define $body(r) = body^+(r) \cup body^-(r)$, where $body^+(r) = \{\beta_1, \ldots, \beta_n\}$ and $body^-(r) = \{\beta_{n+1}, \ldots, \beta_{n+m}\}$. We say r is *positive* iff $m = 0$. We say r is a *fact* iff $m = n = 0$. A *normal* (resp., *positive*) *program* P is a finite set of normal (resp., positive) rules.

Let P be a normal program. We denote by HU_P and HB_P the *Herbrand universe* and the *Herbrand base* for P, respectively. A *ground instance* of a rule $r \in P$ is obtained from r by replacing each variable in r by an element from HU_P. We denote by $ground(r)$ (resp., $ground(P)$) the set of all ground instances of r (resp., rules in P). An *interpretation* I for P is a subset of HB_P. Such I is a *model* of a ground rule r iff $\alpha \in I$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \varnothing$. We say I is a *model* of a normal program P iff I is a model of every $r \in ground(P)$.

The *Gelfond-Lifschitz reduct* of a normal program P relative to $I \subseteq HB_P$, denoted P^I, is the (possibly infinite) ground positive program obtained from $ground(P)$ by

1. deleting every rule r such that $B^-(r) \cap I \neq \varnothing$, and
2. deleting the negative body from each remaining rule.

Since P^I is positive, it has a unique minimal model. A *stable model* of a normal program P is an interpretation $I \subseteq HB_P$ such that I is the minimal model of P^I. Note that every stable model of P is also a minimal model of P.

3.3 Stable Models for Guarded Normal TGDs

Let Σ be a finite set of guarded NTGDs, and let D be a database over the same schema as Σ. Given an NTGD $\sigma = \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, the *functional transformation* of σ, denoted σ^f, is the normal rule $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \Psi(\mathbf{X}, \mathbf{f}_\sigma(\mathbf{X}, \mathbf{Y}))$, where $\mathbf{f}_\sigma$ is a vector of function symbols $f_{\sigma,Z}$ for σ, one for every variable Z in $\mathbf{Z}$. Given a set Σ of NTGDs, the *functional transformation* of Σ, denoted Σ^f, is obtained from Σ by replacing each TGD σ in Σ by σ^f. Note that the functional transformation of a finite set of guarded TGDs is a positive program.

Definition 1 Let Σ be a finite set of guarded NTGDs, and let D be a database over the same schema as Σ. Then, a *stable model* for D and Σ is a stable model of the normal program $P = D \cup \Sigma^f$, having D as a collection of ground facts.

Non-constant terms (i.e., those involving function symbols) that occur in stable models for D and Σ are henceforth considered as *nulls*. Let Σ^+ be the finite set of TGDs obtained from Σ by dropping all negative literals from the rules' bodies. Recall the definition of the guarded chase forest for D and Σ^+ from (Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012). The *upper guarded chase forest* for D and Σ, denoted *g-chase*(D, Σ), is the guarded chase forest for D and Σ^+. The following lemma shows that the set of atoms occurring in *g-chase*(D, Σ) provides an overapproximation for any stable model for D and Σ. From this, we infer below that query answering under guarded normal Datalog$^\pm$ with the stable models semantics is decidable. For the proof, we refer to (Gottlob et al. 2014).

Lemma 2 *Let Σ be a finite set of guarded NTGDs, and let D be a database over the same schema as Σ. If S is a stable model for D and Σ, then all atoms of S occur as the label of some node in g-chase(D, Σ).*

Decidability of query answering relative to the stable model semantics can be obtained as a consequence of Rabin's Theorem (1969) via an encoding into monadic second-order logic over the infinite m-ary tree (for suitable $m \in \mathbb{N}$).

Theorem 3 *There is an algorithm that, given a database D, a finite set Σ of guarded NTGDs, and a covered UNBCQ Q, decides whether Q is true in all stable models for D and Σ.*

Proof (Sketch). Given D, Σ, and Q, compute an MSO sentence $\varphi_{D,\Sigma,Q}$ such that Q is true in all stable models for D and Σ iff $T \models \varphi_{D,\Sigma,Q}$. The basic idea for constructing the MSO sentence is to label a subset of the nodes of the full m-ary tree T with a *type* (α, N) consisting of an atom α and a set N of atoms whose arguments occur in α, and then to verify that the subtree T' of T induced by all labelled nodes corresponds to a stable model. Due to space limitations, and because we obtain stricter complexity bounds for query answering in the coming sections, we omit the details of the MSO-translation, and refer the reader to the extended version of this paper (Gottlob et al. 2014). □

4 Disjunctive Rules with Stratified Negation

Disjunctive Normal TGDs and BCQs. Given a relational schema $\mathcal{R}$, a *disjunctive normal TGD (DNTGD)* σ has the form $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms and negated atoms over $\mathcal{R}$, and $\Psi(\mathbf{X}, \mathbf{Z})$ is a disjunction of atoms over $\mathcal{R}$. We denote by $head(\sigma)$ the set of atoms in the head of σ. All other notions like $body^+(\sigma)$, $body^-(\sigma)$, guardedness, and $guard(\sigma)$ are defined as for NTGDs. A DNTGD σ is *satisfied* in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h for all the variables and constants in the body of σ that maps (i) all atoms of $body^+(\sigma)$ to atoms of D and (ii) no atom of $body^-(\sigma)$ to atoms of D, then there exists an extension h' of h that maps some atoms of $head(\sigma)$ to atoms of D.

A *union of normal Boolean conjunctive queries (UNBCQ)* Q is a disjunction of $k > 0$ NBCQs Q_i. It is *safe* (resp., *covered*) iff all its NBCQs Q_i are safe (resp., covered).

We assume the reader to be familiar with the chase procedure. The *disjunctive normal chase* is an extension of the chase. It differs to the chase in that an application of a DNTGD σ to a database D via a homomorphism $h\colon body^+(\sigma) \rightarrow D$ (that maps atoms $\beta \in body^-(\sigma)$ to non-atoms in D) generates more than one possible successor database—one database $D \cup \{h(\psi)\}$ for each disjunct $\psi \in head(\sigma)$. The set of all databases obtained using this version of the chase is denoted by $chase(D, \Sigma)$. Note that DNTGDs may be applied several times with the same homomorphism, to introduce several of the head atoms.

Canonical Model Semantics. A *stratification* of a set Σ of disjunctive normal TGDs on $\mathcal{R}$ is a mapping $\mu\colon \mathcal{R} \rightarrow \{0, 1, \ldots, k\}$ such that for all $\sigma \in \Sigma$:

- for all positive literals $R(\bar{u})$ in the body of σ, and for all atoms $R'(\bar{u}')$ in the head of σ, we have $\mu(R) \leqslant \mu(R')$;

- for all negative literals $\neg R(\bar{u})$ in the body of σ, and for all atoms $R'(\bar{u}')$ in the head of σ, we have $\mu(R) < \mu(R')$.

We call k the *length* of μ. For each $i \in \{0, 1, \ldots, k\}$, let Σ_i be the set of all $\sigma \in \Sigma$ such that i is the minimum of $\mu(R)$, where R ranges over all the predicates in the head of σ. We say that Σ is *stratified* if there is a stratification of Σ.

Definition 4 *Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a set of DNTGDs on $\mathcal{R}$, and Q a UNBCQ over $\mathcal{R}$. Suppose that Σ has a stratification of length k. Define:*

- $\mathcal{S}^0(D, \Sigma) := \{D\}$;
- $\mathcal{S}^{i+1}(D, \Sigma) := \bigcup_{S \in \mathcal{S}^i(D,\Sigma)} chase^S(S, \Sigma_i)$, *if* $i \leqslant k$;
- $\mathcal{S}(D, \Sigma) := \mathcal{S}^{k+1}(D, \Sigma)$.

We write $D \cup \Sigma \models_{\mathsf{strat}} Q$ iff $S \models Q$ for all $S \in \mathcal{S}(D, \Sigma)$.

5 Query Answering under Disjunctive Rules with Stratified Negation

This section presents our results on answering queries over disjunctive guarded Datalog$^{\pm}$ programs with stratified negation. Our main result is:

Theorem 5 *Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a stratified set of guarded DNTGDs on $\mathcal{R}$, and Q a covered UNBCQ over $\mathcal{R}$. Deciding $D \cup \Sigma \models_{\mathsf{strat}} Q$ has the following complexity:*

1. 2-EXPTIME-*complete in general;*
2. EXPTIME-*complete in case the maximum arity w of the relation symbols in $\mathcal{R}$ is fixed;*
3. coNP-*complete in case $|\mathcal{R}|$, w, and the number of atoms in Q are fixed.*

Thus, extending guarded Datalog$^{\pm}$ by the feature of disjunction and stratified negation—even allowing negative literals in the query—does not change the complexity of query answering in the general and in the bounded arity case, while it increases the data complexity. The data complexity is coNP already for the extension of guarded Datalog$^{\pm}$ by disjunction (Bourhis, Morak, and Pieris 2013), which proves coNP-hardness in Theorem 5. The other hardness results of Theorem 5 follow from answering BCQs under guarded TGDs being 2-EXPTIME-hard in general and EXPTIME-hard for bounded arities (Calì, Gottlob, and Kifer 2013).

For the upper bounds, we devise an algorithm to answer UNBCQs over stratified sets of guarded DNTGDs. The algorithm extends the QCHECK algorithm in (Calì, Gottlob, and Kifer 2013), but requires significantly new ideas. Our exposition follows the one of QCHECK in that paper in that we first explain how to decide entailment or non-entailment of atoms, and then describe the general algorithm.

5.1 Checking Entailment of Ground Atoms

One of the first steps of our algorithm is to decide if certain ground atoms are true in some "stratified model" or false in all "stratified models" $S \in \mathcal{S}(D, \Sigma)$. Here, an atom α is *ground* (relative to a database D, which, however, will always be clear from the context) iff $dom(\alpha) \subseteq dom(D)$.

Proposition 6 below states that (non-)entailment of ground atoms can be checked within the desired complexity bounds. For stating and proving it, we need the following definitions. For atoms α, let $dom(\alpha) := dom(\{\alpha\})$. The databases D_1 and D_2 are *consistent* iff, for all α with $dom(\alpha) \subseteq dom(D_1) \cap dom(D_2)$, we have that $\alpha \in D_1$ iff $\alpha \in D_2$. We say that D_2 *embeds* D_1 iff $D_1 \subseteq D_2$, and D_1 and D_2 are consistent. Let $\mathcal{S}^*(D, \Sigma)$ be the set of all $S \in \mathcal{S}(D, \Sigma)$ that embed D.

Proposition 6 *Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a stratified set of guarded DNTGDs on $\mathcal{R}$, and α a ground atom over $\mathcal{R}$. Deciding whether some model in $\mathcal{S}^*(D, \Sigma)$ contains α, or if none of the models in $\mathcal{S}^*(D, \Sigma)$ contains α is in:*

- 2-EXPTIME *in general;*
- EXPTIME *in case the maximum arity w of a predicate in $\mathcal{R}$ is bounded;*
- PTIME *in case w and $|\mathcal{R}|$ are bounded.*

Proof (Sketch). We extend the ACHECK algorithm from (Calì, Gottlob, and Kifer 2013). ACHECK checks if an atom (atomic BCQ, in general) α is entailed by a database D and a set Σ of guarded TGDs. To do this, it guesses a sequence $\alpha_1, \ldots, \alpha_n$ of atoms such that $\alpha_1 \in D$, $\alpha_n = \alpha$, and each α_i, $i > 1$, can be obtained by a TGD σ_i in Σ whose guard is matched to α_{i-1}. To ensure that the *side atoms* β of each σ_i (i.e., the non-guard atoms in the body of σ_i required to "fire" σ_i) can be derived as well, it launches side computations that check (similarly as for α) if there is a similar path for β etc.

Our algorithm, called $\mathsf{ACHECK}_{\vee,\mathsf{strat}}$, generalizes this approach. The main difficulty is that under a stratified set Σ of guarded DNTGDs, there is not a single model (namely, the chase model) to explore, but many models $S \in \mathcal{S}^*(D, \Sigma)$. However, every such model can be constructed on the fly by applying the DNTGDs in Σ, which is enough to search for a model that contains α, or—by duality—to check that no model contains α. To be a bit more precise, each disjunct in the head of a DNTGD represents a possible way of extending the part of a model derived so far. Thus, when faced with a disjunction in the head of a DNTGD, it suffices to guess the disjunct β that leads to a model S containing α (and the set of all atoms in such an S with $dom(\gamma) \subseteq dom(\beta)$); this guess is verified by appropriate subcomputations. Negative literals in the bodies of DNTGDs pose no problem, since negation is stratified in Σ. We only have to ensure that the stratification is respected when applying the DNTGDs. For details, see the extended paper (Gottlob et al. 2014).

As in (Calì, Gottlob, and Kifer 2013), it is possible to reduce the space required to represent the configurations of $\mathsf{ACHECK}_{\vee,\mathsf{strat}}$ to $|\mathcal{R}| \cdot 2^{O(w)} \cdot \log |D|$, where w is the maximum arity of a predicate in $\mathcal{R}$, via *canonization*. As alternating logarithmic (resp., polynomial, exponential) space corresponds to polynomial (resp., exponential, doubly exponential) time, this yields the complexity bounds. □

5.2 Main Algorithm

We now complete the proof of Theorem 5 by proving the corresponding upper bounds.

Proposition 7 *Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a stratified set of guarded DNTGDs on $\mathcal{R}$, and Q a covered UNBCQ over $\mathcal{R}$. Deciding $D \cup \Sigma \models_{\mathsf{strat}} Q$ can be done within the upper bounds indicated in Theorem 5.*

Proof (Sketch). First, w.l.o.g., Q is a union of BCQs. Indeed, if $\neg\beta = \neg R(x_1, \ldots, x_k)$ occurs in Q, and α is the positive literal in Q covering β, then $\neg\beta$ can be simulated by $\alpha \wedge \neg\beta \rightarrow \bar{R}(x_1, \ldots, x_k)$, where $\bar{R}$ is a fresh predicate. This increases the length of a stratification of Σ only by one. In Q, we replace $\neg\beta$ by the atom $\bar{R}(x_1, \ldots, x_k)$.

To obtain the complexity bounds, we extend the QCHECK algorithm in (Calì, Gottlob, and Kifer 2013). Handling stratified DNTGDs requires significantly new ideas, though. The original QCHECK algorithm checks if a BCQ is entailed by a database and a set of (weakly) guarded TGDs. It starts by computing the extension $\hat{D}$ of D with all ground atoms entailed by D and the TGDs. Next, it guesses a *squid decomposition* δ of Q. Intuitively, a squid decomposition of Q describes how Q may be mapped to a model S in $\mathcal{S}(D, \Sigma)$. For our purposes, a squid decomposition of Q may be defined as a tuple (Q^+, h, H, T, V), where:

- Q^+ is a *cover* of Q (a BCQ that contains all atoms of Q and at most $|Q|$ other atoms);
- $h\colon var(Q^+) \rightarrow var(Q^+)$,
- $H \subseteq Q^+$ and $T := Q^+ \setminus H$ are the *head* and *tentacles*,
- and T is *acyclic* (i.e., T has a join tree).

It can be shown (Calì, Gottlob, and Kifer 2013) that Q is true in the chase model iff there is a squid decomposition whose head can be mapped to the ground part of the database, and whose tentacles can be mapped to the non-ground part. Thus, the QCHECK algorithm also guesses a homomorphism μ from the head of δ to $\hat{D}$. In the rest of the computation, it checks that μ can be extended to a homomorphism from the tentacles to the result of the chase with D and the TGDs. In our case, we do not have a single "chase model", but many such models $S \in \mathcal{S}(D, \Sigma)$ into which we have to map Q. In particular, there are many extensions $\hat{D}$ of D with ground atoms such that some $S \in \mathcal{S}(D, \Sigma)$ embeds $\hat{D}$. Our algorithm thus starts by universally branching over all such extensions $\hat{D}$ of D with ground atoms, and checks that there is a model $S \in \mathcal{S}(D, \Sigma)$ that embeds $\hat{D}$. The latter can be done within the desired complexity bounds by using the algorithm as guaranteed by Proposition 6. In a parallel subcomputation, the algorithm then checks that Q is true in all models $S \in \mathcal{S}^*(\hat{D}, \Sigma)$. Clearly, this algorithm decides $D \cup \Sigma \models_{\mathsf{strat}} Q$ (i.e., if Q is true in all $S \in \mathcal{S}(D, \Sigma)$).

It remains to explain how truth of Q in all $S \in \mathcal{S}^*(\hat{D}, \Sigma)$ can be decided within the desired complexity bounds. To this end, we extend the TCHECK algorithm in (Calì, Gottlob, and Kifer 2013). We highlight only the main changes to TCHECK; (Gottlob et al. 2014) contains a more detailed description. For details regarding TCHECK, see the proof of Theorem 5.9 in (Calì, Gottlob, and Kifer 2013).

The idea of TCHECK is as follows. Given a squid decomposition δ (for which we have found a homomorphism h from its head to the ground part of the chase in the initialization phase of QCHECK), a maximal subtree t of the tentacles of δ, the root v of t, and the assignment h restricted to the *exported variables* of v (i.e., those variables that occur in v but also outside the subtree below v), TCHECK tries to extend the assignment to a homomorphism from the subtree rooted at v to the chase model by deriving an atom where v can be matched to, then deriving the atoms where the children of v can be matched to, and so on. In our case, each model $S \in \mathcal{S}(\hat{D}, \Sigma)$ might satisfy Q via a different embedding or squid decomposition of Q. Thus, rather than maintaining information about the progress of extending the initial homomorphism to a single subtree of a squid decomposition's tentacles throughout the computation, we store information about all possible squid decompositions of Q, and the parts of the squid decompositions that still need to be matched (i.e., have not been mapped into the part of the model constructed so far). Once we are in a state where one squid decomposition is complete (i.e., no part of it is left to be checked), we accept. Thus, if all computation branches accept, we have successfully checked that each $S \in \mathcal{S}^*(\hat{D}, \Sigma)$ satisfies Q. □

6 Simulating Normal Datalog$^{\pm}$ under Stable Models with Stratified Disjunctive Rules

The following theorem shows that NBCQ answering over a database and a finite set of guarded NTGDs under the stable model semantics can be translated in polynomial time into NBCQ answering over the same database and a finite set of guarded DNTGDs with stratified negation.

Theorem 8 *Let D be a database for some schema $\mathcal{R}$, let Σ be a finite set of guarded NTGDs over $\mathcal{R}$, and let Q be an NBCQ over $\mathcal{R}$. Then, Σ and Q can be translated in polynomial time into a stratified finite set of guarded DNTGDs Σ' and an NBCQ Q' such that Q is true in all stable models of D and Σ iff Q' is true in all canonical models of D and Σ'.*

Proof. We define the stratified finite set of guarded DNTGDs $\Sigma' = \Sigma'_1 \cup \Sigma'_2 \cup \Sigma'_3 \cup \Sigma'_4$ and the NBCQ Q' as follows:
• Σ'_1 generates the potential atoms that may occur in a stable model for D and Σ. To this end, it "executes" the program $P^+ = (D \cup \Sigma^f)^+$. For each $R \in \mathcal{R}$, let R^* be a fresh predicate of the same arity as R. Then, Σ'_1 consists of all TGDs that are obtained from an NTGD in Σ by dropping all negative literals from its body, and replacing each predicate R by R^*. Furthermore, for each k-ary predicate $R \in \mathcal{R}$, the set Σ'_1 contains the TGD $R(x_1, \ldots, x_k) \rightarrow R^*(x_1, \ldots, x_k)$.
• Σ'_2 "guesses" a partition of the set of all atoms derivable from P^+ into two parts. For each predicate $R \in \mathcal{R}$, let R^+ and R^- be fresh predicates of the same arity as R. The idea is that R^+ and R^--atoms are those belonging to the first and second part of the partition, respectively. To "guess" the partition, for each k-ary predicate $R \in \mathcal{R}$, we add the following disjunctive TGD and negative constraint to Σ'_2:

$$R^*(x_1, \ldots, x_k) \rightarrow R^+(x_1, \ldots, x_k) \vee R^-(x_1, \ldots, x_k),$$
$$R^+(x_1, \ldots, x_k) \wedge R^-(x_1, \ldots, x_k) \rightarrow \bot.$$

• Σ'_3 generates all consequences of the program P^S, where S is the set of all the R^+-atoms (with R^+ renamed to R). More precisely, for each predicate $R \in \mathcal{R}$, let $\hat{R}$ be a predicate of the same arity as R. Consider an NTGD in Σ:

$$R_1(\mathbf{x}_1) \wedge \cdots \wedge R_k(\mathbf{x}_k) \wedge \neg S_1(\mathbf{y}_1) \wedge \cdots \wedge \neg S_\ell(\mathbf{y}_\ell) \rightarrow \exists \mathbf{z}\, T(\mathbf{u}).$$

• Then, Σ'_3 contains the following NTGD:

$$\hat{R}_1(\mathbf{x}_1) \wedge \cdots \wedge \hat{R}_k(\mathbf{x}_k) \wedge S_1^-(\mathbf{y}_1) \wedge \cdots \wedge S_\ell^-(\mathbf{y}_\ell) \rightarrow \exists \mathbf{z}\, \hat{T}(\mathbf{u}).$$

Furthermore, for each k-ary predicate $R \in \mathcal{R}$, the set Σ'_3 contains the TGD $R(x_1, \ldots, x_k) \rightarrow \hat{R}(x_1, \ldots, x_k)$.

• Σ'_4 verifies that the consequences of P^S (represented by all the $\hat{R}$-atoms) coincide with the set S (represented by all the R^+-atoms). To this end, for each k-ary predicate $R \in \mathcal{R}$, it contains the following negative constraints:

$$\hat{R}(x_1, \ldots, x_k) \wedge R^-(x_1, \ldots, x_k) \rightarrow \bot \qquad (\text{“}\hat{R} \subseteq R^+\text{”})$$
$$R^+(x_1, \ldots, x_k) \wedge \neg \hat{R}(x_1, \ldots, x_k) \rightarrow \bot \qquad (\text{“}R^+ \subseteq \hat{R}\text{”}).$$

• Finally, Q' is obtained from Q by replacing every positive (resp., negative) literal $R(\mathbf{x})$ (resp., $\neg R(\mathbf{x})$) by $R^+(\mathbf{x})$ (resp., $\neg R^+(\mathbf{x})$).

We now show that Q is false in some stable model for D and Σ iff Q' is false in some canonical model for D and Σ'. We here consider only the case that Q is a BCQ; the line of argumentation can easily be generalized to NBCQs.

“$\Rightarrow$” Let S be a stable model for D and Σ such that $S \not\models Q$. Let T be the least fixpoint of P^+. Define:

$$S' = \{R^*(\mathbf{a}) \mid R(\mathbf{a}) \in T\} \cup \{R^+(\mathbf{a}) \mid R(\mathbf{a}) \in S\} \cup \{R^-(\mathbf{a}) \mid R(\mathbf{a}) \in T \setminus S\} \cup \{\hat{R}(\mathbf{a}) \mid R(\mathbf{a}) \in S\}.$$

It is then not difficult to verify that S' is a model of D and Σ' under stratified semantics, and $S' \not\models Q'$ (since $\{\mathbf{a} \mid R(\mathbf{a}) \in S\} = \{\mathbf{a} \mid R^+(\mathbf{a}) \in S'\}$).

“$\Leftarrow$” Let S' be a model for D and Σ' obtained using the chase (for disjunctive TGDs with stratified negation) such that $S' \not\models Q'$. For a predicate R and a database D, let $D(R)$ be the set of all the tuples $\mathbf{a}$ with $R(\mathbf{a}) \in D$. Then, $T := \{R(\mathbf{a}) \mid \mathbf{a} \in S'(R^*)\}$ is the unique minimal model for P^+, and $\{S'(R^+), S'(R^-)\}$ forms a partition of $S'(R^*)$. Moreover, the NTGDs in Σ'_4 enforce $S'(\hat{R}) \cap S'(R^-) = \varnothing$ and $S'(R^+) \subseteq S'(\hat{R})$. Since $S'(\hat{R}) \subseteq S'(R^*)$ (only a subset of the rules which generate R^* are triggered), and $S'(R^-) = S'(R^*) \setminus S'(R^+)$, we thus have $S'(\hat{R}) = S'(R^+)$. By the construction of the rules Σ'_3, it follows that $S := \{R(\mathbf{a}) \mid \mathbf{a} \in S'(R^+)\}$ is a stable model for D and Σ. Moreover, since $S' \not\models Q'$, and $S(R) = S'(R^+)$, we have $S \not\models Q$. □

The following example illustrates the above translation.

Example 3 Consider the database $D = \{P(a)\}$ and the following finite set of NTGDs Σ:

$$P(x) \wedge \neg R(x) \rightarrow Q(x), \qquad P(x) \wedge \neg Q(x) \rightarrow R(x).$$

Then, Σ' consists of the following DNTGDs (note that some of the rules (like $R^*(x) \rightarrow R^*(x)$) are omitted, because they would add nothing to Σ' under D):

$$P(x) \rightarrow P^*(x), \quad P^*(x) \rightarrow Q^*(x), \quad P^*(x) \rightarrow R^*(x),$$
$$P^*(x) \rightarrow P^+(x) \vee P^-(x), \qquad P^+(x) \wedge P^-(x) \rightarrow \bot,$$
$$Q^*(x) \rightarrow Q^+(x) \vee Q^-(x), \qquad Q^+(x) \wedge Q^-(x) \rightarrow \bot,$$
$$R^*(x) \rightarrow R^+(x) \vee R^-(x), \qquad R^+(x) \wedge R^-(x) \rightarrow \bot,$$
$$P(x) \rightarrow \hat{P}(x),$$
$$\hat{P}(x) \wedge R^-(x) \rightarrow \hat{Q}(x), \quad \hat{P}(x) \wedge Q^-(x) \rightarrow \hat{R}(x),$$
$$\hat{P}(x) \wedge P^-(x) \rightarrow \bot, \qquad P^+(x) \wedge \neg \hat{P}(x) \rightarrow \bot,$$
$$\hat{Q}(x) \wedge Q^-(x) \rightarrow \bot, \qquad Q^+(x) \wedge \neg \hat{Q}(x) \rightarrow \bot,$$
$$\hat{R}(x) \wedge R^-(x) \rightarrow \bot, \qquad R^+(x) \wedge \neg \hat{R}(x) \rightarrow \bot.$$

To obtain a proper stratified program, we could replace falsum ($\bot$) in the head of $P^+(x) \wedge \neg \hat{P}(x) \rightarrow \bot$ and the corresponding rules for Q and R by an atom $F(x)$ over a fresh predicate F, and put $F(x)$ as a disjunct to our query. Then, F would be in the upper stratum, and all other predicates would be in the lower stratum. ■

The following complexity results for UNBCQ answering in guarded normal Datalog$^\pm$ under the stable model semantics are an immediate corollary of Theorems 8 and 5.

Corollary 9 *Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a finite set of guarded NTGDs on $\mathcal{R}$, and Q a covered UNBCQ over $\mathcal{R}$. Then, deciding whether Q is true in all stable models of D and Σ has the following complexity:*

1. 2-EXPTIME-*complete in general;*
2. EXPTIME-*complete in case the maximum arity w of the relation symbols in $\mathcal{R}$ is fixed;*
3. coNP-*complete in case $|\mathcal{R}|$, w, and the number of atoms in Q are fixed.*

7 Stable Model Semantics for DLs

We now discuss how our decidability results for the stable model semantics can directly be applied to obtain extensions of DLs with non-monotonic negation. The proposed logics are essentially the same as in (Gottlob et al. 2012a), but the crucial difference is that non-monotonic negation is now interpreted via the stable model semantics. This can lead to better query answers, as Example 5 below demonstrates.

We extend *DL-Lite*$_\mathcal{R}$ (which is underlying the OWL 2 QL profile) and *DL-Lite*$_{\mathcal{R},\sqcap}$ (Calvanese et al. 2007; Poggi et al. 2008), and $\mathcal{ELHI}$ (Baader, Brandt, and Lutz 2005) with nonmonotonic negation under the stable model semantics.

Definition 10 Recall that a *DL-Lite*$_{\mathcal{R},\sqcap}$ knowledge base consists of a pair $(\mathcal{T}, \mathcal{A})$, where the TBox $\mathcal{T}$ is a finite set of concept and role inclusion axioms $U_1 \sqcap \cdots \sqcap U_n \sqsubseteq V$, and the ABox $\mathcal{A}$ is a finite set of concept and role membership axioms $C(a)$ and $R(a, b)$, respectively. A *DL-Lite*$_{\mathcal{R},\sqcap,\text{not}}$ knowledge base $(\mathcal{T}, \mathcal{A})$ consists of a finite set of inclusion axioms $\mathcal{T}$ and a finite set of membership axioms $\mathcal{A}$, where:

- Any *DL-Lite*$_{\mathcal{R},\sqcap}$ inclusion axiom is a *DL-Lite*$_{\mathcal{R},\sqcap,\text{not}}$ inclusion axiom.
- If $U_1 \sqcap \cdots \sqcap U_n \sqsubseteq V$ and $U'_1 \sqcap \cdots \sqcap U'_m \sqsubseteq V$ with $n, m > 0$ are both either concept or role inclusion axioms in *DL-Lite*$_{\mathcal{R},\sqcap}$, and V is positive (i.e., not of the form $V = \neg V'$), then $U_1 \sqcap \cdots \sqcap U_n \sqcap \text{not}\, U'_1 \sqcap \cdots \sqcap \text{not}\, U'_m \sqsubseteq V$ is

a *DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$ concept or role inclusion axiom, respectively. Here, the U_i's and U_i''s contain no conjunction, and $\mathrm{not}\, U_i'$ denotes the negation as failure (as opposed to the classical negation "$\neg$" in *DL-Lite*).

- For any concept A, any role R, and any individuals a, b, the expressions $A(a)$ and $R(a, b)$ are concept and role membership axioms, respectively.

A *DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$ knowledge base $(\mathcal{T}, \mathcal{A})$ is a *DL-Lite*$_{\mathcal{R},\mathrm{not}}$ knowledge base iff all inclusion axioms in $\mathcal{T}$ contain precisely one positive atom on the left-hand side.

Finally, we define $\mathcal{ELHI}_{\mathrm{not}}$ as the extension of $\mathcal{ELHI}$ that allows formulas of the form $\mathrm{not}\, C$ for atomic concepts $C = A$ and for concepts $C = \exists R.B$ to occur in top-level conjunctions on the left-hand side of concept inclusions and formulas $\mathrm{not}R$ and $\mathrm{not}R^-$ for any role R on the left hand side of role inclusions (in addition to at least one positive concept or one positive role, respectively, to ensure guardedness). ■

The semantics of *DL-Lite*$_{\mathcal{R},\mathrm{not}}$ (resp., *DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$) is defined by translating a given *DL-Lite*$_{\mathcal{R},\mathrm{not}}$ (resp., *DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$) knowledge base KB into a normal guarded Datalog$^{\pm}$ program P_{KB} and by computing the stable model semantics of P_{KB}. More details on the translation into normal Datalog$^{\pm}$ can be found in (Gottlob et al. 2012a). Similarly it is straightforward to translate $\mathcal{ELHI}_{\mathrm{not}}$ knowledge bases into normal guarded Datalog$^{\pm}$-programs in order to obtain a stable semantics for $\mathcal{ELHI}_{\mathrm{not}}$.

The following example demonstrates how Example 1 from the introduction can be formalized in $\mathcal{ELHI}_{\mathrm{not}}$.

Example 4

$$\begin{aligned} \mathrm{Person} \sqcap \mathrm{notEven} &\sqsubseteq \mathrm{Odd} \\ \mathrm{Person} \sqcap \mathrm{notOdd} &\sqsubseteq \mathrm{Even} \\ \mathrm{Person} \sqcap \mathrm{Even} &\sqsubseteq \exists \mathrm{hasParent.Odd} \\ \mathrm{Person} \sqcap \mathrm{Odd} &\sqsubseteq \exists \mathrm{hasParent.Even}. \end{aligned}$$

If we consider the ABox $\{\mathrm{Person}(\mathrm{mary})\}$, we obtain two (infinite) stable models for the knowledge base that look very similar to the models obtained in Example 1, the two models correspond to the case in which $\mathrm{Odd}(\mathrm{mary})$ and to the case in which $\mathrm{Even}(\mathrm{mary})$ holds. ■

In the next example, the stable model semantics leads to better query answers than the well-founded semantics.

Example 5

$$\begin{aligned} \mathrm{Hotel} \sqcap \mathrm{Perfect} &\sqsubseteq \exists \mathrm{Beach} \sqcap \exists \mathrm{Pool} \\ \mathrm{FiveStar} &\sqsubseteq \mathrm{Hotel} \\ \mathrm{FiveStar} \sqcap \mathrm{not}\exists \mathrm{Pool} &\sqsubseteq \exists \mathrm{Beach} \\ \mathrm{FiveStar} \sqcap \mathrm{not}\exists \mathrm{Beach} &\sqsubseteq \exists \mathrm{Pool} \\ \exists \mathrm{Beach} &\sqsubseteq \exists \mathrm{SwimOpp} \\ \exists \mathrm{Pool} &\sqsubseteq \exists \mathrm{SwimOpp} \\ \mathrm{Hotel} \sqcap \exists \mathrm{SwimOpp} &\sqsubseteq \mathrm{Excellent}. \end{aligned}$$

Given the ABox $\{\mathrm{FiveStar}(\mathrm{ritz})\}$, the well-founded semantics for the knowledge base would only contain the atoms $\{\mathrm{FiveStar}(\mathrm{ritz}), \mathrm{Hotel}(\mathrm{ritz})\}$. In contrast to that, all stable models of the same knowledge base contain in addition the atoms $\exists \mathrm{SwimOpp}(\mathrm{ritz})$ and $\mathrm{Excellent}(\mathrm{ritz})$. This correctly reflects the fact that, while we are unable to know whether a given five star hotel has a swimming pool or a beach, we can be sure that one of these facts is true. ■

The decidability results for query answering relative to the stable model semantics can directly be applied to DLs.

Theorem 11 *Let $\mathcal{L}$ be DL-Lite*$_{\mathcal{R},\mathrm{not}}$*, DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$ *or $\mathcal{ELHI}_{\mathrm{not}}$. Then, given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a covered UNBCQ[2] Q, deciding whether Q is true under the stable model semantics of $\mathcal{K}$ is in EXPTIME (resp., coNP) in the combined (resp., data) complexity. For $\mathcal{ELHI}_{\mathrm{not}}$ the combined complexity is EXPTIME-complete. In all cases, the data complexity is coNP-complete.*

Proof (Sketch). The upper bounds for the complexity are an immediate consequence of Corollary 9. The lower bound for combined complexity of query answering over a $\mathcal{ELHI}_{\mathrm{not}}$ knowledge base is a consequence of the EXPTIME lower bound for subsumption checking in $\mathcal{ELI}$ (Baader, Brandt, and Lutz 2008, Theorem 3), as a subsumption $C \sqsubseteq D$ can be checked by answering the query $C \sqcap \mathrm{not}D$. The lower bound for the data complexity follows from a reduction of the UNSAT-problem; see (Gottlob et al. 2014). □

8 Summary and Outlook

This paper has provided the first fully general stable model semantics for guarded existential rules with negation. We proved that query answering is decidable under this semantics, and we determined the complexity of this problem for various settings. On the way to these results, we defined disjunctive rule sets under FO semantics augmented by stratified negation, and also studied the complexity of query answering based on this formalism, which is of interest in its own right. Finally, we showed how our results can be used for adorning well-known DLs with stable model negation.

For space reasons, many aspects and possible extensions of our work had to remain undiscussed and will be given more attention in the full paper. One feature that we already mentioned are EGDs. Another issue is the unique name assumption (UNA). Our approach for defining the SMS follows the standard semantics for logic programming with function symbols, where different Skolem terms are considered to be different elements that cannot be unified. In a precise sense, this means that our semantics applies the UNA not only to database constants, but also to invented null values, i.e., Skolem terms. As long as negation is not used, this does not matter. But with negation, this may become an issue in some cases. Reconsider Example 1 and let $\Sigma_c = \Sigma_a \cup \{$*Person(x), not Parent(x,x)*$\rightarrow ok\}$. Under our current semantics, $\{$*Person(mary)*$\} \cup \Sigma_c \models ok$. However, if one does not apply the UNA, then ok would not be derivable, because there would exist models in which, e.g., *mary = f(mary)*. We are able to define a version of the stable model semantics without the UNA, too, and we call this the *equality-friendly SMS*, in analogy to (Gottlob et al. 2012a). This semantics is defined in a totally analogous way to the one defined here, except that instead of using stratified GDNTGDs, we use guarded fixpoint logic. As will be shown in the full paper, decidability and the same complexity results hold.

[2] See (Gottlob et al. 2014) for a definition.

Ongoing work includes in particular the SMS for disjunctive guarded TGDs with negation (GDNTGDs). The semantics for this is, again, directly inherited from the corresponding SMS for disjunctive logic programming with function symbols. As this can be encoded into MSO in a similar way as the normal stable semantics, query answering is decidable. By a similar encoding into MSO, we have also obtained decidability for the SMS for disjunctive weakly guarded and weakly frontier-guarded TGDs with negation. We are currently still exploring the precise complexity of all these problems. We also plan to extend suitable versions of the interesting MKNF approach (Motik and Rosati 2010) by our techniques, so to achieve larger decidable classes.

Acknowledgments. This work was supported by the EPSRC under the grant EP/H051511/1 "ExODA", by the European Research Council under the FP7/2007-2013/ERC grant 246858 "DIADEM", and by a Yahoo! Research Fellowship. Georg Gottlob is a James Martin Senior Fellow, and also gratefully acknowledges a Royal Society Wolfson Research Merit Award. The work was carried out in the context of the James Martin Institute for the Future of Computing.

References

Alviano, M.; Faber, W.; Leone, N.; and Manna, M. 2012. Disjunctive Datalog with existential quantifiers: Semantics, decidability, and complexity issues. *TPLP* 12(4/5):701–718.

Arenas, M.; Gottlob, G.; and Pieris, A. 2014. Expressive languages for querying the Semantic Web. In *Proc. of PODS*. In press.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, 364–369.

Baader, F.; Brandt, S.; and Lutz, C. 2008. Pushing the $\mathcal{EL}$ envelope further. In *Proc. of OWLED*.

Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; and Salvat, E. 2009. Extending decidable cases for rules with existential variables. In *Proc. of IJCAI*, 677–682.

Baget, J.-F.; Mugnier, M.-L.; Rudolph, S.; and Thomazo, M. 2011. Walking the complexity lines for generalized guarded existential rules. In *Proc. of IJCAI, Volume II*, 712–717.

Baget, J.-F.; Leclère, M.; and Mugnier, M.-L. 2010. Walking the decidability line for rules with existential variables. In *Proc. of KR*, 466–476.

Beeri, C., and Vardi, M. Y. 1981. The implication problem for data dependencies. In *Proc. of ICALP*, 73–85.

Beeri, C., and Vardi, M. Y. 1984. A proof procedure for data dependencies. *J. ACM* 31(4):718–741.

Bourhis, P.; Morak, M.; and Pieris, A. 2013. The impact of disjunction on query answering under guarded-based existential rules. In *Proc. of IJCAI*, 796–802.

Calì, A.; Gottlob, G.; Lukasiewicz, T.; Marnette, B.; and Pieris, A. 2010. Datalog+/–: A family of logical knowledge representation and query languages for new applications. In *Proc. of LICS*, 228–242.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.* 48:115–174.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.* 14:57–83.

Calì, A.; Gottlob, G.; and Pieris, A. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.* 193:87–128.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning* 39(3):385–429.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv.* 33(3):374–425.

Eiter, T., and Simkus, M. 2010. FDNC: Decidable nonmonotonic disjunctive logic programs with function symbols. *ACM Trans. Comput. Log.* 11(2).

Eiter, T.; Ianni, G.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2008. Combining answer set programming with description logics for the Semantic Web. *Artif. Intell.* 172(12/13):1495–1539.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of ICLP/SLP*, 1070–1080.

Gottlob, G.; Hernich, A.; Kupke, C.; and Lukasiewicz, T. 2012a. Equality-friendly well-founded semantics and applications to description logics. In *Proc. of AAAI*, 757–764.

Gottlob, G.; Manna, M.; Morak, M.; and Pieris, A. 2012b. On the complexity of ontological reasoning under disjunctive existential rules. In *Proc. of MFCS*, 1–18.

Gottlob, G.; Hernich, A.; Kupke, C.; and Lukasiewicz, T. 2014. Stable model semantics for guarded existential rules and description logics. Technical report, Department of Computer Science, University of Oxford.

Gottlob, G.; Manna, M.; and Pieris, A. 2013. Combining decidability paradigms for existential rules. *TPLP* 13(4/5):877–892.

Hernich, A.; Kupke, C.; Lukasiewicz, T.; and Gottlob, G. 2013. Well-founded semantics for extended Datalog and ontological reasoning. In *Proc. of PODS*, 225–236.

Lukasiewicz, T. 2010. A novel combination of answer set programming with description logics for the Semantic Web. *IEEE Trans. Knowl. Data Eng.* 22(11):1577–1592.

Magka, D.; Krötzsch, M.; and Horrocks, I. 2013. Computing stable models for nonmonotonic existential rules. In *Proc. of IJCAI*, 1031–1038.

Maier, D.; Mendelzon, A. O.; and Sagiv, Y. 1979. Testing implications of data dependencies. *ACM Trans. Database Syst.* 4(4):455–469.

Motik, B., and Rosati, R. 2010. Reconciling description logics and rules. *J. ACM* 57(5).

Motik, B.; Horrocks, I.; and Sattler, U. 2009. Bridging the gap between OWL and relational databases. *J. Web Sem.* 7(2):74–89.

Poggi, A.; Lembo, D.; Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2008. Linking data to ontologies. *J. Data Semantics* 10:133–173.

Rabin, M. O. 1969. Decidability of second-order theories and automata on infinite trees. *Trans. Am. Math. Soc.* 141:1–35.

Subrahmanian, V. S. 1999. Nonmonotonic logic programming. *IEEE Trans. Knowl. Data Eng.* 11(1):143–152.

Thomazo, M. 2011. *Conjunctive Query Answering Under Existential Rules — Decidability, Complexity, and Algorithms*. Ph.D. Dissertation, Univ. of Montpellier, France.

Polynomial Combined Rewritings for Existential Rules

Georg Gottlob[1] and **Marco Manna**[2] and **Andreas Pieris**[1]

[1]Department of Computer Science, University of Oxford, UK
[2]Department of Mathematics and Computer Science, University of Calabria, Italy
{*georg.gottlob,andreas.pieris*}*@cs.ox.ac.uk*
manna@mat.unical.it

Abstract

We consider the scenario of ontology-based data access where a conjunctive query is evaluated against a database enriched with intensional knowledge via an ontology. It is generally accepted that true scalability of query answering in this setting can only be achieved by using standard relational database management systems (RDBMSs). An approach to query answering that enables the use of RDBMSs is the so-called polynomial combined approach. We investigate this approach for the main guarded- and sticky-based classes of existential rules, and we highlight the assumptions on the underlying schema which are sufficient for the polynomial combined first-order rewritability of those classes. To the best of our knowledge, this is the first work which explicitly studies the polynomial combined approach for existential rules.

1 Introduction

An ontology is an explicit specification of a conceptualization of an area of interest. One of the main applications of ontologies nowadays is in *ontology-based data access (OBDA)* (Poggi *et al.* 2008), where they are used to enrich the extensional data with intensional knowledge. In this setting, Description Logics (DLs) and rule-based formalisms such as existential rules are popular ontology languages, while conjunctive queries (CQs) are used as a vital querying tool. Therefore, efficient approaches to CQ answering over such languages are of great importance.

It is widely accepted that currently true scalability in OBDA can only be achieved by exploiting standard relational database management systems (RDBMSs), which provide mature and efficient technology for answering queries. A significant step forward in this direction was the introduction of the DL-Lite family of DLs (Calvanese *et al.* 2007; Poggi *et al.* 2008) — the logical underpinning of the OWL 2 QL profile of OWL 2 — and the proposal of the first practical approach to OBDA via query rewriting. Most of the languages of the DL-Lite family have been designed with the aim of being *first-order rewritable*, namely a TBox $\mathcal{T}$ can be incorporated together with a given CQ q into a first-order query $q_{\mathcal{T}}$ such that, for every ABox $\mathcal{A}$, $q_{\mathcal{T}}$ evaluated over $\mathcal{A}$ yields exactly the same result as q evaluated against $\mathcal{A}$ and $\mathcal{T}$. Clearly, $q_{\mathcal{T}}$ can be translated into a standard SQL query, and then submitted to the RDBMS holding $\mathcal{A}$ (as a relational database), where it is evaluated and optimized in the usual way.

Although first-order rewritability is a most desirable property, it suffers from two serious shortcomings: *(i)* rewriting algorithms, in general, generate from a reasonably sized CQ an exponentially sized SQL query — even for simple cases such as OWL 2 QL this blowup is unavoidable (Kikot *et al.* 2012) — which can be prohibitive for efficient execution by an RDBMS, and *(ii)* it applies only to lightweight ontology languages for which the data complexity of CQ answering is below LOGSPACE, and thus useful formalisms with PTIME-hard data complexity are excluded a priori.

A more refined approach to query answering, which overcomes the inherent limitations of first-order rewritability, is the *polynomial combined approach* (Lutz *et al.* 2009), that allows the encoding of the consequences of the ontology in the given database (or ABox). This approach applies to languages which are *polynomially combined first-order rewritable*, i.e., for every CQ q, database D, and ontology Σ, it is possible to rewrite in *polynomial time (i)* q and Σ into a first-order query q_Σ, and *(ii)* D and Σ into a database D_Σ, in such a way that the answer to q_Σ over D_Σ is the same as the answer to q over D and Σ.

The polynomial combined approach has been successfully applied to the DL $\mathcal{ELH}^{dr}_{\bot}$ (Lutz *et al.* 2009), i.e., the extension of $\mathcal{EL}$ with the inconsistent concept, role inclusions, and domain and range restrictions, and also to DL-Lite$^{\mathcal{N}}_{horn}$ (Kontchakov *et al.* 2010; 2011), one the most commonly used DL-Lite formalisms without role inclusions. Notice that DL-Lite$^{(\mathcal{HN})}_{horn}$, the extension DL-Lite$^{\mathcal{N}}_{horn}$ with role hierarchies, has been also considered in (Kontchakov *et al.* 2010) but, while the new database can be constructed in polynomial time, the rewritten query is of exponential size in the number of roles occurring in the ontology.

Although the polynomial combined approach has been already applied to several DLs, this is not the case for existential (a.k.a. Datalog$^{\exists}$) rules of the form $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow \exists \mathbf{Z} p(\mathbf{X}, \mathbf{Z})$. Notice that the investigation of expressive fragments of existential rules that can be employed as an alternative way to represent ontologies is currently a field of intense research (Baget *et al.* 2011a; 2011b; Krötzsch and Rudolph

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Size	Arity	L	G	S	T
∞	∞	?	$\times$	$[\times]$	$\times$
∞	$d \geqslant 2$	✓	$[\times]$	✓*	$[\times]$
∞	$d \leqslant 1$	✓	✓*	✓*	✓*
$c \geqslant 1$	∞	?	$\times^\star$	$[\times]^\star$	$\times^\star$
$c \geqslant 1$	$d \geqslant 0$	✓	✓*	✓	✓*

Table 1: Polynomial combined first-order rewritablity of (L)inear, (G)uarded, (S)ticky and (T)ame rules. The symbol ∞ (resp., $c \geqslant i$ and $d \geqslant i$) asserts that the parameter under consideration is unbounded (resp., bounded by the constants c and d). The expression $d \leqslant 1$ implies that the arity is either 0 or 1. The symbol ? refers to an open problem. The symbol ✓(resp., $\times$) refers to a positive (resp., negative) case. The symbol $[\times]$ refers to a negative case, assuming that PSPACE $\neq$ EXPTIME. Our results are marked with $\star$.

2011; Calì *et al.* 2012a; 2012b; 2013; Leone *et al.* 2012). The only known cases of existential rules which are polynomially combined first-order rewritable are the class of *linear* rules when the arity of the schema is bounded, and the class of *sticky* rules when both the size and the arity of the schema are bounded. Those results are implicit in (Gottlob and Schwentick 2012); however, the goal of that paper was not the investigation of the polynomial combined approach for linear and sticky rules, but the construction of a non-recursive Datalog rewriting of polynomial size.

To the best of our knowledge, this is the first work which explicitly studies the combined approach for existential rules. This problem is considered, by the KR community, an interesting research direction that should be investigated (Kontchakov *et al.* 2011). Towards this direction, we explore guarded- and sticky-based classes of existential rules. Both guardedness and stickiness are well-accepted paradigms. On the one hand, guarded rules form a robust language which captures important DLs such as DL-Lite and $\mathcal{EL}$ (Calì *et al.* 2012a) — a rule is *guarded* if it has a body-atom, called guard, which contains all the body-variables. On the other hand, sticky rules allow for joins in rule-bodies which are not always expressible via guarded rules, and they are able to capture well-known data modeling constructs such as inclusion and multivalued dependencies (Calì *et al.* 2012b). Recently, a class called *tame*, which combines guardedness and stickiness by taming the interaction between the sticky rules and the guard atoms occurring in guarded rules, has been proposed (Gottlob *et al.* 2013).

It is well-known that the problem of evaluating first-order queries over a database is feasible in polynomial space. Therefore, the polynomial combined approach is applicable only to ontology languages for which the combined complexity of CQ answering is at most PSPACE (assuming that PSPACE $\neq$ EXPTIME). Since CQ answering under guarded and tame rules is 2EXPTIME-hard, and under sticky rules is EXPTIME-hard, it is interesting to investigate which assumptions on the underlying schema are sufficient for the polynomial combined first-order rewritability of the above classes. The techniques employed for $\mathcal{ELH}_\bot^{dr}$ and DL-Lite$_{horn}^{\mathcal{N}}$, although quite insightful, leverage specificities of DLs, such as the limit to unary and binary predicates and the special form of DL axioms, so that is not clear how they can be extended to general rule-based languages. Thus, we had to come up with novel techniques beyond the state of the art. Our contributions can be summarized as follows:

- We first show that guarded rules are, in general, not polynomially combined first-order rewritable, even if the underlying schema contains a single predicate. The same holds even if we concentrate on unary and binary predicates (unless PSPACE = EXPTIME). However, guarded rules are polynomially combined first-order rewritable if we consider only unary predicates, or when the size and the arity of the underlying schema are bounded.
- We show that sticky rules are, in general, not polynomially combined first-order rewritable, even if the underlying schema contains a single predicate (unless PSPACE = EXPTIME). Sticky rules are polynomially combined first-order rewritable if we consider schemas of bounded arity.
- Finally, we show that the results for guarded rules described above can be extended to tame rules.

In summary, we form an (almost) complete picture, depicted in Table 1, of the polynomial combined first-order rewritability for the main guarded- and sticky-based classes of existential rules. Notice that the case of linear rules, even when the schema contains a single predicate, is still open. We believe that linear rules are polynomially combined first-order rewritable, and we give some evidence for this in Section 7. At this point, let us clarify that guarded and tame rules are not first-order rewritable since the data complexity of CQ answering under those classes is PTIME-hard, even when the size and the arity of the underlying schema are bounded. Moreover, sticky rules are strictly more expressive than OWL 2 QL, and thus they are not polynomially first-order rewritable, even if the arity is fixed. Hence, our results enlarge the family of ontology languages which are suitable for OBDA purposes. Full proofs can be found in *https://www.mat.unical.it/datalog-exists/pub/gmp14.pdf*.

2 Preliminaries

Technical Definitions. We define the following pairwise disjoint (infinite countable) sets: a set $\mathbf{C}$ of *constants*, a set $\mathbf{N}$ of *labeled nulls*, and a set $\mathbf{V}$ of regular *variables*. A *term* t is a constant, null, or variable. An *atom* has the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. Let $arity(p)$ be the arity of the predicate p. For an atom $\underline{a}$, we denote $dom(\underline{a})$ and $var(\underline{a})$ the set of its terms and the set of its variables, respectively; these extend to sets of atoms. Conjunctions of atoms are often identified with the sets of their atoms. An *instance* I is a (possibly infinite) set of atoms $p(\mathbf{t})$, where $\mathbf{t}$ is a tuple of constants and nulls. Given a set of predicates $\mathcal{R}$, let $I_{|\mathcal{R}} = \{p(\mathbf{t}) \in I \mid p \in \mathcal{R}\}$, i.e., the restriction of I on $\mathcal{R}$. A *database* D is a finite instance such that $dom(D) \subset \mathbf{C}$. Given a set T of terms, we denote by $base(T, \mathcal{R})$ the set of atoms that can be formed using terms of T and predicates of $\mathcal{R}$. A *homomorphism* is a substitution $h : \mathbf{C} \cup \mathbf{N} \cup \mathbf{V} \to \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$ which is the identity on $\mathbf{C}$. Two sets of atoms A, A' are *homomorphically equivalent*, written $A \longleftrightarrow A'$, if there exist homomorphisms h and h' such that $h(A) \subseteq A'$ and $h'(A') \subseteq A$. We assume the reader is familiar with *conjunctive queries (CQs)*. The

answer to q over an instance I is denoted $q(I)$. A Boolean CQ (BCQ) q has a positive answer over I, denoted $I \models q$, if $() \in q(I)$.

A *Datalog*$^\exists$ *rule* (or simply *rule*) σ is a constant-free first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, p(\mathbf{X}, \mathbf{Z})$, where $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z} \subset \mathbf{V}$, $\varphi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms, and $p(\mathbf{X}, \mathbf{Z})$ is a single atom; $\varphi(\mathbf{X}, \mathbf{Y})$ is the *body* of σ, denoted $body(\sigma)$, while $p(\mathbf{X}, \mathbf{Z})$ is the *head* of σ, denoted $head(\sigma)$. Given a set Σ of rules, $sch(\Sigma)$ is the set of predicates occurring in Σ, and $arity(\Sigma)$ is defined as $\max_{p \in sch(\Sigma)}\{arity(p)\}$. For brevity, we will omit the universal quantifiers in front of rules. An instance I satisfies σ, written $I \models \sigma$, if the following holds: whenever there exists a homomorphism h such that $h(\varphi(\mathbf{X}, \mathbf{Y})) \subseteq I$, then there exists $h' \supseteq h_{|\mathbf{X}}$, where $h_{|\mathbf{X}}$ is the restriction of h on $\mathbf{X}$, such that $h'(p(\mathbf{X}, \mathbf{Z})) \in I$; I satisfies a set Σ of rules, denoted $I \models \Sigma$, if I satisfies each σ of Σ. The *models* of a database D and a set Σ of rules, denoted $mods(D, \Sigma)$, is the set of instances $\{I \mid I \supseteq D \text{ and } I \models \Sigma\}$. The *answer* to a CQ q w.r.t. D and Σ is defined as the set of tuples $ans(q, D, \Sigma) = \bigcap_{I \in mods(D,\Sigma)}\{\mathbf{t} \mid \mathbf{t} \in q(I)\}$. The answer to a BCQ q is *positive*, denoted $D \cup \Sigma \models q$, if $ans(q, D, \Sigma) \neq \varnothing$. The problem of *CQ answering* is defined as follows: given a CQ q, a database D, a set Σ of rules, and a tuple of constants $\mathbf{t}$, decide whether $\mathbf{t} \in ans(q, D, \Sigma)$. In case q is a BCQ, the above problem is called *BCQ answering*. These decision problems are LOGSPACE-equivalent (implicit in (Chandra and Merlin 1977)), and we thus focus only on BCQ answering. For query answering purposes, we can assume, w.l.o.g., that both the BCQ q and the database D contain only predicates of $sch(\Sigma)$.

We are going to employ the *chase procedure*, which works on an instance through the so-called *chase rule* defined as follows. Consider an instance I, and a rule σ as defined above. Then, σ is *applicable* to I if there exists a homomorphism h such that $h(\varphi(\mathbf{X}, \mathbf{Y})) \subseteq I$. Let $I' = I \cup \{h'(p(\mathbf{X}, \mathbf{Z}))\}$, where $h' \supseteq h_{|\mathbf{X}}$ is such that $h'(Z)$ is a "fresh" null not occurring in I, for each $Z \in \mathbf{Z}$. The result of applying σ to I with h is I', and we write $I\langle\sigma, h\rangle I'$; in fact, $I\langle\sigma, h\rangle I'$ defines a single *chase step*. The chase algorithm for a database D and a set Σ of rules consists of an exhaustive application of chase steps in a fair fashion, which leads to a (possibly infinite) model of D and Σ, denoted $chase(D, \Sigma)$; for more details see, e.g., (Calì *et al.* 2012b). The instance $chase(D, \Sigma)$ is a *universal model* of D and Σ, i.e., for each $I \in mods(D, \Sigma)$, there exists a homomorphism that maps $chase(D, \Sigma)$ to I, and thus $D \cup \Sigma \models q$ iff $chase(D, \Sigma) \models q$, for each BCQ q (implicit in (Fagin *et al.* 2005)).

Fragments of Datalog$^\exists$. A rule σ is called *guarded* if there exists $\underline{a} \in body(\sigma)$ which contains all the variables occurring in $body(\sigma)$ (Calì *et al.* 2013). Conventionally, the leftmost such an atom is the *guard* of σ, denoted $guard(\sigma)$, while the non-guard atoms are called the *side* atoms of σ. The class of guarded Datalog$^\exists$, denoted GUARDED, is defined as the family of all possible sets of guarded rules. Given a set $\Sigma \in$ GUARDED, we denote by $sch(\Sigma)^-$ the predicates of Σ which occur in at least one side atom in Σ; clearly, $sch(\Sigma)^- \subseteq sch(\Sigma)$. Guarded rules with one body-atom are called *linear* (Calì *et al.* 2012a), and the corresponding class is denoted LINEAR.

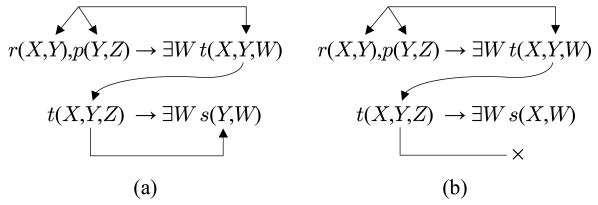


Figure 1: The Sticky Property.

Sticky sets of rules have been proposed in (Calì *et al.* 2012b). The central property of stickiness is that, during the chase, terms which are associated with body-variables that appear more than once (i.e., join variables) always are propagated (or "stick") to the inferred atoms. This is illustrated in Figure 1; the first case is sticky, while the second is not. The associated class is denoted STICKY.

Towards the identification of even more expressive decidable fragments of Datalog$^\exists$, a condition, called *tameness*, which allows the consolidation of guardedness with stickiness has been recently proposed (Gottlob *et al.* 2013). In particular, a set Σ of rules is called *tame* if it can be partitioned into $\{\Sigma[g], \Sigma[s]\}$, where $\Sigma[g]$ is a set of guarded rules and $\Sigma[s]$ is a sticky set of rules, and in addition, none of the rules of $\Sigma[s]$ "feeds" the guard atom of a guarded rule during the construction of the chase. In other words, whenever a guarded rule σ is applied with homomorphism h, then its guard must be mapped by h into an atom obtained from a guarded rule. The corresponding class is denoted TAME.

Fix a class $\mathbb{C} \in \{\mathsf{GUARDED}, \mathsf{LINEAR}, \mathsf{STICKY}, \mathsf{TAME}\}$. Given two integer constants $c > 0$ and $d \geqslant 0$, we define $\mathbb{C}[c, d] = \{\Sigma \in \mathbb{C} \mid |sch(\Sigma)| \leqslant c \text{ and } arity(\Sigma) \leqslant d\}$ as the subclass of $\mathbb{C}$ where the size and the maximum arity of the underlying schema are bounded by c and d, respectively. We define the subclass of $\mathbb{C}$ where the size of the underlying schema is bounded by c as $\mathbb{C}[c, \infty] = \bigcup_{j \geqslant 0} \mathbb{C}[c, j]$, and the subclass of $\mathbb{C}$ where the maximum arity of the underlying schema is bounded by d as $\mathbb{C}[\infty, d] = \bigcup_{i > 0} \mathbb{C}[i, d]$.

3 The Polynomial Combined Approach

The formal definition of polynomial combined first-order rewritability, introduced in (Lutz *et al.* 2009), which is at the basis of the polynomial combined approach, is as follows:

Definition 1 *A Datalog*$^\exists$ *class* $\mathbb{C}$ *is* combined first-order rewritable *if, for every BCQ q, for every database D, and for every $\Sigma \in \mathbb{C}$, it is possible to rewrite:*

- *q and Σ, independently of D, into a first-order query q_Σ,*
- *D and Σ, independently of q, into a database D_Σ,*

such that $D \cup \Sigma \models q$ iff $D_\Sigma \models q_\Sigma$. If q_Σ and D_Σ are constructible in polynomial time, then $\mathbb{C}$ is polynomially combined (P-combined) first-order rewritable.

Combined Reducibility. To show that a Datalog$^\exists$ class is P-combined first-order rewritable, one may exploit the

fact that existing formalisms already enjoy this property. Towards this direction, we introduce the notion of polynomial combined reducibility:

Definition 2 *Consider two Datalog$^\exists$ classes $\mathbb{C}$ and $\mathbb{C}'$. We say that $\mathbb{C}$ is* combined reducible *to $\mathbb{C}'$ if, for every BCQ q, database D, and $\Sigma \in \mathbb{C}$, we can construct:*

- *a BCQ q' independently of D,*
- *a database D' independently of q, and*
- *a set $\Sigma' \in \mathbb{C}'$ independently of q and D,*

such that $D \cup \Sigma \models q$ iff $D' \cup \Sigma' \models q'$. If q', D' and Σ' are constructible in polynomial time, then $\mathbb{C}$ is polynomially combined (P-combined) reducible *to $\mathbb{C}'$, and the employed reduction is called P-combined.*

Obviously, the set of P-combined first-order rewritable Datalog$^\exists$ classes is closed under P-combined reductions:

Theorem 3 *Consider two Datalog$^\exists$ classes $\mathbb{C}, \mathbb{C}'$. If $\mathbb{C}$ is P-combined reducible to $\mathbb{C}'$, and $\mathbb{C}'$ is P-combined first-order rewritable, then also $\mathbb{C}$ is P-combined first-order rewritable.*

The above result suggests the following: to show that a Datalog$^\exists$ class $\mathbb{C}$ is P-combined first-order rewritable, it suffices to reduce query answering under $\mathbb{C}$ to query answering under some P-combined first-order rewritable formalism via a P-combined reduction.

Polynomial Witness Property. A semantic property which is sufficient for P-combined first-order rewritability is the *polynomial witness property*, introduced in (Gottlob and Schwentick 2012). Roughly, a Datalog$^\exists$ class $\mathbb{C}$ enjoys this property if, whenever a query q is entailed by a database D and a set $\Sigma \in \mathbb{C}$, then q is already entailed by a finite part of $chase(D, \Sigma)$ (the witness) of polynomial size in q and Σ.

Definition 4 *A Datalog$^\exists$ class $\mathbb{C}$ enjoys the* polynomial witness property (PWP) *if there exists a polynomial f such that, for every BCQ q, database D, and $\Sigma \in \mathbb{C}$, $D \cup \Sigma \models q$ implies the existence of a chase sequence $I_i \langle h_i, \sigma_i \rangle I_{i+1}$ of D w.r.t. Σ, for $0 \leqslant i < f(|q|, |\Sigma|)$, such that $I_{f(|q|,|\Sigma|)} \models q$.*

The following key result is implicit in (Gottlob and Schwentick 2012):

Proposition 5 *If a Datalog$^\exists$ class $\mathbb{C}$ enjoys the PWP, then $\mathbb{C}$ is P-combined first-order rewritable.*

4 Guarded Datalog$^\exists$

By making different assumptions on the underlying schema, we trace the frontier between those sets of guarded rules which are P-combined first-order rewritable and those which are not. On the one hand, we show that guarded Datalog$^\exists$ is not P-combined first-order rewritable even if the underlying schema contains a single predicate, or even if we focus on unary and binary predicates. On the other hand, guarded Datalog$^\exists$ is P-combined first-order rewritable if we consider only unary predicates, or if the size and the arity of the underlying schema are bounded by an integer constant; the latter forms one of the main results of this paper.

It is well-known that query answering under guarded rules is 2EXPTIME-hard (Calì *et al.* 2013). Since the evaluation of a first-order query is feasible in polynomial space, and also PSPACE $\subsetneq$ 2EXPTIME, we get that GUARDED is not P-combined first-order rewritable. Interestingly, the same argument applies when the underlying schema contains only one predicate. More precisely, query answering under GUARDED$[1, \infty]$ is already 2EXPTIME-hard. This can be shown by a reduction from query answering under arbitrary guarded rules; let us explain the reduction by means of a simple example. Consider the following instance of query answering under guarded Datalog$^\exists$:

$$\begin{array}{rcl} D & = & \{p(a), s(a, b, c)\} \\ \Sigma & = & \{p(X), s(X, Y, Z) \rightarrow \exists W\, r(X, W)\} \\ q & = & \exists X \exists Y\, p(X), r(X, Y). \end{array}$$

The idea is to simulate the predicates p, r and s using a single predicate sim of higher arity; in fact, the atom $p(a)$ can be encoded as $sim(p, a, \diamond_1, \diamond_2, p, r, s, \diamond_1, \diamond_2)$, where $\diamond_1, \diamond_2$ are "dummy" constants not occurring in D, while $s(a, b, c)$ can be encoded as $sim(s, a, b, c, p, r, s, \diamond_1, \diamond_2)$. Then, the rule of Σ will be rewritten as

$$sim(P_1, X, V_1, V_2, \mathbf{T}), sim(P_3, X, Y, Z, \mathbf{T}) \rightarrow \exists W\, sim(P_2, X, W, V_2, \mathbf{T}),$$

where $\mathbf{T} = (P_1, P_2, P_3, V_1, V_2)$, while the query q will be rewritten as

$$\exists X \exists Y \exists \mathbf{T}\, sim(P_1, X, V_1, V_2, \mathbf{T}), sim(P_2, X, Y, V_2, \mathbf{T}).$$

From the above discussion, we get the following result:

Theorem 6 GUARDED$[1, \infty]$ *is not P-combined first-order rewritable.*

The above result follows also from Theorem 3 and the fact that the reduction from multiple predicates to a single predicate is P-combined.

Let us now consider the bounded arity case. Query answering under the description logic $\mathcal{ELI}$, i.e., $\mathcal{EL}$ extended with inverse roles, is EXPTIME-hard (implicit in (Baader *et al.* 2008)). Since each $\mathcal{ELI}$-TBox $\mathcal{T}$ can be translated into a set $\Sigma_\mathcal{T}$ of guarded rules, where $sch(\Sigma_\mathcal{T})$ contains only unary and binary predicates, the next result follows:

Theorem 7 *If* PSPACE $\neq$ EXPTIME*, then* GUARDED$[\infty, 2]$ *is not P-combined first-order rewritable.*

It is believed that PSPACE $\subsetneq$ EXPTIME, and therefore it is unlikely for GUARDED$[\infty, 2]$ to be P-combined first-order rewritable. However, if we concentrate on unary predicates, then it is possible to establish a positive result. In particular, a set $\Sigma \in$ GUARDED$[\infty, 1]$ may contain only rules of the form $p_1(X), \ldots, p_n(X) \rightarrow \underline{a}$, for $n \geqslant 1$, where $\underline{a}$ is either $p(X)$ or $\exists Z\, p(Z)$. This fact allows us to show that GUARDED$[\infty, 1]$ enjoys the PWP. Hence, by Proposition 5, we immediately get the following result:

Theorem 8 GUARDED$[\infty, 1]$ *is P-combined first-order rewritable.*

Let us now investigate whether guarded Datalog$^\exists$ is P-combined first-order rewritable if we bound both the size and the arity of the schema. Notice that query answering under GUARDED$[c, d]$ is NP-complete (Calì *et al.* 2013), and thus the answer to this question is likely to be affirmative; the rest of this section is devoted to show that this is the case. Towards this direction, we exploit the following result established in (Gottlob and Schwentick 2012):

Proposition 9 $\mathsf{LINEAR}[\infty, d]$ *enjoys the PWP.*

The above result, together with Proposition 5, implies that $\mathsf{LINEAR}[\infty, d]$ is P-combined first-order rewritable. Therefore, to establish our desired result, by Theorem 3, it suffices to show that $\mathsf{GUARDED}[c, d]$ is P-combined reducible to $\mathsf{LINEAR}[\infty, d]$. A key notion employed in the studies of query answering under guarded rules is the *type* of an atom. Given a database D and a set $\Sigma \in \mathsf{GUARDED}$, the type of an atom $\underline{a} \in \mathit{chase}(D, \Sigma)$ is defined as $\mathit{type}(\underline{a}, D, \Sigma) = \{\underline{b} \in \mathit{chase}(D, \Sigma) \mid \mathit{dom}(\underline{b}) \subseteq \mathit{dom}(\underline{a})\}$. Intuitively, whenever $\underline{a}$ is associated with the guard atom of a rule σ during a chase step application, then the side atoms of σ are necessarily associated with atoms of $\mathit{type}(\underline{a}, D, \Sigma)$. The abstract idea underlying our P-combined reduction of query answering under $\mathsf{GUARDED}$ to query answering under LINEAR is as follows: given a guarded rule σ, encode the shape of the type τ of $\mathit{guard}(\sigma)$ in a predicate $[\tau]$, and then replace σ with a linear rule of the form $[\tau](\mathbf{X}, \mathbf{Y}) \rightarrow \exists Z\, p(\mathbf{X}, Z)$. Let us now give some extra terminology which is needed for the definition of the reduction.

4.1 Technical Definitions and Notation

Fix two sets $\mathcal{R}, \mathcal{R}'$ of predicates, where $\mathcal{R}' \subseteq \mathcal{R}$. A *guarded type* for a predicate $p \in \mathcal{R}$ w.r.t. $\mathcal{R}'$ is defined as a pair $(\underline{a}, S)$, where $\underline{a}$ is an atom of the form $p(t_1, \ldots, t_n)$ with $t_i \in \{1, \ldots, \mathit{arity}(p)\}$, and $S \subseteq \mathit{base}(\mathit{dom}(\underline{a}), \mathcal{R}') \setminus \{\underline{a}\}$. Let $\mathit{gtypes}(p, \mathcal{R}')$ be the set of all possible guarded types for p w.r.t. $\mathcal{R}'$. We define the guarded types of $\mathcal{R}$ w.r.t. $\mathcal{R}'$ as the set $\mathit{gtypes}(\mathcal{R}, \mathcal{R}') = \{\mathit{gtypes}(p, \mathcal{R}')\}_{p \in \mathcal{R}}$. Given $\tau = (\underline{a}, S) \in \mathit{gtypes}(\mathcal{R}, \mathcal{R}')$, we refer to $\underline{a}$ by $\mathit{guard}(\tau)$ and to $(\{\underline{a}\} \cup S)$ as $\mathit{atoms}(\tau)$, respectively. The arity of τ, denoted $\mathit{arity}(\tau)$, is the maximum integer occurring in $\mathit{guard}(\tau)$.

Example 1 Let $\mathcal{R} = \{p, s, r\}$ and $\mathcal{R}' = \{s, r\}$, where p is a ternary predicate, while s and r are binary predicates. A possible guarded type τ for p w.r.t. $\mathcal{R}'$ is the pair

$$(p(1,1,2), \{s(1,2), s(1,1), r(2,1), r(2,2)\}),$$

with $\mathit{arity}(\tau) = 2$. ∎

Roughly speaking, a guarded type describes the equality type of a guard-atom and a set of further side atoms which are "covered" by the guard. The integer arguments occurring in a guarded type, which can be seen as constants, are mere indicators of equalities among arguments. It is important to clarify that their actual values have no semantic meaning beyond this. For normalization reasons, we adopt the convention that the integers appearing in $\mathit{guard}(\tau)$, for some guarded type τ, will always start by 1, and will be consecutive, except for repeated integers. Thus, when constructing new guarded types from existing ones, we will have to rename the arguments of a newly constructed set of atoms in order to follow the above convention. To this purpose, we use a renaming function ρ whose formal definition is obvious and is omitted; for example, given $\tau = (\underline{a}, S)$, where $\underline{a} = p(2, 2, 4, 1)$ and $S = \{s(1,2), s(4,1)\}$, $\rho(\mathit{atoms}(\tau)) = \{p(1,1,2,3), s(3,1), s(2,3)\}$.

Given a guarded type $\tau = (\underline{a}, S)$, and a tuple of terms $\mathbf{t} = (t_1, \ldots, t_n)$, where $n = \mathit{arity}(\tau)$, the *instantiation* of τ over $\mathbf{t}$, denoted $\tau(\mathbf{t})$, is the set of atoms $\{h_{\mathbf{t}}(\underline{b}) \mid \underline{b} \in \mathit{atoms}(\tau)\}$, where $h_{\mathbf{t}} = \{i \rightarrow t_i\}_{i \in [n]}$; for brevity, $[n] = \{1, \ldots, n\}$.

Example 2 Let τ be the guarded type for p given in Example 1. Then, $\tau((a, b))$ is the pair

$$\{p(a,a,b), s(a,b), s(a,a), r(b,a), r(b,b)\}$$

with $h_{(a,b)} = \{1 \rightarrow a, 2 \rightarrow b\}$. ∎

The *projection* of $\tau = (\underline{a}, S)$ over $I \subseteq [\mathit{arity}(\tau)]$ is the set of atoms $\Pi_I(\tau) = \{\underline{b} \in \mathit{atoms}(\tau) \mid \mathit{dom}(\underline{b}) \subseteq I\}$. E.g., given $\tau = (p(1,2,3), \{r(1), s(2)\})$, then $\Pi_{\{1,2\}}(\tau) = \{r(1), s(2)\}$. Having the above notions in place, we are now ready to show that $\mathsf{GUARDED}[c, d]$ is P-combined reducible to $\mathsf{LINEAR}[\infty, d]$.

4.2 A Reduction from GUARDED to LINEAR

Let us first give a general reduction GL of query answering under guarded to query answering under linear Datalog$^\exists$. Fix a database D and a set $\Sigma_0 \in \mathsf{GUARDED}$. We are going to construct a database $D^\star$ and a set $\Sigma^\star \in \mathsf{LINEAR}$ such that, for every BCQ q, $D \cup \Sigma_0 \models q$ iff $D^\star \cup \Sigma^\star \models q$.

Step 1: Normalization. The first step is to normalize Σ_0 in such a way that each rule contains at most one existentially quantified variable which occurs at the last position of the head-atom. More precisely, for a rule $\sigma \in \Sigma_0$, if σ is already in normal form, then $\mathsf{N}(\sigma) = \{\sigma\}$; otherwise, assuming that $\mathit{head}(\sigma) = \underline{a}$, $\mathbf{X} = \mathit{var}(\mathit{body}(\sigma)) \cap \mathit{var}(\mathit{head}(\sigma))$, and $Z_1, \ldots, Z_m$ are the existentially quantified variables of σ, let $\mathsf{N}(\sigma)$ be the set of rules

$$\begin{array}{rcl}
\mathit{body}(\sigma) & \rightarrow & \exists Z_1\, p^1_\sigma(\mathbf{X}, Z_1) \\
p^1_\sigma(\mathbf{X}, Z_1) & \rightarrow & \exists Z_2\, p^2_\sigma(\mathbf{X}, Z_1, Z_2) \\
& \ldots & \\
p^{m-1}_\sigma(\mathbf{X}, Z_1, \ldots, Z_{m-1}) & \rightarrow & \exists Z_m\, p^m_\sigma(\mathbf{X}, Z_1, \ldots, Z_m) \\
p^m_\sigma(\mathbf{X}, Z_1, \ldots, Z_m) & \rightarrow & \underline{a},
\end{array}$$

where p^i_σ is an $(|\mathbf{X}|+i)$-ary auxiliary predicate not occurring in $\mathit{sch}(\Sigma_0)$, for each $i \in [m]$. Let $\Sigma = \bigcup_{\sigma \in \Sigma_0} \mathsf{N}(\sigma)$. □

A key tool that we shall use in our reduction is the *completion* of a (finite) instance I w.r.t. Σ, i.e., the process of enriching I with all the possible atoms that can be entailed from I and Σ without altering the domain of I. Formally, the result of such a process, denoted $\mathit{complete}(I, \Sigma)$, is the set of atoms $\{\underline{a} \mid \mathit{dom}(a) \subseteq \mathit{dom}(I)$ and $(I \cup \Sigma) \models \underline{a}\}$. It is important to say that the atom entailment problem under guarded rules is decidable (Calì *et al.* 2013), and thus the (finite) set $\mathit{complete}(I, \Sigma)$ can be explicitly computed. Henceforth, a guarded type for a predicate $p \in \mathit{sch}(\Sigma)$ is always w.r.t. $\mathit{sch}(\Sigma)^-$, and we will simply write $\mathit{gtypes}(p)$ for the set $\mathit{gtypes}(p, \mathit{sch}(\Sigma)^-)$. Moreover, we will refer to $\mathit{gtypes}(\mathit{sch}(\Sigma), \mathit{sch}(\Sigma)^-)$ as $\mathit{gtypes}(\Sigma)$.

Step 2: Database Construction. We define

$$D^\star = \left\{ [\tau](\mathbf{t}^1) \;\middle|\; \begin{array}{l} p(\mathbf{t}) \in D, \\ \tau \in \mathit{gtypes}(p), \\ \mathit{guard}(\tau) \simeq p(\mathbf{t}), \\ \tau(\mathbf{t}^1) \subseteq \mathit{complete}(D, \Sigma) \end{array} \right\},$$

where $[\tau]$, for some guarded type τ, is a new $arity(\tau)$-ary predicate not occurring in $sch(\Sigma)$, $guard(\tau) \simeq p(\mathbf{t})$ means that $guard(\tau)$ and $p(\mathbf{t})$ have the same equality type (e.g., $p(1,2,1,3,2) \simeq p(t_1,t_2,t_1,t_3,t_2)$), and $\mathbf{t}^1$ is obtained from $\mathbf{t}$ by keeping only the first occurrence of each term occurring in $\mathbf{t}$ (e.g., if $\mathbf{t} = (t_1,t_2,t_1,t_3,t_2)$, then $\mathbf{t}^1 = (t_1,t_2,t_3)$). Intuitively, an atom $[\tau](\mathbf{t}^1)$, occurring in $D^\star$ due to an atom $p(\mathbf{t}) \in D$, encodes a possible subset of $chase(D,\Sigma)$ which contains the atoms that may act as side atoms whenever $p(\mathbf{t})$ plays the role of the guard in a chase application. □

Example 3 Assume that $D = \{p(a,a), p(a,b), s(b)\}$ and $\Sigma = \{p(X,Y), s(Y) \rightarrow s(X)\}$. For the atom $p(a,a) \in D$, an atom $[\tau](a)$ will be added in $D^\star$, where τ is a guarded type of the form $(p(1,1), S)$. Observe that the only atom, apart from $p(a,a)$, which is entailed by D and Σ, and contains only the term a, is $s(a)$; therefore, S is either empty or is the set $\{s(1)\}$. Consequently, for $p(a,a)$, the atoms $[(p(1,1),\varnothing)](a)$, $[(p(1,1),\{s(1)\})](a)$ will be added in $D^\star$. Consider now the atom $p(a,b) \in D$. It is easy to verify that the set of atoms $\bigcup_{S \in \mathcal{P}(\{p(1,1),s(1),s(2)\})}\{[(p(1,2),S)]\}$, where $\mathcal{P}(X)$ denotes the powerset of X, will be added in $D^\star$. Finally, for $s(b) \in D$, only the atom $[(s(1),\varnothing)](b)$ will be added in $D^\star$, since no other atom than $s(b)$, which contains only the term b, is entailed by D and Σ. ■

Step 3: Rules Construction. The set $\Sigma^\star$ consists of the following two groups of rules:

1. The *type generator*, denoted $\Sigma^\star_G$, which is responsible for generating new guarded types from existing ones, and
2. The *unfolder*, denoted $\Sigma^\star_U$, that is responsible for unfolding a derived guarded type τ, i.e., to explicitly construct the atoms over the original schema $sch(\Sigma_0)$ which are encoded in τ so that they can be queried directly.

Let us now define formally $\Sigma^\star_G$ and $\Sigma^\star_U$; let $w = arity(\Sigma)$.

Step 3.1: Type Generator. For each $\sigma \in \Sigma$ of the form

$$body(\sigma) \rightarrow \exists Z\, p(X_1,\ldots,X_n,Z),$$

and for each guarded type $\tau \in gtypes(\Sigma)$ for which there exists a homomorphism h such that $h(body(\sigma)) \subseteq atoms(\tau)$ and $h(guard(\sigma)) = guard(\tau)$, in $\Sigma^\star_G$ there exists the rule:

$$[\tau](V_1,\ldots,V_m) \rightarrow \exists Z\, [\tau'](W_1,\ldots,W_k,Z),$$

where $m = arity(\tau)$, and

- $\{V_1,\ldots,V_m\} \subseteq var(body(\sigma))$, and $h(V_i) = i$, for each $i \in [m]$,
- $\{W_\ell\}_{\ell\in[k]} \subseteq \{X_\ell\}_{\ell\in[n]}$ and $(h(W_1),\ldots,h(W_k)) = (h(X_1),\ldots,h(X_n))^1$,
- $\tau' = (\rho(\underline{a}), (complete(\rho(\{\underline{a}\}\cup S),\Sigma)\setminus\{\rho(\underline{a})\})_{|sch(\Sigma)^-})$, where $\underline{a}$ is the atom $p(h(X_1),\ldots,h(X_n),w+1)$ and S is the set of atoms $\Pi_{\{h(X_1),\ldots,h(X_n)\}}(\tau)$.

No other rules occur in $\Sigma^\star_G$. □

Example 4 Assume that Σ consists of the rules

$$\sigma_1 \;:\; p(X,Y,X,Z,W), s(X,Z) \rightarrow \exists U\, r(Z,Y,X,U)$$
$$\sigma_2 \;:\; r(X,X,Y,Z) \rightarrow t(X,Z).$$

Let us focus our attention on σ_1. A possible guarded type τ for the predicate of the guard of σ_1, that is, the predicate p, for which there exists a homomorphism that maps $body(\sigma_1)$ to $atoms(\tau)$, and $guard(\sigma_1)$ to $guard(\tau)$ is:

$$\tau = (p(1,2,1,2,3), \{s(1,1), s(1,2)\}).$$

Therefore, in $\Sigma^\star_G$ we have the linear rule:

$$[\tau](X,Y,W) \rightarrow \exists U\, [\tau'](Y,X,U),$$

where $\tau' = (r(1,1,2,3), \{s(2,2), s(2,1), t(1,3)\})$. ■

Step 3.2: Unfolder. For each $\tau \in gtypes(\Sigma_0)$, if $guard(\tau)$ is of the form $p(t_1,\ldots,t_n)$, in $\Sigma^\star_U$ there exists the rule:

$$[\tau](V_1,\ldots,V_m) \rightarrow p(W_1,\ldots,W_n),$$

where

- $\{V_1,\ldots,V_m\} \subset \mathbf{V}$ and $(V_1,\ldots,V_m) = (W_1,\ldots,W_n)^1$,
- $t_i = t_j$ implies $W_i = W_j$, for each $i,j \in [n]$.

No other rules occur in $\Sigma^\star_U$. □

Example 5 Consider the guarded type

$$\tau = (p(1,2,1,2,3), \{s(1,1), s(1,2)\}).$$

Due to τ, in $\Sigma^\star_U$ we have the linear rule:

$$[\tau](X,Y,Z) \rightarrow p(X,Y,X,Y,Z).$$

Notice that, for any type of the form $(p(1,2,1,2,3), S)$, i.e., that has the same guard as τ, the rule added to $\Sigma^\star_U$ is the same as the one above. ■

We proceed to show that GL is indeed a reduction of query answering under guarded Datalog$^\exists$ to query answering under linear Datalog$^\exists$. To this aim, we exploit the central property of the type of an atom. In particular, the subtree of the *guarded chase forest* of D and Σ — the forest obtained from $chase(D,\Sigma)$ by keeping only the guards and their children — rooted at $\underline{a}$ is determined by $type(\underline{a},D,\Sigma)$ (modulo renaming of nulls) (Calì *et al.* 2013). This implies that, given an atom $\underline{a} = p(t_1,\ldots,t_n,z)$, where z is a null value invented in $\underline{a}$, we can construct $type(a,D,\Sigma)$ from $\underline{a}$ and the restriction of $type(a,D,\Sigma)$ to $\{t_1,\ldots,t_n\}$. This justifies the definition of the type generator, where we complete the known part of the type, denoted S, of a generated atom $\underline{a}$ by calling the procedure $complete$ using as a database the instance $(\{\underline{a}\} \cup S)$.

The above property allows us to show two useful lemmas. The first one asserts that, for each $\underline{a} \in chase(D,\Sigma)$, GL encodes $type(\underline{a},D,\Sigma)$ in a complete way. The function $\gamma_{\underline{a}}$ renames the arguments of an atom of $type(\underline{a},D,\Sigma)$ into integers. For example, given $\underline{a} = p(a,z_1,a,b)$, then $\gamma_{\underline{a}}(\underline{a}) = p(1,2,1,3)$; the definition of $\gamma_{\underline{a}}$ is obvious and it is omitted. Let $\tau_{\underline{a}} = (\gamma_{\underline{a}}(\underline{a}), \gamma_{\underline{a}}(type(\underline{a},D,\Sigma)\setminus\{\underline{a}\}))$.

Lemma 10 *There is a homomorphism h such that, $p(\mathbf{t}) \in chase(D, \Sigma)$ implies $[\tau_{p(\mathbf{t})}](h(\mathbf{t}^1)) \in chase(D^\star, \Sigma^\star_G)$.*

The second technical lemma states that the reduction GL encodes in a sound way the various types that can be realized in the instance constructed by the chase:

Lemma 11 *There is a homomorphism h such that, $[\tau](\mathbf{t}) \in chase(D^\star, \Sigma^\star_G)$ implies $h(\tau(\mathbf{t})) \subseteq chase(D, \Sigma)$.*

By exploiting the above technical lemmas, we can show that $chase(D, \Sigma_0)$ and $chase(D^\star, \Sigma^\star)$, after eliminating the atoms with predicates not occurring in $sch(\Sigma_0)$, are homomorphically equivalent:

Proposition 12 *It holds that,*

$$chase(D, \Sigma_0) \longleftrightarrow chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)}.$$

Proof (sketch). By construction, there exists a homomorphism μ that maps $chase(D, \Sigma_0)$ to $chase(D, \Sigma)_{|sch(\Sigma_0)}$. Due to $\Sigma^\star_U$, we get that h_1, provided by Lemma 10, maps $chase(D, \Sigma)_{|sch(\Sigma_0)}$ to $chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)}$. Therefore,

$$h_1(\mu(chase(D, \Sigma_0))) \subseteq chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)}.$$

Conversely, let h_2 be the homomorphism as in Lemma 11. Since $chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)} \subseteq \cup_{[\tau](\mathbf{t}) \in chase(D^\star, \Sigma^\star_G)} \tau(\mathbf{t})$, h_2 maps $chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)}$ to $chase(D, \Sigma)_{|sch(\Sigma_0)}$. Clearly, by construction, there exists a homomorphism μ' that maps $chase(D, \Sigma)_{|sch(\Sigma_0)}$ to $chase(D, \Sigma_0)$. Thus,

$$\mu'(h_2(chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)})) \subseteq chase(D, \Sigma_0).$$

Consequently, due to $(h_1 \circ \mu)$ and $(\mu' \circ h_2)$, we get that $chase(D, \Sigma_0) \longleftrightarrow chase(D^\star, \Sigma^\star)_{|sch(\Sigma_0)}$. □

Using Proposition 12, we can show that, for every BCQ q, $chase(D, \Sigma_0) \models q$ iff $chase(D^\star, \Sigma^\star) \models q$, since q contains only predicates of $sch(\Sigma_0)$, and the next result follows:

Theorem 13 *For every BCQ q, $D \cup \Sigma_0 \models q$ iff $D^\star \cup \Sigma^\star \models q$.*

A natural question is whether GL is a P-combined reduction. Recall that we are interested in sets of guarded rules where the size and the maximum arity of the underlying schema are bounded by a constant. In this case, as we show below, indeed GL is P-combined.

4.3 The Reduction GL is P-combined

By construction, $D^\star$ depends only on D and Σ_0, while $\Sigma^\star$ depends only on Σ_0; hence, GL is a combined reduction. Thus, to establish the desired result, it remains to show that, if $\Sigma_0 \in \mathsf{GUARDED}[c, d]$, then $D^\star$ and $\Sigma^\star$ are constructible in polynomial time. Via a combinatorial analysis, we can establish some upper bounds on the number of all possible guarded types, and also on the size of a guarded type τ, denoted $|\tau|$, and defined as $|atoms(\tau)|$.

Lemma 14 *It holds that,*

1. $|gtypes(p)| \leqslant arity(p)^{arity(p)} \cdot 2^{|sch(\Sigma)^-| \cdot arity(p)^w}$, *for each $p \in sch(\Sigma)$,*
2. $|gtypes(\Sigma)| \leqslant |sch(\Sigma)| \cdot w^w \cdot 2^{|sch(\Sigma)^-| \cdot w^w}$, *and*
3. $|\tau| \leqslant |sch(\Sigma)| \cdot w^w$, *for each $\tau \in gtypes(\Sigma)$.*

Henceforth, we assume that $\Sigma_0 \in \mathsf{GUARDED}[c, d]$. Observe that the maximum number of existentially quantified variables that can appear in a rule of Σ_0 is $arity(\Sigma_0)$. Therefore, $|\Sigma| \in \mathcal{O}(|\Sigma_0|)$ and $|sch(\Sigma)| \in \mathcal{O}(|\Sigma_0|)$. Since the auxiliary predicates introduced during the normalization procedure do not appear in side atoms, we get that $sch(\Sigma)^- \subseteq sch(\Sigma_0)$, and thus $|sch(\Sigma)^-| \leqslant c$. It is important to note that $arity(\Sigma) = arity(\Sigma_0) \leqslant d$.

To construct $D^\star$ we need, for each atom $p(\mathbf{t}) \in D$, and for each $\tau \in gtypes(p)$ such that $guard(\tau) \simeq p(\mathbf{t})$, to check whether $D \cup \Sigma \models \underline{a}$, for each $\underline{a} \in \tau(\mathbf{t}^1)$. Importantly, the latter (atom entailment) check is feasible in polynomial time; this follows from the fact that atomic query answering under $\mathsf{GUARDED}[c, d]$ can be performed in polynomial time in D and Σ (Calì *et al.* 2013). In the worst case, we need to apply this polynomial check $\sum_{p(\mathbf{t}) \in D} \sum_{\tau \in gtypes(p)} |\tau|$ times. By Lemma 14, and since $|sch(\Sigma)^-| \leqslant c$ and $arity(p) \leqslant d$, for each $p \in sch(\Sigma)$, the crucial entailment check must be performed $|D| \cdot |sch(\Sigma)| \cdot 2^{c \cdot d^d} \cdot d^{2d} \in \mathcal{O}(|D| \cdot |\Sigma_0|)$ times, and hence $D^\star$ is constructible in polynomial time.

We now consider the construction of $\Sigma^\star$. By exploiting Lemma 14, one can show that the maximum number of rules that can appear in $\Sigma^\star_G$ is $\mathcal{O}(|\Sigma_0|^2)$. The construction of each linear rule (apart from some easy steps) amounts to compute the set of atoms $complete(I, \Sigma)_{|sch(\Sigma)^-}$, where I is an instance of size bounded by $|\tau|$ with τ the guarded type under consideration. This requires $\mathcal{O}(|\Sigma_0|^d)$ polynomial entailment checks, and hence $\Sigma^\star_G$ is constructible in polynomial time. To construct the unfolder $\Sigma^\star_U$, we actually need to construct $|gtypes(\Sigma_0)|$ linear rules, each of which can be constructed in time $\mathcal{O}(1)$. Thus, by Lemma 14, the construction of $\Sigma^\star_U$ can be carried out in time $\mathcal{O}(1)$.

From the above analysis, we conclude that $D^\star$ and $\Sigma^\star$ are constructible in polynomial time, and the next result follows:

Proposition 15 $\mathsf{GUARDED}[c, d]$ *is P-combined reducible to* $\mathsf{LINEAR}[\infty, d]$.

By combining Theorem 3 with Propositions 5, 9 and 15, the main result of this section follows:

Theorem 16 $\mathsf{GUARDED}[c, d]$ *is P-combined first-order rewritable.*

5 Tame Datalog$^\exists$

In this section, we investigate the question whether the class of tame rules is P-combined first-order rewritable whenever the size and the arity of the schema are bounded. One may claim that the technique proposed for guarded rules can be directly applied on tame rules in order to obtain the desired result. More precisely, one can suggest the following: it can be shown that $\mathsf{TAME}[c, d]$ is P-combined reducible to a P-combined first-order rewritable class — which in turn implies that $\mathsf{TAME}[c, d]$ is P-combined first-order rewritable — by rewriting the given database D into $D^\star$, and the guarded part $\Sigma[g]$ of the given tame set $\Sigma \in \mathsf{TAME}[c, d]$ into $\Sigma[g]^\star$, while keeping the sticky part $\Sigma[s]$ of Σ untouched. Although this approach is quite promising, there are two non-trivial difficulties:

- $(\Sigma[g]^\star \cup \Sigma[s])$ is sticky, and it is not known whether $\mathsf{STICKY}[\infty, d]$ is P-combined first-order rewritable, and
- GL is not complete on tame rules since a side atom of a guarded rule may depend on the sticky part.

In what follows, we explore the combined approach for sticky rules, and we show that $\mathsf{STICKY}[\infty, d]$ is P-combined first-order rewritable. We also reveal the key reason of the incompleteness of GL, and we briefly explain how it can be adapted in order to regain completeness.

5.1 Stickiness and the Combined Approach

Query answering under sticky sets of rules is EXPTIME-complete (Calì *et al.* 2012b). Interestingly, the problem remains EXPTIME-hard even if the underlying schema contains only one predicate. This can be established by employing the same construction given in the previous section to show an analogous result for guarded rules. Thus, we get the following impossibility result:

Theorem 17 *If* PSPACE $\neq$ EXPTIME*, then* $\mathsf{STICKY}[1, \infty]$ *is not P-combined first-order rewritable.*

Let us now show that $\mathsf{STICKY}[\infty, d]$ is P-combined first-order rewritable by showing first that it enjoys the polynomial witness property. Consider a BCQ q, a database D, and a set $\Sigma \in \mathsf{STICKY}[\infty, d]$. Assume that $D \cup \Sigma \models q$. This implies that there exists a homomorphism h such that $h(q) = H \subseteq \mathit{chase}(D, \Sigma)$. Consider now the proof of H, i.e., the (finite) part $P \subseteq \mathit{chase}(D, \Sigma)$ due to which H is generated. Because of stickiness, apart from the terms occurring in H, all the other terms in P do not participate in a join operation during the construction of P, and therefore can be seen as "don't care" terms. Given an atom $\underline{a} \in P$, let $\underline{a}_\star$ be the atom obtained from $\underline{a}$ by replacing the "don't care" terms, i.e., the terms of $\mathit{dom}(\underline{a}) \setminus \mathit{dom}(H)$, with $\star$. It is possible to show that a witness P' of q can always be constructed such that, for every $\underline{a}, \underline{a}' \in P'$, $\underline{a} \neq \underline{a}'$ implies $\underline{a}_\star \neq \underline{a}'_\star$. Since $|P'| \leqslant n \cdot (|\mathit{dom}(H)| + 1)^d$, where $n = |\mathit{sch}(\Sigma)|$, and $|\mathit{dom}(H)| \leqslant |q| \cdot d$, we conclude that a witness of size $\mathcal{O}(n \cdot |q|^d)$ always exists, and the next result follows:

Lemma 18 $\mathsf{STICKY}[\infty, d]$ *enjoys the PWP.*

The above result improves an existing one asserting that $\mathsf{STICKY}[c, d]$ enjoys the PWP (Gottlob and Schwentick 2012). By Proposition 5 and Lemma 18 we get that:

Theorem 19 $\mathsf{STICKY}[\infty, d]$ *is P-combined first-order rewritable.*

5.2 A Reduction from TAME to STICKY

We now concentrate on the reduction GL. We first show that it is not complete on tame rules. Let $D = \{p(a, b), p(b, c)\}$, and Σ be the tame set of rules consisting of:

$$\begin{array}{lcl} \sigma_1 & : & p(X, Y) \rightarrow \exists Z\, r(Y, Z) \\ \sigma_2 & : & r(X, Y), s(Y, X) \rightarrow \exists Z\, r(Y, Z) \\ \sigma_3 & : & r(X, Y), p(X, Z) \rightarrow s(Y, X), \end{array}$$

where the guarded part $\Sigma[g] = \{\sigma_1, \sigma_2\}$ and the sticky part $\Sigma[s] = \{\sigma_3\}$. The chase of D w.r.t. Σ is

$$C = \{p(a, b), p(b, c), r(b, z_1), r(c, z_2), s(z_1, b), r(z_1, z_3)\}.$$

Let us now construct $D^\star$ and $\Sigma[g]^\star$. It is easy to verify that

$$D^\star = \{[(p(1, 2), \varnothing)](a, b), [(p(1, 2), \varnothing)](b, c)\}.$$

Due to σ_1 and the guarded type $(p(1, 2), \varnothing)$ for p, in $\Sigma[g]^\star$ we have the linear rule

$$[(p(1, 2), \varnothing)](X, Y) \rightarrow \exists Z\, [(r(1, 2), \varnothing)](Y, Z).$$

Moreover, due to σ_2 and the type $(r(1, 2), \{s(2, 1)\})$ for r, in $\Sigma[g]^\star$ we have the linear rule

$$[(r(1, 2), \{s(2, 1)\})](X, Y) \rightarrow \exists Z\, [(r(1, 2), \varnothing)](Y, Z).$$

Notice that several other rules occur in $\Sigma[g]^\star$. However, for this particular example they are meaningless, and therefore they are omitted for brevity; e.g., rules that must occur in $\Sigma[g]^\star$ due to σ_1 and a type of the form $(p(1, 1), S)$ will never be triggered. In addition, in $\Sigma[g]^\star$ we have the linear rules

$$\begin{array}{l} [(p(1, 2), \varnothing)](X, Y) \rightarrow p(X, Y) \\ [(r(1, 2), \varnothing)](X, Y) \rightarrow r(X, Y) \\ [(r(1, 2), \{s(2, 1)\})](X, Y) \rightarrow r(X, Y). \end{array}$$

Observe that $\mathit{chase}(D^\star, \Sigma[g]^\star \cup \Sigma[s])_{|\mathit{sch}(\Sigma)}$ is

$$C^\star = \{p(a, b), p(b, c), r(b, z_1^\star), r(c, z_2^\star), s(z_1^\star, b)\}.$$

The fact that there exists a homomorphism that maps $C^\star$ to C implies that, for every BCQ q, $C^\star \models q$ implies $C \models q$. Actually, the reduction GL is sound even if we consider tame rules. However, it is not also complete: consider the BCQ $q = \exists X \exists Y \exists Z\, r(X, Y), r(Y, Z)$, and observe that $C \models q$ but $C^\star \not\models q$. The reason why $C^\star \not\models q$ is because GL failed to encode the fact that an atom $r(z_1^\star, z_3^\star)$ must also be generated during the construction of $\mathit{chase}(D^\star, \Sigma[g]^\star \cup \Sigma[s])$. More precisely, the crucial linear rule σ of the form

$$[(p(1, 2), \varnothing)](X, Y) \rightarrow \exists Z\, [(r(1, 2), \{s(2, 1)\})](Y, Z)$$

is missing from $\Sigma[g]^\star$. Because of the above rule, the atom $[(r(1, 2), \{s(2, 1)\}](b, z_1^\star)$ will be obtained during the construction of the chase, which in turn will trigger the rule

$$[(r(1, 2), \{s(2, 1)\})](X, Y) \rightarrow \exists Z\, [(r(1, 2), \varnothing)](Y, Z)$$

and the atom $[(r(1, 2), \varnothing)](z_1^\star, z_3^\star)$ will be generated. Then, due to the rule

$$[(r(1, 2), \varnothing)](X, Y) \rightarrow r(X, Y)$$

the desired atom $r(z_1^\star, z_3^\star)$ will be eventually obtained. Thus, in order to reveal the real reason why GL is not complete on tame rules, we need to understand why the crucial rule σ does not occur in $\Sigma[g]^\star$. Clearly, σ should exist in $\Sigma[g]^\star$ due to σ_1 and a guarded type τ of the form $(p(1, 2), S)$, where $S \subseteq \mathit{base}(\{1, 2\}, \mathit{sch}(\Sigma))$. However,

$$\begin{array}{rcl} \tau' & = & (r(1, 2), \mathit{complete}(\{r(1, 2)\} \cup \Pi_{\{2\}}(\tau), \Sigma)) \\ & = & (r(1, 2), \varnothing), \end{array}$$

even if $S = \mathit{base}(\{1, 2\}, \mathit{sch}(\Sigma))$, and therefore there is no way to obtain the head atom $\exists Z\, [(r(1, 2), \{s(1, 2)\})](Y, Z)$. This is because, in order to obtain $s(2, 1)$ during the completion of $(\{r(1, 2)\} \cup \Pi_{\{2\}}(\tau))$ w.r.t. Σ, an atom of the form $p(1, X)$ is needed. However, such an atom does not occur

in $\mathit{complete}(\{r(1,2)\} \cup \Pi_{\{2\}}(\tau), \Sigma)$, and this is exactly the source of incompleteness. It is important to observe that the atom $p(1, X)$ is needed to trigger the rule σ_3, i.e., the sticky rule of Σ, and since X is not propagated in the inferred atom, by stickiness, we can safely conclude that such a term X will never participate in a join operation during the completion process. Hence, what is important is not the actual value of X but the existence of a witness for X.

From the above discussion, we conclude that the way to regain completeness is to add in $\mathit{side}(\tau)$ a witness atom $p(1, \star)$ for $p(1, X)$, and let

$$\tau' = (r(1,2), \mathit{complete}(\{r(1,2)\} \cup \Pi_{\{2,\star\}}(\tau), \Sigma)),$$

and also add an additional unfold rule of the form

$$[(r(1,2), \{s(2,1), p(1,\star)\})](X,Y) \rightarrow \exists Z\, p(X,Z)$$

in order to explicitly unfold the witness atom $p(1, \star)$ by inventing a "fresh" null for each occurrence of $\star$. Notice that in this particular example, the above rule is not needed since $p(1, \star)$ is a witness of the database atom $p(b, c)$, and thus it is not necessary to regenerate it. However, in general, such unfolding rules are crucial.

The above approach can be generalized in order to obtain a combined reduction, denoted TS, of query answering under tame sets of rules to query answering under sticky sets of rules. Let us clarify that $(\Sigma[g]^\star \cup \Sigma[s])$ is indeed sticky since, by construction, for each $\sigma \in \Sigma[g]^\star$, every variable in $\mathit{body}(\sigma)$ occurs only once, and thus there is no way to violate stickiness after the merging of $\Sigma[g]^\star$ and $\Sigma[s]$. Moreover, if the size and arity of the underlying schema is fixed, then TS is P-combined, and the next result follows:

Proposition 20 $\mathsf{TAME}[c, d]$ *is P-combined reducible to* $\mathsf{STICKY}[\infty, d]$.

From Theorem 3, Theorem 19 and Proposition 20, we immediately get the main result of this section:

Theorem 21 $\mathsf{TAME}[c, d]$ *is P-combined first-order rewritable.*

6 Linear Datalog$^\exists$: The Missing Tile

It is open whether Linear Datalog$^\exists$ is P-combined first-order rewritable, and we are currently investigating this problem. The purpose of this section is to show that the solution to the above open problem cannot benefit from existing tools, and therefore novel techniques must be devised.

By exploiting the reduction in the proof of Theorem 6, we can show that LINEAR and $\mathsf{LINEAR}[1, \infty]$ coincide w.r.t. P-combined first-order rewritability. Therefore, in order to establish the desired result, it suffices to focus our attention on sets of linear rules which contain exactly one predicate. Since query answering under LINEAR is PSPACE-hard, the same problem under $\mathsf{LINEAR}[1, \infty]$ is also PSPACE-hard. Thus, in view of the fact that query answering under the (currently) known Datalog$^\exists$ languages which are P-combined first-order rewritable — i.e., the languages considered in this paper — is in NP, the P-combined reducibility approach cannot be applied (unless NP = PSPACE).

Unfortunately, as shown below, the existing techniques of establishing that $\mathsf{LINEAR}[\infty, d]$ is P-combined first-order rewritable, which are based on the PWP, are also not useful, or, at least, immediately applicable. We define the class $\mathsf{SUCC} = \{\Sigma_n\}_{n>0}$, where Σ_n is the set of linear rules

$$\{\mathit{num}(Z, O, \mathbf{B}_{n-i}, Z, O^{i-1}) \rightarrow \mathit{num}(Z, O, \mathbf{B}_{n-i}, O, Z^{i-1})\}_{i \in [n]}$$

with $\mathbf{B}_k = B_1, \ldots, B_k$. Roughly speaking, Σ_n simulates the successor operator on m-digit binary numbers, where $m = n - 2$. The binary number $b_1 b_2 \ldots b_m$ is encoded as $\mathit{num}(Z, O, b_1, \ldots, b_m)$, where Z, O are auxiliary variables that allow us to have access to the bits 0 and 1, respectively. Given the atomic query $q = \mathit{num}(0, 1, 1, \ldots, 1)$ and the database $D = \{\mathit{num}(0, 1, 0, \ldots, 0)\}$, we need to apply $\mathcal{O}(2^m)$ times the chase step in order to entail q. Consequently, SUCC does not enjoy the PWP. Since $\mathsf{SUCC} \subset \mathsf{LINEAR}[1, \infty]$, the next result follows:

Lemma 22 $\mathsf{LINEAR}[1, \infty]$ *does not enjoy the PWP.*

One may think that Lemma 22 implies that $\mathsf{LINEAR}[1, \infty]$ is not P-combined first-order rewritable. However, the PWP is not a necessary condition for the P-combined first-order rewritability. This is established by showing first that a class for which query answering is PTIME-hard does not enjoy the PWP. Since there exist PTIME-hard classes which are P-combined first-order rewritable, e.g., $\mathsf{GUARDED}[c, d]$, the next result follows:

Lemma 23 *There exists a Datalog$^\exists$ class which does not enjoy the PWP, but it is P-combined first-order rewritable.*

All the known formalisms that are P-combined first-order rewritable, enjoy a property close to the PWP which we call *weakly PWP (wPWP)*. Roughly, this property is defined as the PWP, with the difference that the polynomial size witness may depend on the given database. Clearly, SUCC, and thus $\mathsf{LINEAR}[1, \infty]$, does not enjoy the wPWP. However, it is possible to show that SUCC is P-combined first-order rewritable, and thus obtaining the first example which does not ensure a witness of polynomial size, but it is P-combined first-order rewritable.

From the above discussion, we conclude that LINEAR, although it does not guarantee witnesses of polynomial size, is likely to be P-combined first-order rewritable. However, the existing tools are not powerful enough for establishing such a result, and novel techniques must be developed.

7 Conclusion

We considered the polynomial combined approach to query answering under expressive classes of existential rules. To the best of our knowledge, this work is the first attempt to explicitly apply the combined approach to existential rules. We established that guarded and tame rules are polynomially combined first-order rewritable, whenever the size and the arity of the underlying schema are bounded. The same holds also for sticky rules, even if the size of the underlying schema is not bounded. Notice that our techniques immediately show that guarded and tame rules are combined first-order rewritable. The results of this work are, for the moment, of theoretical nature and we do not claim that they

will directly lead to better practical algorithms. A smart implementation and an evaluation of the obtained rewritings will be the subject of future research. We are also planning to investigate the notion of *output-polynomial* combined first-order rewritability, i.e., when the rewriting is of polynomial size but not necessarily constructible in polynomial time.

Acknowledgements. This research has received funding from ERC Grant 246858 (DIADEM). Andreas Pieris also acknowledges the EPSRC Grant EP/H051511/1 (ExODA).

References

Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope further. In *Proc. of OWLED*, 2008.

Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654, 2011.

Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. Walking the complexity lines for generalized guarded existential rules. In *Proc. of IJCAI*, pages 712–717, 2011.

Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.

Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.

Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. of STOCS*, pages 77–90, 1977.

Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

Georg Gottlob and Thomas Schwentick. Rewriting ontological queries into small nonrecursive datalog programs. In *Proc. of KR*, 2012.

Georg Gottlob, Marco Manna, and Andreas Pieris. Combining decidability paradigms for existential rules. *TPLP*, 13(4-5):877–892, 2013.

Stanislav Kikot, Roman Kontchakov, Vladimir V. Podolskii, and Michael Zakharyaschev. Exponential lower bounds and separation for query rewriting. In *Proc. of ICALP*, pages 263–274, 2012.

Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in DL-Lite. In *Proc. of KR*, 2010.

Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to ontology-based data access. In *Proc. of IJCAI*, pages 2656–2661, 2011.

Markus Krötzsch and Sebastian Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In *Proc. of IJCAI*, pages 963–968, 2011.

Nicola Leone, Marco Manna, Giorgio Terracina, and Pierfrancesco Veltri. Efficiently computable Datalog$^{\exists}$ programs. In *Proc. of KR*, 2012.

Carsten Lutz, David Toman, and Frank Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. of IJCAI*, pages 2070–2075, 2009.

Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

Lightweight Description Logics and Branching Time: A Troublesome Marriage

Víctor Gutiérrez-Basulto
Universität Bremen, Germany
University of Liverpool, UK
victor@informatik.uni-bremen.de

Jean Christoph Jung and **Thomas Schneider**
Universität Bremen, Germany
{jeanjung, tschneider}@informatik.uni-bremen.de

Abstract

We study branching-time temporal description logics (BTDLs) based on the temporal logic CTL in the presence of rigid (*time-invariant*) roles and general TBoxes. There is evidence that, if full CTL is combined with the classical $\mathcal{ALC}$ in this way, reasoning becomes undecidable. In this paper, we begin by substantiating this claim, establishing undecidability for a fragment of this combination. In view of this negative result, we then investigate BTDLs that emerge from combining fragments of CTL with lightweight DLs from the $\mathcal{EL}$ and *DL-Lite* families. We show that even rather inexpressive BTDLs based on $\mathcal{EL}$ exhibit very high complexity. Most notably, we identify two *convex* fragments which are undecidable and hard for non-elementary time, respectively. For BTDLs based on *DL-Lite*$_{bool}^{\mathcal{N}}$, we obtain tight complexity bounds that range from PSPACE to EXPTIME.

1 Introduction

Classical description logics (DLs) such as those that underlie the W3C recommendation OWL, are decidable fragments of first order logic and aim at the representation of and reasoning about *static* knowledge. With the objective of enhancing DLs with means to capture temporal aspects of knowledge, over the last 20 years much effort has been spent on the study of temporal description logics (TDLs) as discussed in detail in the surveys (Artale and Franconi 2000; Lutz, Wolter, and Zakharyaschev 2008) and the references therein. TDLs are useful for applications requiring to describe dynamic aspects of the application domain. For example, in the medical domain the description of a term might refer to temporal patterns; hence a medical ontology like SNOMED CT (Baader, Ghilardi, and Lutz 2008) should model the knowledge that, for example, the disease *diabetes* may potentially lead to several ocular disorders in the future.

A prominent approach to TDLs, following Schild's original proposal (Schild 1993), is to combine classical DLs with the standard temporal logics LTL, CTL and CTL* based on a two-dimensional product-like semantics in the style of many-dimensional DLs (Gabbay et al. 2003). In the construction of temporal DLs, besides the logics used in the temporal and DL component, there are a number of further design choices. For example, we can choose whether we apply temporal operators to concepts, roles or TBoxes. We can moreover define some DL roles or concepts as rigid, meaning that they will not change their interpretation over time: for example, since every human has the same genetic disorders during their life, the relation "has genetic disorder" between humans and diseases should be modelled by a rigid role hasGeneticDisease. Consequently, it has been argued that applications of TDLs require rigid roles and general TBoxes; e.g., in temporal data modeling (Artale et al. 2012) or in medical ontologies such as SNOMED CT.

Remarkably, the landscape of the expressivity and computational complexity of linear-time TDLs based on LTL (LTDLs) is well-understood today (Artale et al. 2007; Baader, Ghilardi, and Lutz 2008; Franconi and Toman 2011; Artale et al. 2012). In particular, various design choices have been explored, and both lightweight and expressive DLs have been considered. Notably, it has been shown that LTDLs based on the basic DL $\mathcal{ALC}$ become undecidable in the presence of a general TBox as soon as temporal operators are applied not only to concepts but also to roles, or rigid roles are allowed.

It is interesting to note that, more generally, combinations of DLs and modal logics allowing for rigid roles and general TBoxes tend to have very high computational complexity. Indeed, the standard reasoning problem for DLs is concept satisfiability with respect to a TBox. Via the well-known correspondence of DL combinations with product modal logics (PMLs) (Gabbay et al. 2003, Chapters 3, 14), this reasoning problem relates to the global consequence problem in a product modal logic. For many PMLs, global consequence is known to be undecidable (Gabbay et al. 2003).

In the context of LTDLs, with the aim to attain decidability in the presence of rigid roles and general TBoxes, recent work investigates lightweight DLs from the $\mathcal{EL}$ and *DL-Lite* families as the DL component (Artale et al. 2007; 2012). This choice is supported by the fact that lightweight DLs typically allow for tractable reasoning, and motivated by their importance in applications such as ontology-based data access or the representation of huge bio- and medical ontologies (Baader, Brandt, and Lutz 2005; Artale et al. 2009). The results of this work on LTDLs are two-fold: the studied LTDLs based on *DL-Lite* are decidable while those based on $\mathcal{EL}$ are undecidable. Undecidability in the $\mathcal{EL}$ case is explained by an interaction of two phenomena: the first is

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

non-convexity – roughly speaking, the ability to simulate disjunctions, here using properties of linear-time, as $\Diamond \mathsf{X} \sqcap \Diamond \mathsf{Y}$ implies that one of $\Diamond(\mathsf{X} \sqcap \Diamond \mathsf{Y})$ and $\Diamond(\mathsf{Y} \sqcap \Diamond \mathsf{X})$ is true, where $\Diamond \mathsf{X}$ reads as *eventually* in the future X holds. The second is the known fact that $\mathcal{EL}$ with disjunctions can encode $\mathcal{ALC}$; hence TDLs based on $\mathcal{EL}$ are as hard as the corresponding $\mathcal{ALC}$ variant, which is undecidable in the case of LTDLs.

From the viewpoint of some DL applications, it has been argued that representing the existence of different *possible futures*, inherent to a branching-time structure, is necessary for a more appropriate modeling (Gutiérrez-Basulto, Jung, and Lutz 2012); for example, in medical ontologies where a symptom or disease might evolve in different ways in the future. For instance consider the statement *'each patient having the autoimmune disorder diabetes will* possibly *develop glaucoma in the future'*. In LTDLs this can be modeled as

$$\mathsf{Patient} \sqcap \exists \mathsf{hasAutoDis}.\mathsf{Diabetes} \sqsubseteq \Diamond \exists \mathsf{develops}.\mathsf{Glaucoma},$$

meaning that each diabetes patient will *eventually* develop glaucoma, thus excluding the possibility that a glaucoma will never be developed if diabetes is controlled. In branching-time TDLs we can use the existential path quantifier $\mathbf{E}$ together with $\Diamond$ to more carefully state that there exists a *possible future* where the patient develops glaucoma, leaving open the existence of other possible futures.

Interestingly, opposite to LTDLs, the study of TDLs based on branching-time temporal logics (BTDLs) has been rather limited; mainly focusing on decidability boundaries obtained in the context of first-order branching temporal logic (Hodkinson, Wolter, and Zakharyaschev 2002; Bauer et al. 2004). Only recently tight elementary bounds were presented for combinations with the DLs $\mathcal{ALC}$ and $\mathcal{EL}$ in the case where only *local* roles are allowed (Gutiérrez-Basulto, Jung, and Lutz 2012). Hence, the aim of this paper is to deepen the study of BTDLs by considering rigid roles, that is, we study the case where temporal operators are applied to concepts, and rigid roles and general TBoxes are allowed. In particular, we will analyze the decidability and complexity of the standard reasoning problems. In the light of the known results from the literature, we will focus on BTDLs based on lightweight DLs from the $\mathcal{EL}$ and *DL-Lite* families, but we will lay the groundwork by reconsidering $\mathcal{ALC}$.

Our study starts with the combination of CTL and $\mathcal{ALC}$. We show that its fragment with only the $\mathbf{E}\Diamond$, $\mathbf{A}\Box$ operators is undecidable, combining results for products of 'transitive' modal logics with a well-known DL-technique for reasoning about transitive roles (Tobies 2001). Notably, we obtain a more general result, covering a whole family of PMLs.

We continue by investigating several BTDLs based on $\mathcal{EL}$. We identify non-convex BTDLs based on $\mathcal{EL}$ and thus show undecidability of their subsumption problem similarly to the results described above. As a next step, we focus on identifying convex BTDLs based on $\mathcal{EL}$. Studying them is motivated by the fact that convex logics have a good chance to be computationally well-behaved: examples of convex combinations of DLs with modal logics are the tractable combination of $\mathcal{EL}$ with the CTL operator $\mathbf{E}\Diamond$ in the case where only *local* roles are allowed (Gutiérrez-Basulto, Jung, and Lutz 2012), and various combinations of the modal logic $\mathsf{S5}$ with $\mathcal{EL}$, which have been shown to be easier than the $\mathcal{ALC}$ variant (Lutz and Schröder 2010; Gutiérrez-Basulto et al. 2011). For BTDLs, we indeed identify three convex fragments, based on the temporal operator sets $\{\mathbf{E}\bigcirc\}$, $\{\mathbf{E}\Diamond\}$, and $\{\mathbf{E}\Diamond, \mathbf{A}\Box\}$. Surprisingly, we then show that these fragments exhibit very high complexity: undecidable and hard for non-elementary for the $\mathbf{E}\bigcirc$ and $\mathbf{E}\Diamond$ cases, respectively. The proofs of these results are challenging and technically involved because the absence of disjunction makes it difficult to use standard techniques for establishing lower complexity bounds, such as tilings.

Finally, we investigate BTDLs based on *DL-Lite*$^{\mathcal{N}}_{bool}$, showing that the technique developed by Artale et al. (2012) to prove decidability of LTDLs based on *DL-Lite* is robust in the sense that it can be carried over to *most* BTDLs based on CTL. We obtain tight complexity bounds, namely PSPACE- and EXPTIME-completeness.

Throughout the paper, we will omit most technical proof details which, together with additional technical notation, can be found in the long version available online: http://tinyurl.com/kr14tdl

2 Preliminaries

We introduce CTL$_{\mathcal{ALC}}$, a temporal DL based on the classical DL $\mathcal{ALC}$. The other temporal DLs studied in this paper are fragments or variants of CTL$_{\mathcal{ALC}}$, introduced later as needed. Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite sets of *concept names* and *role names*, respectively. We assume that $\mathsf{N_R}$ is partitioned into two countably infinite sets $\mathsf{N_{rig}}$ and $\mathsf{N_{loc}}$ of *rigid role names* and *local role names*, respectively. CTL$_{\mathcal{ALC}}$*-concepts* C are defined by the following grammar

$$C := \top \mid A \mid \neg C \mid C \sqcap D \mid \exists r.C \mid \mathbf{E}\bigcirc C \mid \mathbf{E}\Box C \mid \mathbf{E}(C \mathcal{U} D)$$

where A ranges over $\mathsf{N_C}$, r over $\mathsf{N_R}$. As usual, we use $\bot$ to abbreviate the concept $\neg\top$, $C \sqcup D$ for $\neg(\neg C \sqcap \neg D)$, and $\forall r.C$ for $\neg\exists r.\neg C$. We additionally have the following temporal abbreviations (Clarke, Grumberg, and Peled 1999): $\mathbf{E}\Diamond C = \mathbf{E}(\top \mathcal{U} C)$, $\mathbf{A}\Box C = \neg\mathbf{E}\Diamond\neg C$, $\mathbf{A}\Diamond C = \neg\mathbf{E}\Box\neg C$, $\mathbf{A}(C \mathcal{U} D) = \neg\mathbf{E}(\neg D \mathcal{U} (\neg C \wedge \neg D)) \wedge \neg\mathbf{E}\Box\neg D$. We moreover consider the *strict* $(\cdot^{<})$ versions of the temporal operators: $\mathbf{E}\Box^{<} C = \mathbf{E}\bigcirc\mathbf{E}\Box C$, $\mathbf{E}(C \mathcal{U}^{<} D) = \mathbf{E}\bigcirc\mathbf{E}(C \mathcal{U} D)$.

The semantics of classical, non-temporal DLs, such as $\mathcal{ALC}$ is given in terms of *interpretations* of the form $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, where Δ is a non-empty set called the *domain* and $\cdot^{\mathcal{I}}$ is an *interpretation function* that maps each $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}} \subseteq \Delta$ and each $r \in \mathsf{N_R}$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta \times \Delta$. The semantics of CTL$_{\mathcal{ALC}}$ is given in terms of temporal interpretations, which are infinite trees in which every node is associated with a classical interpretation. For the purposes of this paper, a *tree* is a directed graph $T = (W, E)$ where $W \subseteq (\mathbb{N} \setminus \{0\})^*$ is a prefix-closed non-empty set of *nodes* and $E = \{(w, wc) \mid wc \in W, w \in \mathbb{N}^*, c \in \mathbb{N}\}$ a set of *edges*; we generally assume that $wc \in W$ and $c' < c$ implies $wc' \in W$ and that E is a total relation. The node $\varepsilon \in W$ is the *root* of T. For brevity and since E can be reconstructed from W, we will usually identify T with W. In the context of temporal DLs we refer to nodes of T as *time points or worlds*.

A *temporal interpretation* is a structure $\mathfrak{I} = (\Delta, T, \{\mathcal{I}_w\}_{w \in W})$ where $T = (W, E)$ is a tree, and for each $w \in W$, $\mathcal{I}_w$ is an interpretation with domain Δ, such that $r^{\mathcal{I}_w} = r^{\mathcal{I}_{w'}}$ for all $r \in \mathsf{N_{rig}}$ and $w, w' \in W$. We usually write $A^{\mathfrak{I},w}$ instead of $A^{\mathcal{I}_w}$, and intuitively $d \in A^{\mathfrak{I},w}$ means that in the interpretation $\mathfrak{I}$, the object d is an instance of the concept name A *at time point* w. Moreover, note that we make the *constant domain assumption*, that is, all time points share the same domain Δ. Intuitively, this means that objects are not created or destroyed over time.

We now define the semantics of CTL$_{\mathcal{ALC}}$-concepts. A *path* in a tree $T = (W, E)$ starting at a node w is a minimal set $\pi \subseteq W$ such that $w \in \pi$ and for each $w' \in \pi$, there is a $c \in \mathbb{N}$ with $w'c \in \pi$. We use $\mathsf{Paths}(w)$ to denote the set of all paths starting at the node w. For a path $\pi = w_0 w_1 w_1 \cdots$ and $i \geq 0$, we use $\pi[i]$ to denote w_i. The mapping $\cdot^{\mathfrak{I},w}$ is extended from concept names to CTL$_{\mathcal{ALC}}$-concepts as follows:

$$\begin{aligned}
\top^{\mathfrak{I},w} &= \Delta; \qquad (C \sqcap D)^{\mathfrak{I},w} = C^{\mathfrak{I},w} \cap D^{\mathfrak{I},w};\\
(\exists r.C)^{\mathfrak{I},w} &= \{d \in \Delta \mid \exists e.(d,e) \in r^{\mathfrak{I},w} \wedge e \in C^{\mathfrak{I},w}\};\\
(\mathbf{E}\bigcirc C)^{\mathfrak{I},w} &= \{d \in \Delta \mid d \in C^{\mathfrak{I},\pi[1]} \text{ for some } \pi \in \mathsf{Paths}(w)\};\\
(\mathbf{E}\Box C)^{\mathfrak{I},w} &= \{d \in \Delta \mid \forall j \geq 0. d \in C^{\mathfrak{I},\pi[j]} \text{ for some } \pi \in \mathsf{Paths}(w)\};\\
(\mathbf{E}(C \mathcal{U} D))^{\mathfrak{I},w} &= \{d \in \Delta \mid \exists\, j \geq 0.(d \in D^{\mathfrak{I},\pi[j]} \wedge\\
&\qquad (\forall\, 0 \leq k < j.\, d \in C^{\mathfrak{I},\pi[k]})) \text{ for some } \pi \in \mathsf{Paths}(w)\}.
\end{aligned}$$

A *general CTL$_{\mathcal{ALC}}$-TBox* $\mathcal{T}$ is a finite set of *concept inclusions (CIs)* $C \sqsubseteq D$ with C, D CTL$_{\mathcal{ALC}}$ concepts.

A temporal interpretation $\mathfrak{I}$ is a *model* of a concept C if $C^{\mathfrak{I},\varepsilon} \neq \emptyset$; it is a *model* of $C \sqsubseteq D$, written $\mathfrak{I} \models C \sqsubseteq D$, if $C^{\mathfrak{I},w} \subseteq D^{\mathfrak{I},w}$ for all $w \in W$; it is a model of a TBox $\mathcal{T}$ if it satisfies all CIs in $\mathcal{T}$. Thus, a TBox $\mathcal{T}$ is interpreted globally in the sense that it has to be satisfied at *every* time point.

We say that a concept C is *satisfiable with respect to a CTL$_{\mathcal{ALC}}$-TBox* $\mathcal{T}$ if there is a common model of C and $\mathcal{T}$. We then consider the following *concept satisfiability problem*: given a concept C and a TBox $\mathcal{T}$, decide whether there is a common model of C and $\mathcal{T}$.

We will denote fragments of CTL$_{\mathcal{ALC}}$ by putting the available temporal operators as a superscript; for example, CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$ denotes the fragment having as the only operators available $\mathbf{E}\Diamond$ and $\mathbf{A}\Box$.

3 Undecidability of CTL combined with $\mathcal{ALC}$

We begin our study by looking at CTL$_{\mathcal{ALC}}$. To our knowledge, there are no decidability or complexity results specifically for this combination in the literature. However, there are two results about logics that are close to CTL$_{\mathcal{ALC}}$: On the one hand, it follows from results by (Hodkinson, Wolter, and Zakharyaschev 2002) that a combination of the two-variable guarded fragment of first-order logic (which subsumes $\mathcal{ALC}$) with CTL is undecidable, but that logic is more expressive than our CTL$_{\mathcal{ALC}}$: for example, it allows for CIs in the scope of temporal operators. On the other hand, there is a close correspondence between CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$ and the PML $\mathsf{S4} \times \mathsf{K}$, whose classical satisfiability problem is decidable (Gabbay et al. 2003), though hard for non-elementary time (Göller, Jung, and Lohrey 2012). This correspondence is based on the fact that $\mathsf{S4} \times \mathsf{K}$ has the tree model property and thus $\mathsf{S4} \times \mathsf{K}$ is the logic of all bi-modal Kripke frames (relational structures) that are the product of the reflexive and transitive closure of a tree ($\mathsf{S4}$) and a tree (K). Therefore, in CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$ with one rigid role and no further roles, the CTL branching time structures correspond to the $\mathsf{S4}$-dimension and $\mathcal{ALC}$ corresponds to K.

The correspondence with PMLs, however, is only of restricted use for our study of the concept satisfiability problem *relative to global TBoxes:* this problem corresponds to the global consequence problem in modal logic which, to the best of our knowledge, has not been studied for $\mathsf{S4} \times \mathsf{K}$. Therefore, the above result for classical satisfiability cannot easily be transferred to CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$. We will in fact show that concept satisfiability relative to global TBoxes in CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$ is undecidable, exploiting undecidability of satisfiability for a family of transitive PMLs which includes $\mathsf{S4} \times \mathsf{K4}$ (Gabelaia et al. 2005). We prove that the global consequence problem of $\mathsf{S4} \times \mathsf{K}$ can encode the satisfiability problem of $\mathsf{S4} \times \mathsf{K4}$, using a technique by Tobies (2001).

We first rephrase a consequence of Gabelaia et al.'s result in BTDL notation and will then sketch how to employ Tobies's technique to establish undecidability of CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$.

Theorem 1 (Gabelaia et al. 2005) *The following problem is undecidable for $\mathcal{L} \in \{CTL^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}, CTL^{\mathbf{E}\Diamond^<,\mathbf{A}\Box^<}_{\mathcal{ALC}}\}$. Given an $\mathcal{L}$-concept C that uses at most one rigid role r and no other roles, is C satisfiable in a temporal interpretation $\mathfrak{I} = (\Delta, T, \{\mathcal{I}_w\}_{w \in W})$ where $r^{\mathfrak{I},w}$ is transitive for all w ?*

We sketch how to employ Tobies's technique to transfer this result to our standard concept satisfiability problem for CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$ and CTL$^{\mathbf{E}\Diamond^<,\mathbf{A}\Box^<}_{\mathcal{ALC}}$.

Theorem 2 *Concept satisfiability w.r.t. general TBoxes for CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$ and CTL$^{\mathbf{E}\Diamond^<,\mathbf{A}\Box^<}_{\mathcal{ALC}}$ with rigid roles is undecidable.*

Proof. We only sketch the proof for CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$; the strict variant can be treated using the same arguments. Consider a CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$-concept C that uses at most one rigid role r and no other roles, and assume that C is in negation normal form (NNF), i.e., negation occurs only in front of atomic concepts in C. We use $\mathsf{cl}(C)$ to denote the set of NNFs of all subconcepts of C and of their negations, and $\sim D$ to denote the NNF of $\neg D$ for any $D \in \mathsf{cl}(C)$. For every $\forall r.D \in \mathsf{cl}(C)$, reserve a fresh concept name X_D. We define a translation $\cdot^{\mathsf{tr}}$ from CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$-concepts to CTL$^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$-concepts.

$$\begin{aligned}
A^{\mathsf{tr}} &= A^{\mathsf{tr}} & (\forall r.D)^{\mathsf{tr}} &= X_D\\
(\neg A)^{\mathsf{tr}} &= \neg A^{\mathsf{tr}} & (\exists r.D)^{\mathsf{tr}} &= \neg X_{\sim D}\\
(D_1 \sqcap D_2)^{\mathsf{tr}} &= D_1^{\mathsf{tr}} \sqcap D_2^{\mathsf{tr}} & (\mathbf{E}\Diamond D)^{\mathsf{tr}} &= \mathbf{E}\Diamond D\\
(D_1 \sqcup D_2)^{\mathsf{tr}} &= D_1^{\mathsf{tr}} \sqcup D_2^{\mathsf{tr}} & (\mathbf{A}\Box D)^{\mathsf{tr}} &= \mathbf{A}\Box D
\end{aligned}$$

Let $\mathcal{T}_C$ be the following TBox.

$$\mathcal{T}_C = \{X_D \equiv \forall r.D^{\mathsf{tr}},\ X_D \sqsubseteq \forall r.X_D \mid \forall r.D \in \mathsf{cl}(C)\}$$

It suffices to show that C is satisfiable in an interpretation that makes r transitive everywhere iff C^{tr} is satisfiable w.r.t. $\mathcal{T}_C$ in $\mathrm{CTL}^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$. For the "$\Rightarrow$" direction, take a model $\mathfrak{I}$ of C with r being transitive, and additionally interpret X_D as $\forall r.D$. Then a standard induction shows that the modified $\mathfrak{I}$ satisfies C^{tr}, and transitivity of r ensures that it is a model of $\mathcal{T}_C$. For the "$\Leftarrow$" direction, take a model $\mathfrak{J}$ of C^{tr} and $\mathcal{T}_C$ and extend the interpretation of r in every world to its transitive closure. Then $\mathfrak{J}$ being a model of $\mathcal{T}_C$ can be used to show that the modified $\mathfrak{J}$ still satisfies C. ❑

In fact, we are able to prove a more general result of independent interest in the field of PMLs. All notation used in the following theorem is standard for PMLs (Gabbay et al. 2003), except for $\mathfrak{C}^+$, which refers to the class of all transitive frames in the frame class $\mathfrak{C}$.

Theorem 3 *For any classes $\mathfrak{C}_1, \mathfrak{C}_2$ of frames, where $\mathfrak{C}_2$ contains only transitive frames and both $\mathfrak{C}_1^+, \mathfrak{C}_2$ contain frames of arbitrarily large finite or infinite depth, the global satisfiability problem for $\mathsf{Log}(\mathfrak{C}_1 \times \mathfrak{C}_2)$ is undecidable.*

4 CTL combined with $\mathcal{EL}$

In this section, we consider fragments of $\mathrm{CTL}_{\mathcal{EL}}$, the fragment of $\mathrm{CTL}_{\mathcal{ALC}}$ that disallows the constructor $\neg$ (and thus also the abbreviations $C \sqcup D$, $\forall r.C$, $\mathbf{A}\Box$, ...). As an example, consider the following $\mathrm{CTL}_{\mathcal{EL}}$-TBox:

$$\exists \mathsf{hasAutoDis.Diabetes} \sqsubseteq \mathbf{E}\Diamond(\exists \mathsf{develops.Glaucoma} \sqcap \mathbf{E}\Diamond \exists \mathsf{hasTreat.EyeDrops})$$
$$\mathbf{E}\Diamond \exists \mathsf{hasTreat.EyeDrops} \sqsubseteq \mathbf{E}\Diamond \exists \mathsf{hasEffect.ChangeEyeColor}$$

It says that everyone with diabetes may eventually develop a glaucoma which may eventually be treated using eye drops, and that such a treatment may lead to a change in eye color.

Because of the absence of negation, satisfiability in $\mathrm{CTL}_{\mathcal{EL}}$ is trivial; as in non-temporal $\mathcal{EL}$, we therefore consider subsumption as the central reasoning problem. Formally, a concept D *subsumes* a concept C w.r.t. a $\mathrm{CTL}_{\mathcal{EL}}$ TBox $\mathcal{T}$, written $\mathcal{T} \models C \sqsubseteq D$, if $C^{\mathfrak{I}} \subseteq D^{\mathfrak{I}}$ for all temporal interpretations $\mathfrak{I}$ that are a model of $\mathcal{T}$. For example, the TBox above implies that $\exists \mathsf{hasAutoDis.Diabetes} \sqsubseteq \mathbf{E}\Diamond \exists \mathsf{hasEffect.ChangeEyeColor}$. We then consider the following *concept subsumption problem:* given concepts C, D and a $\mathrm{CTL}_{\mathcal{EL}}$ TBox $\mathcal{T}$, decide whether $\mathcal{T} \models C \sqsubseteq D$.

In the rest of the section we focus on the study of several fragments of $\mathrm{CTL}_{\mathcal{EL}}$. In this context, we view each of the operators $\mathbf{E}\bigcirc$, $\mathbf{E}\Diamond$, $\mathbf{E}\Box$, $\mathbf{A}\Diamond$, $\mathbf{A}\Box$, $\mathbf{E}\mathcal{U}$ and their strict versions as primitive instead of as an abbreviation.

4.1 Non-Convex Fragments

We next show undecidability of certain fragments of $\mathrm{CTL}_{\mathcal{EL}}$ using techniques developed in the context of *linear-time* temporal extensions of $\mathcal{EL}$ (Artale et al. 2007). In particular, the non-convexity of these extensions, and therefore their capability to 'express' disjunction, is exploited to show that they are as complex as the corresponding $\mathcal{ALC}$ variant.

Before we present the undecidability results we recall the standard notion of convexity.

Definition 1 *A logic $\mathcal{L}$ is called* convex *if, for all $\mathcal{L}$-TBoxes $\mathcal{T}$ and all $\mathcal{L}$-concepts $C, D_1, \ldots, D_n$, $n \geq 2$, whenever $\mathcal{T} \models C \sqsubseteq D_1 \sqcup \cdots \sqcup D_n$, then $\mathcal{T} \models C \sqsubseteq D_i$ for some $i \in \{1, \ldots, n\}$.*

Theorem 4 *Concept subsumption w.r.t. general TBoxes is undecidable for*

(a) $\mathrm{CTL}^{\mathbf{E}\Diamond,\mathbf{A}\Diamond}_{\mathcal{EL}}$ (b) $\mathrm{CTL}^{\mathbf{E}\Diamond,\mathbf{E}\bigcirc}_{\mathcal{EL}}$ (c) $\mathrm{CTL}^{\mathbf{E}\mathcal{U}}_{\mathcal{EL}}$
(d) $\mathrm{CTL}^{\mathbf{E}\mathcal{U}^<}_{\mathcal{EL}}$ (e) $\mathrm{CTL}^{\mathbf{E}\Diamond^<,\mathbf{E}\Box^<}_{\mathcal{EL}}$ (f) $\mathrm{CTL}^{\mathbf{E}\Diamond,\mathbf{E}\Box}_{\mathcal{EL}}$

Proof. The proof is in two steps. First, we show the non-convexity (in opposition to convexity) of these fragments of $\mathrm{CTL}_{\mathcal{EL}}$. Second, we reduce from satisfiability w.r.t. TBoxes in the corresponding $\mathrm{CTL}_{\mathcal{ALC}}$ fragment.

Lemma 5 *The fragments* (a)–(f) *of $\mathrm{CTL}_{\mathcal{EL}}$ are non-convex.*

Proof. The non-convexity of (a)–(d) was previously established in (Gutiérrez-Basulto, Jung, and Lutz 2012), for the witnesses see Figure 1. In the same place, the fragments (e) and (f) were stated to be convex, too; however, we show here that this is not the case. Interestingly, this shows that the convexity of $\mathrm{CTL}^{\mathbf{E}\Diamond}_{\mathcal{EL}}$, which we show later (Theorem 6), is rather fragile since $\mathbf{E}\Diamond$ and $\mathbf{E}\Box$ allow only for existential quantification and there is no possibility of talking about 'next' time points, which in principle might make us think of $\mathrm{CTL}^{\mathbf{E}\Diamond,\mathbf{E}\Box}_{\mathcal{EL}}$ as a temporal analog of $\mathcal{EL}$. Alas, the presence of both operators makes it non-convex. We begin by presenting the witnesses used to prove non-convexity and their intuition.

To show non-convexity of $\mathrm{CTL}^{\mathbf{E}\Diamond,\mathbf{E}\Box}_{\mathcal{EL}}$ we use the following witness.

$$\mathcal{T} = \{C \sqsubseteq \mathbf{E}\Diamond(D \sqcap A),\ D \sqsubseteq \mathbf{E}\Diamond(C \sqcap A)\} \text{ and}$$
$$D_1 = \mathbf{E}\Diamond(C \sqcap D),\ D_2 = \mathbf{E}\Box\mathbf{E}\Diamond A.$$

Intuitively, $\mathcal{T}$ stipulates that every instance d of C has the possibility of eventually being an instance of D and A, and thereafter has the possibility of eventually being an instance of C and A, but there is a choice on whether d becomes an instance of C and D at the same time point, or whether d infinitely alternates between C and D.

Although the disjunction expressed in $\mathrm{CTL}^{\mathbf{E}\Diamond,\mathbf{E}\Box}_{\mathcal{EL}}$ relies on the non-strict interpretation of $\mathbf{E}\Diamond$ and $\mathbf{E}\Box$, the following shows that the strict version is also non-convex.

$$\mathcal{T} = \{C \sqsubseteq \mathbf{E}\Diamond^< B,\ B \sqsubseteq D \sqcap \mathbf{E}\Box^< D\} \text{ and}$$
$$D_1 = \mathbf{E}\Diamond^<\mathbf{E}\Diamond^< B,\ D_2 = \mathbf{E}\Box^< D$$

Intuitively, every instance d of C has two choices on becoming an instance of B for the first time, namely, either it does it at the immediate successor, or at some point after it.

Formally, in both cases we show non-convexity by proving that $\mathcal{T} \models C \sqsubseteq \bigsqcup D_i$, but $\mathcal{T} \not\models C \sqsubseteq D_i$. ❑

Having established the non-convexity of the above logics we can devise a reduction from the satisfiability problem w.r.t. TBoxes in the corresponding $\mathcal{ALC}$ variant. First, recall that $\mathrm{CTL}^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{ALC}}$ and $\mathrm{CTL}^{\mathbf{E}\Diamond^<,\mathbf{A}\Box^<}_{\mathcal{ALC}}$ are undecidable (Theorem 2). Note moreover that all fragments above contain at least the $\mathbf{E}\Diamond$ or $\mathbf{E}\Diamond^<$ operator. Hence all the $\mathcal{ALC}$ variants of these fragments are undecidable.

$\mathbf{E}\Diamond\mathbf{A}\Diamond$	$\mathcal{T}=\emptyset,\ C=\mathbf{A}\Diamond A\sqcap\mathbf{A}\Diamond B$ $D_1=\mathbf{E}\Diamond(A\sqcap\mathbf{E}\Diamond B),\ D_2=\mathbf{E}\Diamond(B\sqcap\mathbf{E}\Diamond A)$
$\mathbf{E}\Diamond\mathbf{E}\bigcirc$	$\mathcal{T}=\emptyset,\ C=\mathbf{E}\Diamond A,\ D_1=A,\ D_2=\mathbf{E}\bigcirc\mathbf{E}\Diamond A$
$\mathbf{E}\mathcal{U}$	$\mathcal{T}=\emptyset,\ C=\mathbf{E}(A\mathcal{U}B),\ D_1=B,\ D_2=A$
$\mathbf{E}\mathcal{U}^<$	$\mathcal{T}=\emptyset,\ C=\mathbf{E}(A\mathcal{U}^<B),\ D_1=\mathbf{E}(A\mathcal{U}^<A),$ $D_2=\mathbf{E}(B\mathcal{U}^<B)$

Figure 1: Non-convexity witnesses

The crucial point of the reduction is the elimination of disjunction, that is, encoding CIs $C\sqsubseteq A\sqcup B$ present in the $\mathcal{ALC}$ variants. To this aim, we use the non-convexity witnesses (*cf.* witnesses presented above and Fig. 1) which, as discussed, allow to describe 'choices'. ❑

4.2 Convex Fragments

In this section, we will show that the logics $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$, $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$, and $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond\mathbf{A}\Box}$ are convex, which suggests that they have limited expressive power. Indeed, convex logics are typically computationally well-behaved (Oppen 1980; Baader, Brandt, and Lutz 2005; Lutz and Wolter 2012). Our convexity result is contrasted by two observations: first, the linear-time TDL based on $\Diamond$ and $\mathcal{EL}$ with rigid roles is non-convex (Artale et al. 2007) which, intuitively, is due to the linear structure of time. Second, we will show later that $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ is undecidable because its expressive power is sufficient to encode the halting problem for two-counter machines, which is equivalent to that of Turing machines.

Theorem 6 *$\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$, $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$, and $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond,\mathbf{A}\Box}$ are convex.*

Proof. (sketch) We start with $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ and first observe that $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$, if restricted to a single role that is rigid, is a notational variant of a fragment of the product modal logic $\mathsf{KD}\times\mathsf{K}$, where K is the basic (uni-)modal logic and KD is the modal logic of the class of frames where every state has a successor. We can therefore use standard results and machinery from product modal logics – namely unravelling tolerance, the standard translation to first-order logic, and first-order axiomatizations of product frames – to achieve two things: we will translate $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$-TBoxes and -concepts into a fragment of first-order logic that generalizes the Horn fragment, and we will axiomatize the rigidity of roles and the properties of the temporal successor relation in the same fragment. This will allow us to encode logical consequence in $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$, even with multiple non-rigid roles, as first-order formulas in this fragment. Using the fact that such formulas are preserved under a standard operation on structures called *direct product*, it will then be easy to establish convexity.

In the following, we sketch some details of this argument. Consider a $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$-TBox $\mathcal{T}$ and $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$-concepts $C, D_1, \dots, D_n$, $n\geq 2$. Let $r_1,\dots,r_k$ be the rigid roles occurring in $\mathcal{T}, C, D_1, \dots, D_n$, and let $r_{k+1},\dots,r_\ell$ be the non-rigid roles. We denote by $\mathsf{ST}(\mathcal{T})$ the standard translation of $\mathcal{T}$ into first-order logic. Assuming $\mathcal{T}$ to be in a normal form that extends the usual one for $\mathcal{EL}$ (cf. Baader, Brandt, and Lutz 2005), it is easy to see that $\mathsf{ST}(\mathcal{T})$ is equivalent to a conjunction of embedded implicational dependencies (EIDs) (Fagin 1982), i.e., to a conjunction of FO-sentences

$$\forall x_1\dots x_m.((E_1\wedge\dots\wedge E_n)\rightarrow\exists x_{m+1}\dots x_k.(F_1\wedge\dots\wedge F_\ell))$$

where each E_i is a relational formula $Px_{j_1}\dots x_{j_d}$ and each F_i is either a relational formula or an equality $x_j=x_{j'}$.

Furthermore, we need to express rigidity of the roles $r_1,\dots,r_k$ as well as the property that the temporal "direct successor" relation is a total tree. For rigidity, we borrow from the theory of product modal logics (Gabbay et al. 2003): we have to say that each pair (R_i, S) of binary predicates representing the rigid role r_i and the temporal successor relation is left-commutative, right-commutative and satisfies the Church-Rosser property. For example, left-commutativity is given by the following EID.

$$\mathsf{LC}=\textstyle\bigwedge_{i=1}^{k}\forall xyz(R_ixy\wedge Syz\rightarrow\exists u(Sxu\wedge R_iuz))$$

The other two properties can be expressed via analogous EIDs RC and CRP. For enforcing the total tree, we only need to say that S is total because standard translations of $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$-TBoxes are preserved under unravelling and under taking point-generated substructures. Totality of S can be expressed by the EID $\mathsf{D}=\forall x\exists y(Sxy)$.

We now consider the conjunction

$$\varphi(\mathcal{T})=\mathsf{LC}\wedge\mathsf{RC}\wedge\mathsf{CRP}\wedge\mathsf{D}\wedge\mathsf{ST}(\mathcal{T})$$

which, by classical results (Chang and Keisler 1990; Fagin 1982) is preserved under *direct products*: If n structures $\mathcal{M}_1,\dots,\mathcal{M}_n$ satisfy an EID, then so does their direct product, which has as its domain the cross-product of the domains of $\mathcal{M}_1,\dots,\mathcal{M}_n$ and interprets every m-ary predicate P as the set of m-tuples whose projections to the domain of each $\mathcal{M}_i$ are in the interpretation of P.

In order to show convexity of $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ via the contrapositive, we observe that the statement

$$\mathcal{T}\not\models C\sqsubseteq D_i\text{ for all }i\ \text{ implies }\ \mathcal{T}\not\models C\sqsubseteq D_1\sqcup\dots\sqcup D_n$$

is equivalent to

$$\varphi(\mathcal{T})\not\models\forall x.(Cx\rightarrow D_ix)\text{ for all }i\ \text{ implies}$$
$$\varphi(\mathcal{T})\not\models\forall x.(Cx\rightarrow D_1x\vee\dots\vee D_nx).$$

The claim of this property can be established easily, consulting the direct product of the n structures witnessing the non-entailments in the hypotheses.

To carry this reasoning over to $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$ and $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond\mathbf{A}\Box}$, we replace totality in the above argument with reflexivity and transitivity. Consequently, we have to replace the conjunct D in $\varphi(\mathcal{T})$ with the conjunction of the EIDs $\mathsf{T}=\forall x(xR_hx)$ and $\mathsf{4}=\forall xyz(xR_hy\wedge yR_hz\rightarrow xR_hz)$. Unfortunately, $\mathsf{ST}(\mathcal{T})$ now contains translations of axioms containing $\mathbf{A}\Box$, which are not generally equivalent to EIDs. Using additional arguments, however, we can show that $\mathsf{ST}(\mathcal{T})$ for these two

logics is then preserved under direct products *of structures where the relation S representing the temporal successor relation is interpreted as a total relation*. Therefore, the above $\varphi(\mathcal{T})$ remains preserved under direct products of such structures. Since $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$ and $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond\mathbf{A}\Box}$ are restricted to such structures, the above argument goes through.

It is worth recalling that convexity of both $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ and $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$ does not imply that $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc,\mathbf{E}\Diamond}$ is convex: in order to translate a TBox using both the $\mathbf{E}\bigcirc$ and the $\mathbf{E}\Diamond$ operator into FO, we would have to use two binary relations $S_{\mathbf{E}\bigcirc}$ and $S_{\mathbf{E}\Diamond}$ and state that $S_{\mathbf{E}\Diamond}$ is the reflexive transitive closure of $S_{\mathbf{E}\bigcirc}$, which is not expressible in FO. ❑

We continue our study by showing that $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ lacks the *finite model property (FMP)*: consider the TBox $\mathcal{T}$ consisting of the following concept inclusions

$$A \sqsubseteq \exists r.A, \quad A \sqsubseteq \mathbf{E}\bigcirc B, \quad \exists r.B \sqsubseteq B',$$
$$\exists r.B' \sqsubseteq B', \qquad \mathbf{E}\bigcirc(B \sqcap B') \sqsubseteq C, \qquad \exists r.C \sqsubseteq C,$$

with r a rigid role. It is not hard to see that $\mathcal{T} \not\models A \sqsubseteq C$ but for every finite model $\mathfrak{I}$ of $\mathcal{T}$, $\mathfrak{I} \models A \sqsubseteq C$. In fact, we show that this also holds if the two occurrences of $\mathbf{E}\bigcirc$ in $\mathcal{T}$ are replaced by $\mathbf{E}\Diamond$. Therefore, the following holds.

Theorem 7 *$\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ and $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$ lack the FMP.*

Interestingly, in the same way the TBox presented above enforces a temporal tree with infinite branching degree, relying on rigid roles.

We next study the complexity of subsumption relative to general TBoxes in $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ and $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$. In Theorem 2, we already showed that the $\mathcal{ALC}$ variant of the latter is undecidable. For the former, observe that the corresponding $\mathcal{ALC}$-version is a notational variant of the product logic $\mathsf{K} \times \mathsf{K}$, for which the global consequence problem (and thus reasoning relative to global TBoxes) is known to be undecidable.

We will prove undecidability for $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$ and a nonelementary lower bound for $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\Diamond}$, thus showing that even the convex fragments have a high complexity. These high complexity results are surprising because convex logics often exhibit better computational properties. Specifically, the lack of expressiveness prohibits reductions that were previously used to show undecidability particularly for product modal logics, for instance reductions from tiling problems: without disjunction we cannot express that in some point of the grid one of the tile types has to be true. However, we show how to exploit rigidity of the roles for proving these results. Let us remark that the proven lower bounds remain valid when we adopt the *expanding domain assumption* (Gabbay et al. 2003); in particular, the same proofs work.

4.3 Undecidability with the Next Operator

We proceed to show undecidability of the combination of $\mathcal{EL}$ with $\mathbf{E}\bigcirc$.

Theorem 8 *Concept subsumption w.r.t. general $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$-TBoxes is undecidable.*

We present a reduction from the reachability problem for *two-counter machines (TCM)*. A TCM is a pair (Q, P) with states $Q = \{0, \dots, n\}$, where 0 is the *initial state* and n is the *halting state*, and corresponding instructions $P = \{I_0, \dots, I_{n-1}\}$. Additionally, there are two *registers* 1 and 2. For all $i < n$, we have that

- either $I_i = \mathsf{inc}(c, j)$ is an *incrementation instruction* with $c \in \{1, 2\}$ a register and j the subsequent state;
- or $I_i = \mathsf{dec}(c, j, k)$ is a *decrementation instruction* with $c \in \{1, 2\}$ a register and j the subsequent state if register c contains value 0 and k the subsequent state otherwise.

A *configuration* is a triple (i, n, m) where i is the current state and n and m are the contents of registers 1 and 2, respectively. We write $(i, n_1, n_2) \Rightarrow_M (j, m_1, m_2)$ if one of the following holds:

- $I_i = \mathsf{inc}(c, j)$, $m_c = n_c + 1$ and $m_{\overline{c}} = n_{\overline{c}}$ (where $\overline{c} = 3 - c$);
- $I_i = \mathsf{dec}(c, j, k)$, $m_c = n_c = 0$ and $m_{\overline{c}} = n_{\overline{c}}$;
- $I_i = \mathsf{dec}(c, k, j)$, $n_c > 0$, $m_c = n_c - 1$, and $m_{\overline{c}} = n_{\overline{c}}$.

The *computation of M* is the unique longest configuration sequence $(p_0, n_0, m_0) \Rightarrow_M (p_1, n_1, m_1) \Rightarrow_M \dots$ such that $p_0 = n_0 = m_0 = 0$. The *halting problem for TCM*, i.e., to decide whether a given TCM M reaches state n, is undecidable (Minsky 1967).

For our purposes, it is convenient to define a variant of the halting problem. Fix some TCM $M = (Q, I)$ with $Q = \{0, \dots, n\}$. An *extended state (e-state) x for M* is either $x = q$ for some $q \in Q$ with I_q an increment instruction or a pair $x = (q, f)$ with $f \in \{0, +\}$ and I_q a decrement instruction. Denote with $\overline{x}$ the state q if $x = q \in Q$ or $x = (q, f)$. A finite sequence of e-states $x_0, \dots, x_k$ is called *halting* if $x_k = n$ and there are numbers $n_0, m_0, \dots, n_k, m_k$ such that $x_0 = n_0 = m_0 = 0$ and $(\overline{x_i}, n_i, m_i) \Rightarrow_M (\overline{x_{i+1}}, n_{i+1}, m_{i+1})$ for all i, and additionally, if $x_i = (q, 0)$ and $I_q = \mathsf{dec}(c, \cdot, \cdot)$ then either $c = 1$ and $n_i = 0$ or $c = 2$ and $m_i = 0$, and if $x_i = (q, +)$ and $I_q = \mathsf{dec}(c, \cdot, \cdot)$, then either $c = 1$ and $n_i > 0$ or $c = 2$ and $m_i > 0$. It is easy to verify that M halts iff there is a halting sequence of e-states.

For the reduction let M be as above and assume without loss of generality that I_0 is an inc-instruction. Moreover, define a set of rigid roles, consisting of one role for each e-state:

$$\mathsf{ROL} = \{r_i \mid I_i = \mathsf{inc}(\cdot, \cdot)\} \cup \{r_{i0}, r_{i+} \mid I_i = \mathsf{dec}(\cdot, \cdot, \cdot)\}.$$

Finally, introduce concept names $Q_0, \dots, Q_n$ representing the states of M, for each $c \in \{1, 2\}$ concepts C_c marking elements that contribute to the value in register c and concepts Tail_c and Head_c that mark the bounds of register c.

We construct a $\text{CTL}_{\mathcal{EL}}^{\mathbf{E}\bigcirc}$-TBox $\mathcal{T}_M$ and identify two concept names A, B such that:

Lemma 9 *$\mathcal{T}_M \models A \sqsubseteq B$ iff there is a halting sequence of e-states.*

The idea is (i) to generate all sequences of e-states for M using the roles in ROL and (ii) to check if one of them is halting. Point (i) is done using the concept inclusions

$$A \sqsubseteq \exists r_0.(S \sqcap A), \qquad A \sqsubseteq \exists r.A \text{ for all } r \in \mathsf{ROL} \setminus \{r_0\}$$

where the first inclusion additionally ensures that after creating an r_0 successor, some special concept name S is satisfied. Intuitively, domain elements satisfying A in some world stipulate an infinite tree where the path to each node corresponds to *the reverse of a sequence of e-states*. For example, the branch $r_{3+}r_{20}r_{2+}r_1r_0$ of such a tree corresponds to the sequence $0, 1, (2, +), (2, 0), (3, +)$ of e-states for M.

For achieving Point (ii), we simulate the behavior of M *backwards* over the tree mentioned above. A configuration (i, n_1, n_2) of M will be represented in the temporal dimension by a time point that has a (time) successor for each register: one successor labeled with Head_1 and another with Head_2. Both successors have some (not necessarily direct) successor that is labeled with Tail_c and all worlds between Tail_c and Head_c are labeled with C_c. Further, the distance between Tail_c and Head_c is precisely the value n_c of the register c. Moreover, all these worlds are labeled with the current state Q_i.

Note that a valid computation of M always starts in state 0 and that by the above CI all corresponding nodes are labeled with S. Thus, we can use

$$S \sqsubseteq \mathbf{E}\bigcirc\bigsqcap_{c\in\{1,2\}}\mathbf{E}\bigcirc(\mathsf{Head}_c \sqcap \mathsf{Tail}_c \sqcap Q_0)$$

in order to enforce (the encoding of) an initial configuration of M starting in every node labeled with S. The begin of the counter never changes along a computation:

$$\exists r.\mathsf{Head}_c \sqsubseteq \mathsf{Head}_c \text{ for all } r \in \mathsf{ROL}, c \in \{1, 2\}.$$

It remains to describe how the ends Tail_c, the states Q_i, and the intermediate labels C_c are updated according to the instructions I. For instructions $I_i = \mathsf{inc}(c, j)$, we add the inclusions

$$\begin{aligned}
\exists r_i.(Q_i \sqcap \mathsf{Tail}_c) &\sqsubseteq Q_j \sqcap C_c \sqcap \mathbf{E}\bigcirc(Q_j \sqcap \mathsf{Tail}_c)\\
\exists r_i.(Q_i \sqcap C_c) &\sqsubseteq Q_j \sqcap C_c\\
\exists r_i.(Q_i \sqcap \mathsf{Tail}_{\overline{c}}) &\sqsubseteq Q_j \sqcap \mathsf{Tail}_{\overline{c}}\\
\exists r_i.(Q_i \sqcap C_{\overline{c}}) &\sqsubseteq Q_j \sqcap C_{\overline{c}}
\end{aligned}$$

For instructions $I_i = \mathsf{dec}(c, j, k)$, add the inclusions

$$\begin{aligned}
\exists r_{i0}.(Q_i \sqcap \mathsf{Head}_c \sqcap \mathsf{Tail}_c) &\sqsubseteq Q_j \sqcap \mathsf{Tail}_c\\
\exists r_{i0}.(Q_i \sqcap C_{\overline{c}}) &\sqsubseteq Q_j \sqcap C_{\overline{c}}\\
\exists r_{i0}.(Q_i \sqcap \mathsf{Tail}_{\overline{c}}) &\sqsubseteq Q_j \sqcap \mathsf{Tail}_{\overline{c}}\\
\exists r_{i+}.(Q_i \sqcap C_c \sqcap \mathbf{E}\bigcirc\mathsf{Tail}_c) &\sqsubseteq Q_k \sqcap \mathsf{Tail}_c\\
\exists r_{i+}.(Q_i \sqcap C_c \sqcap \mathbf{E}\bigcirc C_c) &\sqsubseteq Q_k \sqcap C_c\\
\exists r_{i+}.(Q_i \sqcap C_{\overline{c}}) &\sqsubseteq Q_k \sqcap C_{\overline{c}}\\
\exists r_{i+}.(Q_i \sqcap \mathsf{Tail}_{\overline{c}}) &\sqsubseteq Q_k \sqcap \mathsf{Tail}_{\overline{c}}
\end{aligned}$$

Note that in order to correctly update a configuration along some role r_i or r_{if}, $f \in \{0, +\}$, it is required that (1) the state of the configuration is in fact i, i.e., labeled with Q_i, and (2) for decrementation instructions $\mathsf{dec}(c, \cdot, \cdot)$ the mentioned conditions are satisfied. For example, if $f = 0$, then the value currently encoded in register c has to be zero. If this is not the case, none of the above CIs for register c can be applied; thus, no information is propagated for register c and the sequence under consideration does not correspond to a valid computation.

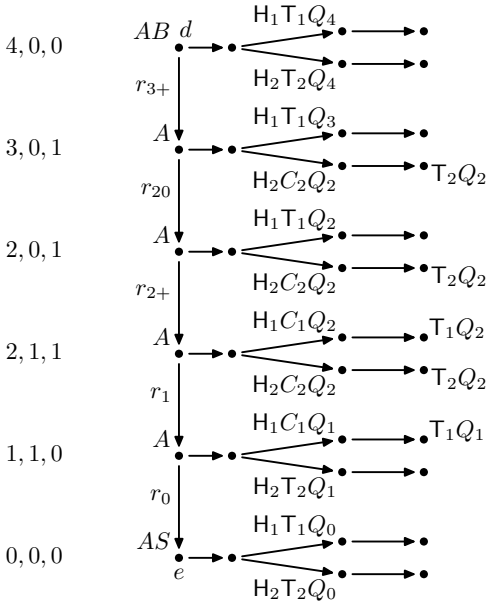


Figure 2: Example computation for $\mathcal{T}_M$.

Finally, a halting configuration can be detected by checking whether the encoding of some configuration has label Q_n in both registers. This is expressed by the CI

$$\mathbf{E}\bigcirc(\mathbf{E}\bigcirc(\mathsf{Head}_1 \sqcap Q_n) \sqcap \mathbf{E}\bigcirc(\mathsf{Head}_2 \sqcap Q_n)) \sqsubseteq B$$

in order to label accepting configurations with B.

Example 1 *Let $M = (Q, I)$ be a TCM with $Q = \{0, \ldots, 4\}$ and $I_0 = \mathsf{inc}(1, 1)$, $I_1 = \mathsf{inc}(2, 2)$, and $I_2 = \mathsf{dec}(1, 3, 2)$, $I_3 = \mathsf{dec}(2, 2, 4)$. It is easy to check that M halts and that the witnessing sequence of e-states is $0, 1, (2, +), (2, 0), (3, +), 4$. The construction of $\mathcal{T}_M$ implies that for any instance d of A, there is another domain element that is reachable via the sequence of roles $r_{3+}, r_{20}, r_{2+}, r_1, r_0$. This sequence is depicted in Figure 2. Note that by $\mathcal{T}_M$, e is labeled with S and starting from e we find the encoding of the initial configuration of M (the temporal relation is drawn horizontally and H, T abbreviate $\mathsf{Head}, \mathsf{Tail}$). $\mathcal{T}_M$ induces the configurations from bottom to top. The encoding of the configurations is found in the right side of the picture; the corresponding configurations are depicted on the left side. It can be seen that the top-most configuration is actually halting and enforces finally label B in point d. Note that only the consequences of* one *sequence of e-states are depicted. In particular, in a full model, all instances of A are also instances of B.*

4.4 Hardness for Non-Elementary Time with the Transitive Future Operator

We begin by noting that the reduction presented in the previous section does not easily transfer to the case when we consider the transitive future operator $\mathbf{E}\Diamond$ instead of the next

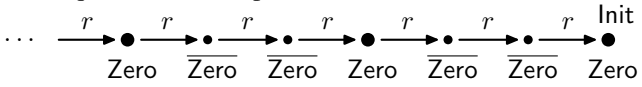
operator $\mathbf{E}\bigcirc$. The main problem is that we cannot update the concept Tail_c for a decrement instruction because the fact *'there is a future with* Tail_c *but no* C_c *until then'* is not expressible. This does not come as a surprise since it is known that the impossibility of talking about next time points brings considerable technical difficulties. For example, for proving lower bounds for products of transitive logics quite intricate techniques are required (Gabelaia et al. 2005).

In this section, we show that also the combination of $\mathbf{E}\Diamond$ with $\mathcal{EL}$ exhibits high computational complexity.

Theorem 10 *Concept subsumption w.r.t. general* $\mathrm{CTL}^{\mathbf{E}\Diamond}_{\mathcal{EL}}$*-TBoxes is inherently nonelementary.*

In order to prove the non-elementary lower bound, we use a technique in the style of (Stockmeyer 1974), which has already been used in the context of many-dimensional modal logics; for instance, for concept satisfiability for $\mathrm{LTL}_{\mathcal{ALC}}$ *without* TBoxes (Gabbay et al. 2003). In a nutshell, this technique consists of two steps: the first step is to find some encoding for numbers and show that arbitrarily big elementary numbers can be 'enforced'. In the second step, a standard reduction is used to show, e.g., k-EXPTIME-hardness of the problem for every $k \geq 1$.

We will proceed along these lines. For step 1, the encoding of a number n, we will provide a TBox $\mathcal{T}$ such that in any model of $\mathcal{T}$ containing a sequence of domain elements connected via a rigid role r the following holds. If some element of the sequence, say d, is an instance of the concept name Init, then every element that can reach d via $k \cdot n$ steps along r is an instance of a concept name Zero, and all other elements are instances of a concept name $\overline{\mathsf{Zero}}$. For $n = 3$, this sequence can be depicted as follows.



Formally, we say that a TBox $\mathcal{T}$ *counts modulo* n if there exist concept names Init, Zero, $\overline{\mathsf{Zero}}$, Fail, and a rigid role r such that for all $i \geq 0$ there are models $\mathfrak{I}$ of $\mathcal{T}$ and $\exists r^i.\mathsf{Init}$[1] with $\mathsf{Fail}^{\mathfrak{I},w} = \emptyset$ for all worlds w, and in all such models $\mathfrak{I}$ it holds for all $i \geq 0$:

(i) $\mathfrak{I} \models \exists r^i.\mathsf{Init} \sqsubseteq \mathsf{Zero}$ iff $i \equiv 0 \bmod n$;

(ii) $\mathfrak{I} \models \exists r^i.\mathsf{Init} \sqsubseteq \overline{\mathsf{Zero}}$ iff $i \not\equiv 0 \bmod n$.

Intuitively, the concept Fail is necessary as negation is not available; it is used to ensure that complementary concept names behave in a complementary way. The use of Fail is straightforward, so in the proof sketches below we will concentrate on points (i) and (ii).

We next show that there are 'small TBoxes that count modulo large numbers'. For making this precise, let us introduce the function $\mathsf{exp} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$:

$$\mathsf{exp}(1, n) := 2^n; \quad \mathsf{exp}(\ell + 1, n) := \mathsf{exp}(\ell, n) \cdot 2^{\mathsf{exp}(\ell, n)}.$$

Lemma 11 *For each* $k, n \geq 1$*, there is a polynomially sized (in* k, n*)* $\mathrm{CTL}^{\mathbf{E}\Diamond}_{\mathcal{EL}}$*-TBox* $\mathcal{T}_{k,n}$ *that counts modulo* $\mathsf{exp}(k, n)$*.*

Proof. (sketch) The proof is by induction on k. For $k = 1$, we describe an $\mathcal{EL}$-TBox that counts modulo 2^n. To this aim, we introduce concept names $X_1, \overline{X_1}, \ldots, X_n, \overline{X_n}$, $C_1, \overline{C_1}, \ldots, C_n, \overline{C_n}$ modeling n bits and carry bits (and their negations), respectively, of a counter and take the CIs

$$\exists r.X_1 \sqsubseteq \overline{X_1} \sqcap C_1 \qquad \exists r.\overline{X_1} \sqsubseteq X_1 \sqcap \overline{C_1}$$
$$C_i \sqcap \exists r.X_{i+1} \sqsubseteq \overline{X_{i+1}} \sqcap C_{i+1}$$
$$C_i \sqcap \exists r.\overline{X_{i+1}} \sqsubseteq X_{i+1} \sqcap \overline{C_{i+1}}$$
$$\overline{C_i} \sqcap \exists r.X_{i+1} \sqsubseteq X_{i+1} \sqcap \overline{C_{i+1}}$$
$$\overline{C_i} \sqcap \exists r.\overline{X_{i+1}} \sqsubseteq \overline{X_{i+1}} \sqcap \overline{C_{i+1}}$$

For initializing the counter and for propagating Fail we include the following CIs for $1 \leq i \leq n$:

$$\mathsf{Init}_1 \sqsubseteq \mathsf{Zero}_1, \qquad \mathsf{Zero}_1 \equiv \overline{X_1} \sqcap \ldots \sqcap \overline{X_n},$$
$$X_i \sqsubseteq \overline{\mathsf{Zero}_1}, \qquad X_i \sqcap \overline{X_i} \sqsubseteq \mathsf{Fail}, \qquad C_i \sqcap \overline{C_i} \sqsubseteq \mathsf{Fail}$$

It should be clear that the TBox $\mathcal{T}_{1,n}$ consisting of all these CIs counts modulo 2^n.

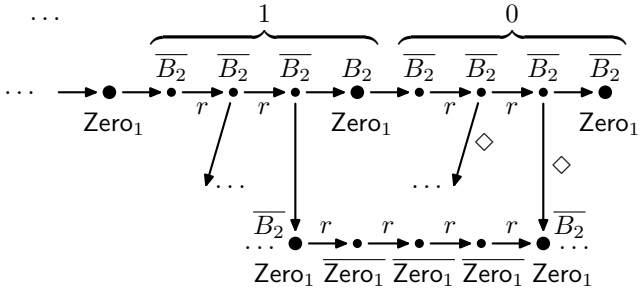
For the induction step $k{>}1$, a number is now represented in binary in $\mathsf{exp}(k-1, n)$ consecutive domain elements along the r-chain; we use additional concepts B_k ($\overline{B_k}$) expressing that a bit is set (not set). We proceed 'inductively':

(1) If Init_k is satisfied in some domain element (in some world), then enforce that the next $\mathsf{exp}(k-1, n)$ domain elements satisfy $\overline{B_k}$; this encodes the number 0.

(2) If a sequence of $\mathsf{exp}(k-1, n)$ domain elements encodes some number, say M, then the preceding sequence of $\mathsf{exp}(k-1, n)$ domain elements is enforced to encode the number $M + 1$.

Clearly, for both steps (1) and (2) it is required to determine $\mathsf{exp}(k-1, n)$ consecutive elements, which is possible due to the induction hypothesis. In particular, for step (2) we mark the begin of the encoding of some number using $\mathcal{T}_{k-1,n}$: a new number begins whenever the concept Zero_{k-1} is satisfied. Also by induction hypothesis, we can mark all domain elements that are *not* the begin of some number.

For the actual incrementation necessary in Point (2), it suffices to communicate between domain elements that have distance (again: along r) precisely $\mathsf{exp}(k-1, n)$ as these domain elements represent the same bit positions in two successive numbers. This is realized by enforcing in each element of the chain a possible future where the bit of this element is stored. In the domain element having precisely distance $\mathsf{exp}(k-1, n)$, this bit is accessed and the correct value is enforced, see the following picture ($M = 0, k = 2$).



It remains to mark the elements of the sequence with Zero_k and $\overline{\mathsf{Zero}_k}$, respectively. First, note that Zero_k can hold only

[1] $\exists r^i$ abbreviates sequences of $\exists r$ with length i.

at the begin of the encoding of some number. So in all 'non-begin' elements we imply $\overline{\mathsf{Zero}_\mathsf{k}}$. For distinguishing the begin elements it suffices to detect the encoding of a number where all $\exp(k-1,n)$ bits are set to 1. Then, starting from the next domain element we will find the encoding of a 0, and we mark its begin with Zero_k. ❑

In the second step we employ the TBoxes $\mathcal{T}_{k,n}$ to reduce the word-problem for deterministic Turing machines with k-fold exponential space restriction to subsumption in $\text{CTL}^{\mathbf{E}\Diamond}_{\mathcal{EL}}$. Theorem 10 is then a direct corollary.

Theorem 12 *Concept subsumption w.r.t. general* $\text{CTL}^{\mathbf{E}\Diamond}_{\mathcal{EL}}$*-TBoxes is* k-EXPSPACE*-hard for every* $k \geq 1$.

Proof. (sketch) Fix any $k \geq 1$ and let M be a deterministic Turing machine which runs within $\exp(k, p(n))$ space for inputs of length n for some polynomial $p(n)$. Let w be some input of length n and set $N := p(n)$. By Lemma 11, there is a TBox $\mathcal{T}_{k,N}$ that counts modulo $\exp(k,N)$ and whose size is polynomial in k, N and hence also polynomial in k, n.

For the reduction, we first include the CI $A \sqsubseteq \exists r.A$ that generates an infinite and rigid r-chain. For every domain element satisfying A, we enforce a possible future that simulates M (again backwards) along the r-chain: $A \sqsubseteq \mathbf{E}\Diamond\mathsf{Init}_M$. A configuration of M is represented by a sequence of $\exp(k,N)$ domain elements connected via r using concept names for every symbol from the tape alphabet, for every state, and for the head. The borders between two configurations are marked using $\mathcal{T}_{k,N}$. We proceed inductively along the lines of the proof of Lemma 11:

(1') enforce the initial configuration of M (including w);

(2') if a sequence of $\exp(k,N)$ domain elements between two borders encodes a configuration of M, the preceding sequence of $\exp(k,N)$ domain elements is enforced to encode the successor configuration.

Obviously, step (1') can be done using a polynomial TBox starting from the concept Init_M. For enforcing tape symbols in step (2') it suffices to communicate between domain elements that have an r-distance of precisely $\exp(k,N)$. Finally, for updating the head position, we might have to communicate between elements having distance $\exp(k,N) \pm 1$. The previous ideas can be easily adapted to this case.

It remains to add the CI $\mathbf{E}\Diamond Q_{acc} \sqsubseteq B$ in order to detect an accepting configuration. It is routine to show that M accepts w iff the constructed TBox implies $A \sqsubseteq B$. ❑

5 Decidability of CTL with *DL-Lite*$^{\mathcal{N}}_{bool}$

In this section, we show that the technique developed by (Artale et al. 2012) to prove decidability of TDLs based on *linear-time* temporal logic and *DL-Lite* is robust in the sense that it can be carried over to *some* TDLs based on CTL and *DL-Lite*$^{\mathcal{N}}_{bool}$ (Artale et al. 2009).

CTL-*DL-Lite*$^{\mathcal{N}}_{bool}$ *concepts* C, D are defined by the following grammar:

$$\bot \mid A \mid \geq q\, r \mid \neg C \mid C \sqcap D \mid \mathbf{E}\bigcirc C \mid \mathbf{E}\Box C \mid \mathbf{E}(C\,\mathcal{U}\,D),$$

where q is a positive integer given in binary, A ranges over concept names and r ranges over $\{r, r^- \mid r \in \mathsf{N_R}\}$.

The semantics for Booleans and temporal operators is defined as in Section 2. We interpret *inverse roles* and *number restrictions* in a temporal interpretation $\mathfrak{I}$ as follows:

$$\begin{aligned}(r^-)^{\mathfrak{I},w} &= \{(d',d) \mid (d,d') \in r^{\mathfrak{I},w}\};\\ (\geq q\, r)^{\mathfrak{I},w} &= \{d \mid \sharp\{d' \mid (d,d') \in r^{\mathfrak{I},w}\} \geq q\}.\end{aligned}$$

Our following result shows that three fragments of CTL-*DL-Lite*$^{\mathcal{N}}_{bool}$ behave in a similar way as temporal extensions of *DL-Lite*$^{\mathcal{N}}_{bool}$ based on linear time: reasoning in these fragments is not harder than in the component logics.

Theorem 13 *Concept satisfiability w.r.t. general TBoxes is* EXPTIME*-complete*
1. for CTL-*DL-Lite*$^{\mathcal{N}}_{bool}$ *without local roles,*
2. for $\text{CTL}^{\mathbf{E}\mathcal{U},\mathbf{E}\Box}$-*DL-Lite*$^{\mathcal{N}}_{bool}$*,*
and PSPACE*-complete*
3. for $\text{CTL}^{\mathbf{E}\Diamond}$-*DL-Lite*$^{\mathcal{N}}_{bool}$*.*

The fragments of points 2,3 do not allow the operator $\mathbf{E}\bigcirc$.

Proof. (sketch) For the upper bound, the proof follows the two-step technique proposed by (Artale et al. 2012). First, we reduce from CTL-*DL-Lite*$^{\mathcal{N}}_{bool}$ to the one-variable fragment $\mathcal{QCTL}^1$ of *first-order branching temporal logic* $\mathcal{QCTL}$ (Hodkinson, Wolter, and Zakharyaschev 2002). Then, the result is further reduced to a $\mathcal{QCTL}^1$-formula *without occurrences of existential quantifiers*, which is essentially a CTL-formula. This way we get the same upper bounds as for the propositional CTL fragments. The lower bounds are inherited from the corresponding fragments (Meier et al. 2009).

The elimination of the quantifiers in the second step does not work for full CTL-*DL-Lite*$^{\mathcal{N}}_{bool}$ because the original shifting technique (Artale et al. 2012, Lemma 4.2) relies on the past being unbounded, which is not the case for the standard CTL semantics. However, the problem can be circumvented by (1) disallowing local roles – which makes shifting superfluous – or (2) restricting the temporal operators to $\mathbf{E}\mathcal{U}$ and $\mathbf{E}\Box$ which are tolerant to a certain variant of unravelling into the temporal direction. Unravelling tolerance is related to stutter invariance (Lamport 1983) and does not apply to $\mathbf{E}\bigcirc$ and therefore to the strict variants $\mathbf{E}\Box^<$ and $\mathbf{E}\mathcal{U}^<$.

We note that the only temporal operators used to encode the TBox and rigidity of roles in the first step are $\mathbf{E}\Diamond$ and $\mathbf{A}\Box$, which are available in $\text{CTL}^{\mathbf{E}\Diamond}$-*DL-Lite*$^{\mathcal{N}}_{bool}$. Hence our translation works for $\text{CTL}^{\mathbf{E}\Diamond}$-*DL-Lite*$^{\mathcal{N}}_{bool}$, too. ❑

6 Conclusions and Future Work

In this paper, we have made progress towards understanding the landscape of the computational complexity and expressivity of BTDLs allowing for rigid roles and general TBoxes. In particular, we showed for some BTDLs based on $\mathcal{EL}$ that we cannot easily conclude undecidability via non-convexity, as in the case of similar LTDLs, since they can be shown to be convex. However, we have been able to squelch the hope that these convex logics are practical because we established a non-elementary lower bound and undecidability, respectively.

The problems left open include determining the precise complexity of $\mathrm{CTL}^{\mathbf{E}\Diamond}_{\mathcal{EL}}$ and of unrestricted CTL-*DL-Lite*$^{\mathcal{N}}_{bool}$. For the latter, we plan to pursue an automata-based approach in the style of (Gutiérrez-Basulto, Jung, and Lutz 2012).

Our investigation provides several research directions that are worth exploring in the near future – for example, one could consider reasoning in the presence of *acyclic TBoxes* instead of general ones, as usually the computational properties of DLs over acyclic TBoxes are better. Furthermore, complexity might also drop when assuming *expanding domains* instead of constant domains.

For BTDLs based on *DL-Lite*, the investigation can be broadened in two ways: by considering restrictions such as *DL-Lite*$_{core}$ or by studying the effect of adding role inclusions. In fact, LTDLs based on *DL-Lite*$_{core}$ are less complex than their *DL-Lite*$_{bool}$ counterpart (Artale et al. 2012). Another possibility is the study of BTDLs with the more expressive *temporal roles*; e.g., where $\mathbf{E}\Diamond$ and $\mathbf{A}\Box$ are applied to roles. LTDLs with temporal roles are undecidable (Artale et al. 2012).

Finally, we believe that our results, together with the suggested restriction to acyclic and empty TBoxes, can have an impact on the study of PMLs, given the known correspondence between PMLs and combinations of DLs and modal-like logics with rigid roles and empty TBoxes: it seems that PMLs with good computational properties may be found among their *positive fragments*, some of which are notational variants of $\mathrm{CTL}^{\mathbf{E}\bigcirc}_{\mathcal{EL}}$, $\mathrm{CTL}^{\mathbf{E}\Diamond}_{\mathcal{EL}}$and $\mathrm{CTL}^{\mathbf{E}\Diamond,\mathbf{A}\Box}_{\mathcal{EL}}$. For instance, the satisfiability problems of $\mathsf{K} \times \mathsf{K}$ and $\mathsf{S4} \times \mathsf{K}$ are non-elementary, and there is hope that the complexity of their positive fragments is only elementary.

Acknowledgements V. Gutiérrez-Basulto was supported by the M8-Project TS-OBDA and UK EPSRC grant EP/H043594/1, and J.C. Jung by the DFG project LU1417/1-1. We thank Frank Wolter and Agi Kurucz for helpful comments and discussions. We also thank the anonymous reviewers for their constructive suggestions.

References

Artale, A., and Franconi, E. 2000. A survey of temporal extensions of description logics. *Ann. Math. Artif. Intell.* 30(1-4):171–210.

Artale, A.; Kontchakov, R.; Lutz, C.; Wolter, F.; and Zakharyaschev, M. 2007. Temporalising tractable description logics. In *Proc. TIME*, 11–22. IEEE.

Artale, A.; Calvanese, D.; Kontchakov, R.; and Zakharyaschev, M. 2009. The DL-Lite family and relations. *J. Artif. Intell. Res. (JAIR)* 36:1–69.

Artale, A.; Kontchakov, R.; Ryzhikov, V.; and Zakharyaschev, M. 2012. A cookbook for temporal conceptual data modelling with description logics. *CoRR*. http://arxiv.org/abs/1209.5571.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proc. IJCAI*, 364–369.

Baader, F.; Ghilardi, S.; and Lutz, C. 2008. LTL over description logic axioms. In *Proc. KR*, 684–694. AAAI Press.

Bauer, S.; Hodkinson, I. M.; Wolter, F.; and Zakharyaschev, M. 2004. On non-local propositional and weak monodic quantified CTL. *J. Log. Comput.* 14(1):3–22.

Chang, C., and Keisler, H. 1990. *Model Theory*. Elsevier.

Clarke, E.; Grumberg, O.; and Peled, D. 1999. *Model checking*. MIT Press.

Fagin, R. 1982. Horn clauses and database dependencies. *J. ACM* 29(4):952–985.

Franconi, E., and Toman, D. 2011. Fixpoints in temporal description logics. In *Proc. of IJCAI*, 875–880. AAAI.

Gabbay, D.; Kurucz, A.; Wolter, F.; and Zakharyaschev, M. 2003. *Many-dimensional modal logics: theory and applications*, volume 148 of *Studies in Logic*. Elsevier.

Gabelaia, D.; Kurucz, A.; Wolter, F.; and Zakharyaschev, M. 2005. Products of 'transitive' modal logics. *J. Symb. Log.* 70(3):993–1021.

Göller, S.; Jung, J. C.; and Lohrey, M. 2012. The complexity of decomposing modal and first-order theories. In *Proc. LICS*, 325–334. ACM/IEEE.

Gutiérrez-Basulto, V.; Jung, J. C.; Lutz, C.; and Schröder, L. 2011. A closer look at the probabilistic description logic Prob-$\mathcal{EL}$. In *Proc. AAAI*.

Gutiérrez-Basulto, V.; Jung, J. C.; and Lutz, C. 2012. Complexity of branching temporal description logics. In *Proc. ECAI*, 390–395.

Hodkinson, I. M.; Wolter, F.; and Zakharyaschev, M. 2002. Decidable and undecidable fragments of first-order branching temporal logics. In *Proc. LICS*, 393–402. IEEE.

Lamport, L. 1983. What good is temporal logic? In *IFIP Congress*, 657–668.

Lutz, C., and Schröder, L. 2010. Probabilistic description logics for subjective uncertainty. In *Proc. KR*. AAAI Press.

Lutz, C., and Wolter, F. 2012. Non-uniform data complexity of query answering in description logics. In *Proc. KR*. AAAI Press.

Lutz, C.; Wolter, F.; and Zakharyaschev, M. 2008. Temporal description logics: A survey. In *Proc. TIME*, 3–14. IEEE.

Meier, A.; Thomas, M.; Vollmer, H.; and Mundhenk, M. 2009. The complexity of satisfiability for fragments of CTL and CTL*. *Int. J. Found. Comput. Sci.* 20(5):901–918.

Minsky, M. L. 1967. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., 1st edition.

Oppen, D. 1980. Complexity, convexity and combinations of theories. *Theor. Comput. Sci.* 12:291–302.

Schild, K. 1993. Combining terminological logics with tense logic. In *Proc. of EPIA*, 105–120. Springer.

Stockmeyer, L. J. 1974. *The Complexity of Decision Problems in Automata Theory and Logic*. Ph.D. Dissertation, MIT, Cambridge, Massasuchets, USA.

Tobies, S. 2001. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. Ph.D. Dissertation, RWTH Aachen.

Finite Model Reasoning in Horn Description Logics

Yazmín Ibañez-García
KRDB Research Centre for Knowledge and Data,
Free University of Bozen-Bolzano, Italy
{ibanezgarcia@inf.unibz.it}

Carsten Lutz and **Thomas Schneider**
Fachbereich Informatik
Universität Bremen, Germany
{clu, tschneider}@informatik.uni-bremen.de

Abstract

We study finite model reasoning in expressive Horn description logics (DLs), starting with a reduction of finite ABox consistency to unrestricted ABox consistency. The reduction relies on reversing certain cycles in the TBox, an approach that originated in database theory, was later adapted to the inexpressive DL *DL-Lite*$_\mathcal{F}$, and is shown here to extend to the expressive Horn DL Horn-$\mathcal{ALCFI}$. The model construction used to establish correctness makes the structure of finite models more explicit than existing approaches to finite model reasoning in expressive DLs that rely on solving systems of inequations over the integers. Since the reduction incurs an exponential blow-up, we then develop a dedicated consequence-based algorithm for finite ABox consistency in Horn-$\mathcal{ALCFI}$ that implements the reduction on-the-fly rather than executing it up-front. The algorithm has optimal worst-case complexity and provides a promising foundation for implementations. We next show that our approach can be adapted to finite (positive existential) query answering relative to Horn-$\mathcal{ALCFI}$ TBoxes, proving that this problem is EXPTIME-complete in combined complexity and PTIME-complete in data complexity. For finite satisfiability and subsumption, we also show that our techniques extend to Horn-$\mathcal{SHIQ}$.

1 Introduction

Many popular expressive description logics (DLs) include both inverse roles and some form of counting such as functionality restrictions. This combination is well-known to result in a loss of the finite model property (FMP) and, consequently, reasoning w.r.t. the class of finite models (*finite model reasoning*) does not coincide with reasoning w.r.t. the class of all models (*unrestricted reasoning*). On the one hand, this distinction is gaining importance because DLs are increasingly used in database applications, where finiteness of models and databases is a central assumption. On the other hand, finite model reasoning is rarely used when DLs are applied in practice, mainly because for many DLs that lack the FMP, no algorithmic approaches to finite model reasoning are known that lend themselves towards efficient implementation.

Among the most widely-known DLs that include both inverse roles and counting are $\mathcal{ALCFI}$, $\mathcal{ALCQI}$, $\mathcal{SHIF}$, and $\mathcal{SHIQ}$, which are prominent fragments of the OWL2 DL ontology language. While finite model reasoning in these DLs is known to have the same complexity as unrestricted reasoning, namely EXPTIME-complete (Lutz, Sattler, and Tendera 2005), the algorithmic approaches are rather different when only finite models are admitted. For unrestricted reasoning, there is a wide range of applicable algorithms such as tableau and resolution calculi, which often perform rather well in practical implementations. For finite model reasoning, all known approaches rely on the construction of some system of inequations (Calvanese 1996; Lutz, Sattler, and Tendera 2005) and then solve this system over the integers; the crux is that the system of inequations is of exponential size in the best case, and consequently it is far from obvious how to come up with efficient implementations. This is also true for the two-variable fragment of first-order logic with counting quantifiers (C2), into which the mentioned DLs can be embedded (Pacholski, Szwast, and Tendera 2000; Pratt-Hartmann 2005), that is, all known approaches to finite model reasoning in C2 rely on solving (at least) exponentially large systems of inequations.

Interestingly, the situation is quite different on the other end of the expressive power spectrum. While $\mathcal{SHIQ}$ et al. belong to the family of expressive DLs, *DL-Lite*$_\mathcal{F}$ is a comparably inexpressive DL that emerged from database applications, but also includes both inverse roles and functionality restrictions and thus lacks the FMP. Building on a technique that was developed in database theory by Cosmadakis, Kanellakis, and Vardi to decide the implication of inclusion dependencies and functional dependencies over finite databases (1990), Rosati has shown that finite model reasoning in *DL-Lite*$_\mathcal{F}$ can be reduced in polynomial time to unrestricted reasoning in *DL-Lite*$_\mathcal{F}$ (2008). The reduction is conceptually simple and relies on completing the TBox by finding certain cyclic inclusions and then 'reversing' them. For example, the cycle

$$\exists r^- \sqsubseteq \exists s \qquad \exists s^- \sqsubseteq \exists r \qquad (\mathsf{funct}\ r^-) \qquad (\mathsf{funct}\ s^-)$$

that consists of existential restrictions in the 'forward direction' and functionality statements in the 'backwards direction' would lead to the addition of the reversed cycle

$$\exists s \sqsubseteq \exists r^- \qquad \exists r \sqsubseteq \exists s^- \qquad (\mathsf{funct}\ r) \qquad (\mathsf{funct}\ s).$$

As a consequence, finite model reasoning in DL-Lite$_\mathcal{F}$ does not require new algorithmic techniques and can be implemented as efficiently as unrestricted reasoning. The reduction

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

also makes explicit the *logical consequences of finite models*; in a sense, it can be viewed as an explicit axiomatization of finiteness.

Given that DL-Lite$_\mathcal{F}$ is only a very small fragment of $\mathcal{ALCFI}$ and $\mathcal{SHIQ}$, this situation raises the question whether the cycle reversion technique extends also to larger fragments of these DLs. In particular, *DL-Lite*$_\mathcal{F}$ is a 'Horn DL', and such logics are well-known to be algorithmically much more well-behaved than non-Horn DLs such as $\mathcal{ALCFI}$ (Baader, Brandt, and Lutz 2005; Calvanese et al. 2007). Maybe, then, this is the reason why cycle reversion works for *DL-Lite*$_\mathcal{F}$?

In this paper, we show that the cycle reversion technique of Cosmadakis et al. extends all the way to the expressive DLs Horn-$\mathcal{ALCFI}$ and Horn-$\mathcal{ALCQI}$. These logics, as well as their extensions Horn-$\mathcal{SHIF}$ and Horn-$\mathcal{SHIQ}$, are popular in ontology-based data access (Hustadt, Motik, and Sattler 2007; Ortiz, Rudolph, and Šimkus 2011; Eiter et al. 2012; Bienvenu, Lutz, and Wolter 2013) and properly extend *DL-Lite*$_\mathcal{F}$ and other relevant Horn DLs such as $\mathcal{ELIF}$ (Krisnadhi and Lutz 2007). We start with showing that finite ABox consistency in Horn-$\mathcal{ALCFI}$ can be reduced to unrestricted ABox consistency in Horn-$\mathcal{ALCFI}$ by cycle reversion; it follows that the same is true for finite satisfiability, finite subsumption, and finite instance checking. While the reduction technique is conceptually similar to that for *DL-Lite*$_\mathcal{F}$, the construction of a finite model in the correctness proof is more demanding. In comparison to approaches to finite model reasoning that rely on solving systems of inequations, though, they make the structure of finite models considerably more explicit.

Another crucial difference to the *DL-Lite*$_\mathcal{F}$ case is that, when completing Horn-$\mathcal{ALCFI}$ TBoxes, the cycles that have to be considered can be of exponential length, and thus the reduction is not polynomial. Consequently, when used in a naive way it can neither be expected to perform well in practice nor be used to (re)prove tight complexity bounds. To address these shortcomings, we develop a dedicated calculus for finite ABox consistency in Horn-$\mathcal{ALCFI}$ that implements the reduction on-the-fly rather than executing it up-front. The calculus is an extension of a consequence-based procedure for unrestricted satisfiability in Horn-$\mathcal{SHIQ}$ that was introduced by Kazakov in (2009) and implemented in the highly performant reasoner CB, first to classify the notoriously difficult Galen ontology. Many other state-of-the art reasoners for Horn-DLs are also based on consequence-based procedures, including ELK (Kazakov, Krötzsch, and Simančík 2011a) and CEL (Baader, Lutz, and Suntisrivaraporn 2006). Our algorithm shares the main feature of other consequence-based procedures to carefully avoid considering 'types' (conjunctions of concept names) that are irrelevant for deciding the problem at hand. We therefore believe that it provides a very promising basis for efficient implementations of finite model reasoning in Horn-$\mathcal{ALCFI}$. It also (re)proves the optimal upper ExpTime complexity bound for finite ABox consistency in this DL. Via a reduction, the cycle reversing reduction and the consequence-based algorithm can be applied also to finite satisfiability and subsumption in Horn-$\mathcal{ALCQI}$.

We then consider the paradigm of ontology-based data access (OBDA), extending our results from finite ABox consistency to answering positive existential queries (PEQs), relative to Horn-$\mathcal{ALCFI}$ TBoxes over finite models. In particular, we show that the reduction based on cycle reversion developed for ABox consistency also works in the case of PEQ answering. The construction of (counter)models in the correctness proofs, however, becomes yet more difficult and technical, and proceeds in two stages. First, we carefully modify the models constructed for finite ABox consistency so that there are no unintended matches of acyclic conjunctive queries (CQs). And second, we take a product with a finite group of high girth to eliminate unintended matches of cyclic CQs. Based on this result, we then prove that finite PEQ entailment (the Boolean version of PEQ answering) in Horn-$\mathcal{ALCFI}$ is ExpTime-complete regarding combined complexity and PTime-complete regarding data complexity. Previously, it was only known that finite CQ answering in (non-Horn) $\mathcal{ALCQI}$ is decidable and in coNP regarding data complexity (Pratt-Hartmann 2009).

Some proof details are deferred to the appendix in the long version: http://tinyurl.com/kr14fmr

2 Preliminaries

We introduce the DLs Horn-$\mathcal{ALCFI}$ and Horn-$\mathcal{ALCQI}$, as well as the reasoning tasks studied in this paper. The original definition of these DLs is based on a notion of polarity and somewhat unwieldy (Hustadt, Motik, and Sattler 2007); alternative and more direct definitions have been proposed later, see for example (Lutz and Wolter 2012). For brevity, we directly introduce Horn-$\mathcal{ALCQI}$ TBoxes in a normal form that is convenient for our purposes and disallows syntactic nesting of operators. It is a minor variation of the normal form proposed in (Kazakov 2009).

Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be countably infinite and disjoint sets of concept names, role names, and individual names. A *role* is either a role name r or an *inverse role* r^-. A *Horn-$\mathcal{ALCQI}$ TBox* $\mathcal{T}$ is a set of *concept inclusions (CIs)* that can take the following forms:

$$K \sqsubseteq A \qquad K \sqsubseteq \bot \qquad K \sqsubseteq \exists r.K'$$
$$K \sqsubseteq \forall r.K' \qquad K \sqsubseteq (\leqslant 1\ r\ K') \qquad K \sqsubseteq (\geqslant n\ r\ K')$$

where K and K' denote a (possibly empty) conjunction of concept names, A a concept name, r a (potentially inverse) role, and $n \geq 2$. Throughout the paper, we will deliberately confuse conjunctions of concept names and sets of concept names. The empty conjunction is abbreviated by $\top$. As usual, we allow to easily switch between role names and their inverse by identifying $(r^-)^-$ and r. A *Horn-$\mathcal{ALCFI}$ TBox* is a Horn-$\mathcal{ALCQI}$ TBox that does not include CIs of the form $K \sqsubseteq (\geqslant n\ r\ K')$.

The semantics of Horn-$\mathcal{ALCQI}$ is based on interpretations as usual, see (Baader et al. 2003) for details. We write $\mathcal{T} \models C \sqsubseteq D$ if the concept inclusion $C \sqsubseteq D$ is satisfied in all models of the TBox $\mathcal{T}$, and $\mathcal{T} \models_{\mathsf{fin}} C \sqsubseteq D$ if the same holds for all finite models. A concept name A is *(finitely) satisfiable* w.r.t. a TBox $\mathcal{T}$ if $\mathcal{T}$ has a (finite) model $\mathcal{I}$ with $A^\mathcal{I} \neq \emptyset$. If $\mathcal{T} \models A \sqsubseteq B$ (resp. $\mathcal{T} \models_{\mathsf{fin}} A \sqsubseteq B$) with A and B concept names, then we say that B is *(finitely) subsumed* by A.

An *ABox* is a finite set of *concept assertions* $A(a)$ and *role assertions* $r(a, b)$ where A is a concept name, r a role name, and a, b are individual names. For simplicity, we make the *standard names assumption*, that is, every interpretation $\mathcal{I}$ interpretes all individuals as themselves; for example $\mathcal{I}$ satisfies $A(a)$ if $a \in A^{\mathcal{I}}$. The standard names assumption implies the unique name assumption (UNA). The results in this paper, however, do not depend on any of these assumptions. Throughout the paper, we sometimes write $r^-(a, b) \in \mathcal{A}$ for $r(b, a) \in \mathcal{A}$ and use $\mathsf{Ind}(\mathcal{A})$ to denote the set of all individual names that occur in $\mathcal{A}$.

We write $\mathcal{A}, \mathcal{T} \models A(a)$ if the ABox assertion $A(a)$ is satisfied in all common models of the ABox $\mathcal{A}$ and the TBox $\mathcal{T}$, and $\mathcal{A}, \mathcal{T} \models_{\mathsf{fin}} A(a)$ if the same holds for all finite models. We then say that a is a *(finite) instance* of A in $\mathcal{A}$ w.r.t. $\mathcal{T}$. An ABox $\mathcal{A}$ is *(finitely) consistent* w.r.t. $\mathcal{T}$ if there is a (finite) model $\mathcal{I}$ of $\mathcal{T}$ that satisfies all assertions in $\mathcal{A}$.

The above notions give rise to four decision problems studied in this paper, which are *finite satisfiability* (of a concept name w.r.t. a TBox), *finite subsumption* (between two concept names w.r.t. a TBox), *finite ABox consistency* (w.r.t. a TBox) and *finite instance checking* (of an ABox individual and a concept name, w.r.t. an ABox and a TBox). There are easy polynomial time reductions from satisfiability to subsumption to instance checking to ABox consistency, which work both in the finite and in the unrestricted case.

The following examples show that, in Horn-$\mathcal{ALCFI}$, finite and unrestricted reasoning do not coincide.

Example 1

$$\mathcal{T} = \{ \quad A \sqsubseteq \exists r.B, \qquad B \sqsubseteq \exists r.B, \\ B \sqsubseteq (\leqslant 1\ r^-\ \top), \quad A \sqcap B \sqsubseteq \bot \qquad \}$$

A is satisfiable w.r.t. $\mathcal{T}$, but not finitely satisfiable. In fact, when $d \in A^{\mathcal{I}}$ in some model $\mathcal{I}$ of $\mathcal{T}$, then there must be an infinite chain $r(d, d_1), r(d_1, d_2), \ldots$ with $d \in A^{\mathcal{I}}$, and $d_2, d_3, \cdots \in B^{\mathcal{I}}$. Since d cannot be in $B^{\mathcal{I}}$ and r is inverse functional, no two elements on the chain can be identified.

$$\mathcal{T}' = \{\ A_1 \sqsubseteq \exists r.A_2, \quad A_2 \sqsubseteq \exists r.(A_1 \sqcap B), \\ \top \sqsubseteq (\leqslant 1\ r^-\ \top) \quad \}$$

The reader might want to verify that $\mathcal{T}' \not\models A_1 \sqsubseteq B$, but $\mathcal{T}' \models_{\mathsf{fin}} A_1 \sqsubseteq B$.

It follows form the observations in (Kazakov 2009) that, for the purposes of deciding satisfiability of concepts in unrestricted models, the normal form for TBoxes introduced above can be assumed without loss of generality because every Horn-$\mathcal{ALCQI}$ TBox $\mathcal{T}$ can be converted in polynomial time into a TBox $\mathcal{T}'$ in the above form such that every model of $\mathcal{T}'$ is a model of $\mathcal{T}$ and, conversely, every model of $\mathcal{T}$ can be converted into a model of $\mathcal{T}'$ by interpreting the concept names that were introduced during normalization. It follows that normal form can be assumed w.l.o.g. both for unrestricted reasoning and for finite model reasoning, and for all reasoning problems considered in this paper.

3 From Finite Models to Unrestricted Models

We show that finite ABox consistency in Horn-$\mathcal{ALCFI}$ can be reduced to unrestricted ABox consistency by reversing certain cycles in the TBox. The reduction exhibited in this section provides a novel decision procedure for finite ABox consistency in Horn-$\mathcal{ALCFI}$ and Horn-$\mathcal{ALCQI}$ (as well as for finite satisfiability, finite subsumption, and finite instance checking) and is the basis for developing a consequence-based procedure in Section 4. It also highlights the logical consequences of finite models in Horn-$\mathcal{ALCFI}$. The material in this section is an extended and improved version of the workshop paper (Ibáñez-García, Lutz, and Schneider 2013).

Reversing Cycles

Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox. A *finmod cycle in $\mathcal{T}$* is a sequence $K_1, r_1, K_2, r_2, \ldots, r_{n-1}, K_n$, with $K_1, \ldots, K_n$ conjunctions of concept names and $r_1, \ldots, r_{n-1}$ (potentially inverse) roles such that $K_n = K_1$ and, for $1 \leq i < n$:

$$\mathcal{T} \models K_i \sqsubseteq \exists r_i.K_{i+1} \text{ and } \mathcal{T} \models K_{i+1} \sqsubseteq (\leqslant 1\ r_i^-\ K_i).$$

By *reversing* a finmod cycle $K_1, r_1, K_2, r_2, \ldots, r_{n-1}, K_n$ in a TBox $\mathcal{T}$, we mean to extend $\mathcal{T}$ with the following concept inclusions, for $1 \leq i < n$:

$$K_{i+1} \sqsubseteq \exists r_i^-.K_i \text{ and } K_i \sqsubseteq (\leqslant 1\ r_i\ K_{i+1}).$$

The *completion* $\mathcal{T}_{\mathsf{f}}$ of a TBox $\mathcal{T}$ is obtained from $\mathcal{T}$ by exhaustively reversing finmod cycles. Note that, although there may be infinitely many finmod cycles, only finitely many CIs can be added by cycle reversion (exponentially many in the size of the original TBox, in the worst case). For finding these finitely many CIs, it clearly suffices to consider finmod cycles in which all triples (r_i, K_{i+1}, r_{i+1}) are distinct. Also note that finding finmod cycles requires deciding unrestricted subsumption, which is decidable and EXPTIME-complete.

Example 2 *The TBox $\mathcal{T}'$ from Example 1 entails (in unrestricted models)*

$$A_1 \sqcap B \sqsubseteq \exists r.A_2, \qquad A_2 \sqsubseteq \exists r.(A_1 \sqcap B), \\ A_2 \sqsubseteq (\leqslant 1\ r^-\ A_1 \sqcap B), \quad A_2 \sqcap B \sqsubseteq (\leqslant 1\ r^-\ A_1).$$

Thus, A_1, r, A_2, r, A_1, is a finmod cycle in $\mathcal{T}'$, which is reversed to

$$A_2 \sqsubseteq \exists r^-.A_1, \qquad A_1 \sqsubseteq \exists r^-.A_2, \\ A_2 \sqsubseteq (\leqslant 1\ r\ A_1), \qquad A_1 \sqsubseteq (\leqslant 1\ r\ A_2).$$

From $A_1 \sqsubseteq \exists r^-.A_2$, $A_2 \sqsubseteq \exists r.(A_1 \sqcap B)$, and $A_2 \sqsubseteq (\leqslant 1\ r\ A_1)$, we obtain $\mathcal{T}'_{\mathsf{f}} \models A_1 \sqsubseteq B$, in correspondence with $\mathcal{T}' \models_{\mathsf{fin}} A_1 \sqsubseteq B$. Note that $\mathcal{T}'$ also contains another finmod cycle, which is $(A_1 \sqcap B), r, A_2, r, (A_1 \sqcap B)$.

The following result shows that TBox completion provides a reduction from finite ABox consistency to unrestricted ABox consistency.

Theorem 3 *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox and $\mathcal{A}$ an ABox. Then $\mathcal{A}$ is finitely consistent w.r.t. $\mathcal{T}$ iff $\mathcal{A}$ is consistent w.r.t. the completion $\mathcal{T}_{\mathsf{f}}$ of $\mathcal{T}$.*

The "only if" direction of Theorem 3 is an immediate consequence of the observation that all CIs added by cycle reversion are entailed by the original TBox in finite models.

Lemma 4 *Let $K_1, r_1, \ldots, r_{n-1}, K_n$ be a finmod cycle in $\mathcal{T}$. Then $\mathcal{T} \models_{\mathsf{fin}} K_{i+1} \sqsubseteq \exists r_i^-.K_i$ and $\mathcal{T} \models_{\mathsf{fin}} K_i \sqsubseteq (\leqslant 1\ r_i\ K_{i+1})$ for $1 \leq i < n$,.*

Proof. We have to show that if $K_1, r_1, \dots, r_{n-1}, K_n$ is a finmod cycle in $\mathcal{T}$ and $\mathcal{I}$ is a finite model of $\mathcal{T}$, then $K_i^{\mathcal{I}} \subseteq (\leqslant 1\ r_i\ K_{i+1})^{\mathcal{I}}$ and $K_{i+1}^{\mathcal{I}} \subseteq (\exists r_i^-.K_i)^{\mathcal{I}}$ for $1 \leq i < n$. We first note that, by the semantics of Horn-$\mathcal{ALCFI}$, we must have $|K_1^{\mathcal{I}}| \leq \cdots \leq |K_n^{\mathcal{I}}|$, thus $K_n = K_1$ yields $|K_1^{\mathcal{I}}| = \cdots = |K_n^{\mathcal{I}}|$. Fix some i with $1 \leq i < n$. Using $|K_i^{\mathcal{I}}| = |K_{i+1}^{\mathcal{I}}|$, $K_i^{\mathcal{I}} \subseteq (\exists r_i.K_{i+1})^{\mathcal{I}}$, and $K_{i+1}^{\mathcal{I}} \subseteq (\leqslant 1\ r_i^-\ K_i)^{\mathcal{I}}$, it is easy to verify that $K_i^{\mathcal{I}} \subseteq (\leqslant 1\ r_i\ K_{i+1})^{\mathcal{I}}$ and $K_{i+1}^{\mathcal{I}} \subseteq (\exists r_i^-.K_i)^{\mathcal{I}}$, as required. ❑

We now prove the "if" direction of Theorem 3, which is much more demanding as it requires to explicitly construct finite models.

Constructing Finite Models

Assume that $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}_\mathsf{f}$. Our aim is to construct a finite model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}_\mathsf{f}$ (and thus also of $\mathcal{T}$). Before we give details of the construction, we introduce some relevant preliminaries.

Let $\mathsf{CN}(\mathcal{T})$ denote the set of concept names used in $\mathcal{T}$ (or, equivalently, in $\mathcal{T}_\mathsf{f}$). A *type for* $\mathcal{T}_\mathsf{f}$ is a subset $t \subseteq \mathsf{CN}(\mathcal{T})$ such that there is a (potentially infinite) model $\mathcal{I}$ of $\mathcal{T}_\mathsf{f}$ and a $d \in \Delta^{\mathcal{I}}$ such that $\mathsf{tp}_{\mathcal{I}}(d) = t$, where

$$\mathsf{tp}_{\mathcal{I}}(d) := \{A \in \mathsf{CN}(\mathcal{T}) \mid d \in A^{\mathcal{I}}\}$$

is the type *realized* at d in $\mathcal{I}$. We use $\mathsf{TP}(\mathcal{T}_\mathsf{f})$ to denote the set of all types for $\mathcal{T}_\mathsf{f}$. For $t, t' \in \mathsf{TP}(\mathcal{T}_\mathsf{f})$ and r a role, we write

- $t \to_r t'$ if $\mathcal{T}_\mathsf{f} \models t \sqsubseteq \exists r.t'$ and t' is maximal with this property;
- $t \to_r^1 t'$ if $t \to_r t'$ and $\mathcal{T}_\mathsf{f} \models t' \sqsubseteq (\leqslant 1\ r^-\ t)$;
- $t\ {}^1\!\leftrightarrow_r^1 t'$ if $t \to_r^1 t'$ and $t' \to_{r^-}^1 t$.

Note that when

$$t_1 \to_{r_1}^1 t_2 \to_{r_2}^1 \cdots \to_{r_{n-1}}^1 t_n = t_1 \qquad (*)$$

then $t_1, r_1, \dots, r_{n-1}, t_n$ is a finmod cycle in $\mathcal{T}_\mathsf{f}$ and the fact that it has been reversed means that all '$\to^1$' in $(*)$ can be replaced with ${}^1\!\leftrightarrow^1$. Types related by ${}^1\!\leftrightarrow_r^1$ are connected very tightly by the TBox $\mathcal{T}_\mathsf{f}$ and are best considered together when building finite models. This is formalized by the notion of a *type class*, which is a non-empty set $P \subseteq \mathsf{TP}(\mathcal{T}_\mathsf{f})$ such that $t \in P$ and $t\ {}^1\!\leftrightarrow_r^1 t'$ implies $t' \in P$, and P is minimal with this condition. Note that the set of all type classes is a partition of $\mathsf{TP}(\mathcal{T}_\mathsf{f})$. We set $P \prec P'$ if there are $t \in P$ and $t' \in P'$ with $t' \subsetneq t$. Let $\prec^+$ be the transitive closure of $\prec$. A proof of the following observation can be found in the appendix.

Lemma 5 *$\prec^+$ is a strict partial order.*

We construct the desired finite model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}_\mathsf{f}$ by starting with an initial interpretation that essentially consists of the ABox $\mathcal{A}$ and then exhaustively applying three *completion rules* denoted with $(\mathbf{c1})$ to $(\mathbf{c3})$, where $(\mathbf{c1})$ is given preference over $(\mathbf{c2})$. Completion repeatedly introduces elements whose existence is required by CIs $K \sqsubseteq \exists r.C$, carefully distinguishing several cases to ensure that no functionality restrictions are violated. We will prove that rule application terminates after finitely many steps, producing a finite model.

During the construction of $\mathcal{I}$, we will make sure that the following invariants are satisfied:

(i1) $\mathsf{tp}_{\mathcal{I}}(d) \in \mathsf{TP}(\mathcal{T}_\mathsf{f})$ for all $d \in \Delta^{\mathcal{I}}$;

(i2) if $(d, d') \in r^{\mathcal{I}} \setminus (\mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}))$, then we have $\mathsf{tp}_{\mathcal{I}}(d) \to_r \mathsf{tp}_{\mathcal{I}}(d')$ or $\mathsf{tp}_{\mathcal{I}}(d') \to_{r^-} \mathsf{tp}_{\mathcal{I}}(d)$;

(i3) if $\mathcal{T}_\mathsf{f} \models K \sqsubseteq (\leqslant 1\ r\ K')$, then $\mathcal{I} \models K \sqsubseteq (\leqslant 1\ r\ K')$.

The initial version of $\mathcal{I}$ is defined by introducing an element for every ABox individual, and an element d_t for each $t \in \mathsf{TP}(\mathcal{T}_\mathsf{f})$. In detail, we set

$$\Delta^{\mathcal{I}} = \mathsf{Ind}(\mathcal{A}) \cup \{\ d_t \mid t \in \mathsf{TP}(\mathcal{T})_\mathsf{f}\ \}$$
$$A^{\mathcal{I}} = \{\ a \in \mathsf{Ind}(\mathcal{A}) \mid A \in \mathsf{tp}_{\mathcal{A}}(a)\ \} \cup \{d_t \mid A \in t\ \}$$
$$r^{\mathcal{I}} = \{\ (a, b) \mid r(a, b) \in \mathcal{A}\ \}$$

where

$$\mathsf{tp}_{\mathcal{A}}(a) := \{A \in \mathsf{CN}(\mathcal{T}) \mid \mathcal{A}, \mathcal{T}_\mathsf{f} \models A(a)\}.$$

The completion rules are described in detail below.

$(\mathbf{c1})$ Choose a $d \in \Delta^{\mathcal{I}}$ such that $\mathsf{tp}_{\mathcal{I}}(d) \to_r^1 t$, $t \not\to_{r^-}^1 \mathsf{tp}_{\mathcal{I}}(d)$, and $d \notin (\exists r.t)^{\mathcal{I}}$. Add a fresh domain element e, and modify the extension of concept and role names such that $\mathsf{tp}_{\mathcal{I}}(e) = t$ and $(d, e) \in r^{\mathcal{I}}$.

$(\mathbf{c2})$ Choose a type class P that is minimal w.r.t. the order $\prec^+$, a $\lambda = s\ {}^1\!\leftrightarrow_r^1 s'$ with $s \in P$, and an element $d \in s^{\mathcal{I}} \setminus (\exists r.s')^{\mathcal{I}}$.

For each $\lambda = s\ {}^1\!\leftrightarrow_r^1 s'$ with $s \in P$, set

$$X_{\lambda,1}^{\mathcal{I}} = s^{\mathcal{I}} \setminus (\exists r.s')^{\mathcal{I}} \qquad X_{\lambda,2}^{\mathcal{I}} = s'^{\mathcal{I}} \setminus (\exists r^-.s)^{\mathcal{I}}.$$

Take (i) a fresh set Δ_s for each $s \in P$ such that $|\biguplus_{s \in P} \Delta_s| \leq 2^{|\mathcal{T}|} \cdot |\Delta^{\mathcal{I}}|$ and (ii) a bijection π_λ between $X_{\lambda,1}^{\mathcal{I}} \cup \Delta_s$ and $X_{\lambda,2}^{\mathcal{I}} \cup \Delta_{s'}$ for each $\lambda = s\ {}^1\!\leftrightarrow_r^1 s'$ with $s, s' \in P$ and r a role name (the concrete construction is detailed below). Now extend $\mathcal{I}$ as follows:

- add all domain elements in $\biguplus_{s \in P} \Delta_s$;
- extend $r^{\mathcal{I}}$ with π_λ, for each $\lambda = s\ {}^1\!\leftrightarrow_r^1 s'$ with $s, s' \in P$ and r a role name;
- interpret concept names so that $\mathsf{tp}_{\mathcal{I}}(d) = s$ for all $d \in \Delta_s$, $s \in P$.

$(\mathbf{c3})$ Choose a $d \in \Delta^{\mathcal{I}}$ such that $\mathsf{tp}_{\mathcal{I}}(d) \to_r t$, $\mathsf{tp}_{\mathcal{I}}(d) \not\to_r^1 t$, and $d \notin (\exists r.t)^{\mathcal{I}}$. Add the edge (d, d_t) to $r^{\mathcal{I}}$, where d_t is the element introduced for type t in the initial version of $\mathcal{I}$.

To complete the description of the rules, we have to show that, in $(\mathbf{c2})$, the sets Δ_s and bijections π_λ indeed exist. Let $n_{\max} = \max\{|s^{\mathcal{I}}| \mid s \in P\}$. For each $s \in P$, set $\Delta_s := \{d_{s,i} \mid |s^{\mathcal{I}}| < i \leq n_{\max}\}$ and define the set of *s-instances* $I_s := s^{\mathcal{I}} \cup \Delta_s$. For each $\lambda = s\ {}^1\!\leftrightarrow_r^1 s'$ with $s, s' \in P$, define

$$R_\lambda := \{(d, e) \in r^{\mathcal{I}} \mid d \in s^{\mathcal{I}} \text{ and } e \in s'^{\mathcal{I}}\}.$$

We first note that it is a consequence of invariant **(i3)** that

$(*)$ the relation R_λ is functional and inverse functional.

In fact, $(d, e_1), (d, e_2) \in R_\lambda$ implies $(d, e_1), (d, e_2) \in r^{\mathcal{I}}$, $d \in s^{\mathcal{I}}$, and $e_1, e_2 \in s'^{\mathcal{I}}$. By λ, $\mathcal{T}_\mathsf{f} \models s \sqsubseteq (\leqslant 1\ r\ s')$. Thus, (**i3**) yields $e_1 = e_2$. Inverse functionality can be shown analogously.

Let R^1_λ be the domain of R_λ, and let R^2_λ be its range. By $(*)$, we have $|R^1_\lambda| = |R^2_\lambda|$. By definition of the sets Δ_s, we have $|I_s| = |I_{s'}|$. Moreover, $R^1_\lambda \subseteq I_s$ and $R^2_\lambda \subseteq I_{s'}$. We can thus choose a bijection π_λ between $I_s \setminus R^1_\lambda$ and $I_{s'} \setminus R^2_\lambda$, which is as required since $I_s \setminus R^1_\lambda = X^{\mathcal{I}}_{\lambda,1} \cup \Delta_s$ and $I_{s'} \setminus R^2_\lambda = X^{\mathcal{I}}_{\lambda,2} \cup \Delta_{s'}$. The construction of the sets Δ_s clearly ensures that their union has the required cardinality.

The following theorem summarizes the statements that remain to be proved in order to show that the construction of $\mathcal{I}$ is well-defined and yields a finite model of $\mathcal{A}$ and $\mathcal{T}_\mathsf{f}$.

Theorem 6

1. *Applying* $(\mathbf{c1})$ *to* $(\mathbf{c3})$ *preserves invariants* $(\mathbf{i1})$ *to* $(\mathbf{i3})$*;*
2. *Application of* $(\mathbf{c1})$ *to* $(\mathbf{c3})$ *terminates;*
3. *$\mathcal{I}$ is a model of $\mathcal{A}$ and $\mathcal{T}_\mathsf{f}$.*

Proof. We refer to the appendix for full proofs and only sketch the central idea in the proof of Point 2 here, going back to (Cosmadakis, Kanellakis, and Vardi 1990). The main issue in the termination proof is to show that no infinite role chain $r_0(d_0, d_1), r_1(d_1, d_2), \dots$ is generated in which all the elements d_i are pairwise distinct. Since every application of a completion rule generates only finitely many elements, any such chain must be generated by infinitely many rule applications. As there are only finitely many types, we must find elements d_i and d_j with $\mathtt{tp}_\mathcal{I}(d_i) = \mathtt{tp}_\mathcal{I}(d_j)$ and such that d_i and d_j were generated by different rule applications. It can be shown that, w.l.o.g., we can assume that the elements on the chain are ordered so that if $j > i$, then d_j was not generated by an earlier rule application than d_i. Analysing the completion rules, it is easy to see that this implies $\mathtt{tp}_\mathcal{I}(d_i) \to^1_{r_i} \mathtt{tp}_\mathcal{I}(d_{i+1}) \to^1_{r_{i+1}} \cdots \to^1_{r_{j-1}} \mathtt{tp}_\mathcal{I}(d_j)$. Since $\mathtt{tp}_\mathcal{I}(d_i) = \mathtt{tp}_\mathcal{I}(d_j)$, this is a finmod cycle, which has been reversed when constructing $\mathcal{T}_\mathsf{f}$, and thus all arrows $\to^1_{r_{i+\ell}}$ can be replaced with ${}^1{\leftrightarrow}^1_{r_{i+\ell}}$. By definition of the completion rules, this means that all of $d_i, \dots, d_j$ were introduced in the same application of (**c2**), which is a contradiction to d_i and d_j being generated by different rule applications. ❑

4 Consequence-Driven Procedure

While completing TBoxes with reversed cycles yields a reduction of finite model reasoning to infinite model reasoning, it blows up the TBox exponentially and is thus not suited for direct implementation. In this section, we build on the results from the previous section to devise a calculus for ABox consistency in Horn-$\mathcal{ALCFI}$ that does not require TBox completion to be carried out up-front, but instead reverses cycles 'on the fly'; moreover, the calculus implicitly groups together cycles that are closely related, potentially reversing a very large number of cycles in only a few steps (see Example 7 below). Our calculus belongs to a family of algorithms that are known as consequence-driven procedures and underly modern and highly efficient reasoners for Horn DLs such as

$$\textbf{R1}\ \frac{}{K \sqcap A \sqsubseteq A} \qquad \textbf{R2}\ \frac{}{K \sqsubseteq \top}$$

$$\textbf{R3}\ \frac{K \sqsubseteq A_i \quad \bigsqcap A_i \sqsubseteq C}{K \sqsubseteq C} \qquad \textbf{R4}\ \frac{K \sqsubseteq \exists r.K' \quad K' \sqsubseteq \forall r^-.A}{K \sqsubseteq A}$$

$$\textbf{R5}\ \frac{K \sqsubseteq \exists r.K' \quad K \sqsubseteq \forall r.A}{K \sqsubseteq \exists r.(K' \sqcap A)} \qquad \textbf{R6}\ \frac{K \sqsubseteq \exists r.K' \quad K' \sqsubseteq \bot}{K \sqsubseteq \bot}$$

$$\textbf{R7}\ \frac{\begin{array}{lll} K \sqsubseteq \exists r.K_1 & K \sqsubseteq \exists r.K_2 & K_1 \sqsubseteq A \\ K \sqsubseteq (\leqslant 1\ r\ A) & & K_2 \sqsubseteq A \end{array}}{K \sqsubseteq \exists r.(K_1 \sqcap K_2)}$$

$$\textbf{R8}\ \frac{\begin{array}{lll} K \sqsubseteq \exists r.K' & K' \sqsubseteq \exists r^-.K_1 & K \sqsubseteq A \\ K' \sqsubseteq (\leqslant 1\ r^-\ A) & & K_1 \sqsubseteq A \end{array}}{K \sqsubseteq A_1 \quad \text{for any } A_1 \in K_1}$$

$$\textbf{R9}\ \frac{\begin{array}{ll} K_i \sqsubseteq \exists r_i.K_{i \oplus_n 1} & \\ K_{i \oplus_n 1} \sqsubseteq (\leqslant 1\ r_i^-\ A_i) & K_i \sqsubseteq A_i \qquad i < n \end{array}}{K_1 \sqsubseteq \exists r_0^-.K_0 \qquad K_0 \sqsubseteq (\leqslant 1\ r_0\ A_1)}$$

Figure 1: Inference Rules

CEL, CB, and ELK (Baader, Lutz, and Suntisrivaraporn 2006; Kazakov 2009; Kazakov, Krötzsch, and Simančík 2011b). It thus establishes a promising foundation for actual implementations of finite-model reasoning in Horn-$\mathcal{ALCFI}$ and, via the reduction in Section 6, in Horn-$\mathcal{ALCQI}$. For simplicity, we start with a calculus for finite satisfiability and finite subsumption. An expansion to finite ABox consistency (and thus to finite instance checking) is sketched afterwards.

The calculus starts with a given TBox $\mathcal{T}$ and then exhaustively applies a set of inference rules. To ease their presentation, we assume that $\mathcal{T}$ is in a normal form that is slightly stricter than the one introduced in Section 2: in CIs $K \sqsubseteq \forall r.K'$ and $K \sqsubseteq (\leqslant 1\ r\ K)'$, K' must be a concept name A. The inference rules are displayed in Figure 1. They preserve the normal form and are applied in the sense that, if the concept inclusions in the precondition (above the line) are already present, then those in the postcondition (below the line) are added. Recall that K stands for a conjunction of concept names, which we read here modulo commutativity. Rule **R1** is applied only if $K \sqcap A$ *occurs* in the current (partially completed) TBox, that is, there is a CI of the form $K \sqcap A \sqsubseteq C$ or $K' \sqsubseteq \exists r.(K \sqcap A)$. The same is true for rule **R2** with K in place of $K \sqcap A$. In rule **R9**, $\oplus_n$ means addition modulo n.

We point out that rules **R1** to **R8** are minor variations of the corresponding rules in the calculus presented by Kazakov (2009), the main difference being that our language does not include role hierarchies. Rule **R9** is novel and deals with reversing cycles on the fly. Note that only the 'first edge' of each cycle is reversed, and that this is sufficient because the cycle can be rotated to make any edge the 'first' one.

Example 7 *Consider the TBox*

$$\mathcal{T} = \{A \sqsubseteq \exists r.(A \sqcap A_1 \sqcap \cdots \sqcap A_n), \quad (1)$$
$$A \sqsubseteq (\leqslant 1\ r^-\ A)\}. \quad (2)$$

Cycle reversion from Section 3 reverses all of the exponentially many cycles K, r, K with $K \subseteq S := \{A, A_1, \ldots, A_n\}$ and $A \in K$, adding $K \sqsubseteq \exists r^-.K$ and $K \sqsubseteq (\leqslant 1\ r\ K)$ for all such K. In contrast, the calculus avoids introducing 'irrelevant' conjunctions K and instead jointly reverses all these cycles by generating $A \sqsubseteq \exists r^-.S$ and $A \sqsubseteq (\leqslant 1\ r\ A)$:

$S \sqsubseteq A$	*from* **R1**	(3)
$A \sqsubseteq A$	*from* **R1**	(4)
$S \sqsubseteq \exists r.S$	*from* (1), (3), **R3**	(5)
$S \sqsubseteq (\leqslant 1\ r^-\ A)$	*from* (2), (3), **R3**	(6)
$S \sqsubseteq \exists r^-.S$	*and*	(7)
$S \sqsubseteq (\leqslant 1\ r\ A)$	*from* (3), (5), (6), **R9**	(8)
$A \sqsubseteq A_i$	*from* (1), (3), (4), (6), (7), **R8**	(9)
$A \sqsubseteq \exists r^-.S$	*from* (7), (9), **R3**	(10)
$A \sqsubseteq (\leqslant 1\ r\ A)$	*from* (8), (9), **R3**	(11)

Note that avoiding to introduce 'irrelevant' conjunctions K as illustrated by Example 7 is a main feature of consequence-based procedures which enables the excellent practical performance typically observed for this class of calculi.

The algorithm terminates after at most exponentially many rule applications since there are only exponentially many different concept inclusions that use the concept and role names of the original TBox. Each rule application can be performed in polynomial time, which is easy to see for the rules **R1**–**R8**. For **R9**, the crucial observation is that it suffices to consider all conjunctions K_0, K_1 and to check whether they are involved in *any* cycle. The latter can easily be done by a variation of directed graph reachability, where the nodes of the graph are the conjunctions that occur in the current TBox and the edges come from inclusions $K \sqsubseteq \exists r.K'$.

The following theorem, which is the main result of this section, states that the calculus is sound and complete.

Theorem 8 *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox, $\widehat{\mathcal{T}}$ be obtained by exhaustively applying Rules* **R1**–**R9***, and let A_0 be a concept name. Then A_0 is finitely satisfiable w.r.t. $\mathcal{T}$ iff $A_0 \sqsubseteq \bot \notin \widehat{\mathcal{T}}$.*

While Theorem 8 is formulated only for finite satisfiability, the algorithm can of course also be used to decide finite subsumption via the usual reduction to finite satisfiability. The following continues Example 7.

Example 9 *Let $\mathcal{T}$ be the TBox from Example 7 and*

$$\mathcal{T}' = \mathcal{T} \cup \{ \quad A \sqsubseteq \exists r.(A \sqcap X_1), \quad (12)$$
$$A \sqsubseteq \exists r.(A \sqcap X_2), \quad (13)$$
$$X_1 \sqcap X_2 \sqsubseteq \bot \quad \} \quad (14)$$

The calculus derives $A \sqsubseteq \bot$, thus A is finitely unsatisfiable w.r.t. $\mathcal{T}'$:[1]

$A \sqcap X_i \sqsubseteq A$	*from* **R1**	(15)
$A \sqsubseteq \exists r.(A \sqcap X_1 \sqcap X_2)$	*from* (11)–(13), (15), **R7**	(16)
$A \sqsubseteq \bot$	*from* (14), (16), **R6**	(17)

[1] A is obviously satisfiable w.r.t. $\mathcal{T}'$ in unrestricted models.

We now prove Theorem 8. The "only if" direction (soundness) is straightforward by verifying that each rule is sound in finite models. In contrast, the "if" direction (completeness) turns out to be surprisingly subtle to establish. The proof strategy is as follows. Assume that $A_0 \sqsubseteq \bot \notin \widehat{\mathcal{T}}$. We construct a (possibly infinite) model $\widehat{\mathcal{I}}$ of $\widehat{\mathcal{T}}$ with $A_0^{\widehat{\mathcal{I}}} \neq \emptyset$ and show that $\widehat{\mathcal{I}}$ is actually a model of $\mathcal{T}_\mathsf{f}$. By Theorem 3, it follows that A_0 is finitely satisfiable w.r.t. $\mathcal{T}$. From now on, assume w.l.o.g. that A_0 actually occurs in $\mathcal{T}$.

To construct $\widehat{\mathcal{I}}$, let $\mathsf{KON}(\widehat{\mathcal{T}})$ denote the set of all conjunctions K such that K occurs in $\widehat{\mathcal{T}}$ (in the sense explained above) and $K \sqsubseteq \bot \notin \widehat{\mathcal{T}}$. The domain $\Delta^{\widehat{\mathcal{I}}}$ consists of finite words $d = K_1 K_2 \cdots K_n \in \mathsf{KON}(\widehat{\mathcal{T}})^*$, and we use $\mathsf{tail}(d)$ to denote K_n. Define $\widehat{\mathcal{I}}$ by starting with

$$\Delta^{\widehat{\mathcal{I}}} = \mathsf{KON}(\widehat{\mathcal{T}})$$
$$A^{\widehat{\mathcal{I}}} = \{K \in \mathsf{KON}(\widehat{\mathcal{T}}) \mid K \sqsubseteq A \in \widehat{\mathcal{T}}\}$$
$$r^{\widehat{\mathcal{I}}} = \emptyset$$

Observe that since A_0 occurs in $\widehat{\mathcal{T}}$ and $A_0 \sqsubseteq \bot \notin \widehat{\mathcal{T}}$, $\Delta^{\widehat{\mathcal{I}}}$ contains the conjunction $K = A_0$ and thus $A_0^{\widehat{\mathcal{I}}} \neq \emptyset$. We finish the construction of $\widehat{\mathcal{I}}$ by exhaustively applying the following rule: if there is some $d \in \Delta^{\widehat{\mathcal{I}}}$ with $\mathsf{tail}(d) \sqsubseteq \exists r.K' \in \widehat{\mathcal{T}}$, K' maximal with this property, and $d \notin (\exists r.K')^{\widehat{\mathcal{I}}}$, then add a fresh element $e = dK'$ to $\Delta^{\widehat{\mathcal{I}}}$, add (d, K') to $r^{\widehat{\mathcal{I}}}$, and add dK' to $A^{\widehat{\mathcal{I}}}$ whenever $K' \sqsubseteq A \in \widehat{\mathcal{T}}$.

We first show that $\widehat{\mathcal{I}}$ is a model of $\widehat{\mathcal{T}}$, which amounts to a case distinction over the forms of CIs that can be present in $\widehat{\mathcal{T}}$, in each case relying on the fact that $\widehat{\mathcal{T}}$ is closed under the rules of the calculus. Details are provided in the appendix.

Lemma 10 $\widehat{\mathcal{I}} \models \widehat{\mathcal{T}}$.

It remains to show that $\widehat{\mathcal{I}}$ is a model of $\mathcal{T}_\mathsf{f}$, which is significantly more difficult to prove than Lemma 10 due to the fact that $\mathcal{T}_\mathsf{f}$ is obtained by reversing all cycles in $\mathcal{T}$ whereas the calculus is more careful to reverse only the 'relevant' ones, as explained above. We start with the observation that, when constructing $\mathcal{T}_\mathsf{f}$, it suffices to close only maximal cycles. More precisely, a cycle $K_1, r_1, K_2, \ldots, K_n$ in a TBox $\mathcal{T}$ is *maximal* if K_{j+1} is maximal with $\mathcal{T} \models K_j \sqsubseteq \exists r_j.K_{j+1}$, for $1 \leq j < n$. Let $\mathcal{T}_\mathsf{f}^{\max}$ be the variation of $\mathcal{T}_\mathsf{f}$ that is obtained by reversing only maximal cycles.

Lemma 11 *$\mathcal{T}_\mathsf{f}$ is equivalent to $\mathcal{T}_\mathsf{f}^{max}$.*

To finish the proof of Theorem 8, let $\mathcal{T}_\mathsf{f}^0, \mathcal{T}_\mathsf{f}^1, \ldots$ be the sequence of TBoxes obtained by starting with $\mathcal{T}_\mathsf{f}^0 = \mathcal{T}$ and then exhaustively closing maximal cycles, that is, $\mathcal{T}_\mathsf{f}^{\max}$ is the limit of this sequence. In the appendix, we prove by induction on i that $\widehat{\mathcal{I}}$ is a model of each $\mathcal{T}_\mathsf{f}^i$, thus of $\mathcal{T}_\mathsf{f}$.

We now briefly consider an extension of our algorithm to ABox consistency, with Figure 2 showing the additional rules. Instead of starting with only a TBox $\mathcal{T}$, the algorithm now begins with a set $\mathcal{T} \cup \mathcal{A}$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ an ABox, and then exhaustively applies rules **R1** to **R12**. In rules **R10**

R10 $\dfrac{K(a) \quad K \sqsubseteq A}{A(a)}$ **R11** $\dfrac{K(a) \quad r(a,b) \quad K \sqsubseteq \forall r.K'}{K'(b)}$

R12 $\dfrac{K_1(a) \quad K_2(a) \quad r(a,b) \quad K(b) \quad K_1 \sqsubseteq (\leqslant 1\ r\ A) \quad K_2 \sqsubseteq \exists r.K' \quad K \sqsubseteq A \quad K' \sqsubseteq A}{K'(b)}$

Figure 2: Additional Inference Rules

to **R12**, $K(a)$ is an abbreviation for $A_1(a) \cdots A_k(a)$ when $K = \{A_1, \ldots, A_k\}$. Recall that rules **R1** and **R2** only apply when the conjunction in their precondition occurs in the partially completed TBox. For the extension with ABoxes, an additional way for K to *occur* is that, for some ABox individual a, $K = \{A \mid A(a)$ is in the partial completion$\}$. It is easy to see that rule application still terminates after exponentially many steps. Let Γ be the set of concept inclusions and ABox assertions finally generated. The algorithm is sound and complete in the sense that $\mathcal{A}$ is finitely inconsistent w.r.t. $\mathcal{T}$ iff there is an ABox individual a and a conjunction K such that Γ contains both $K(a)$ and $K \sqsubseteq \bot$. To prove this, one updates the construction of $\widehat{\mathcal{I}}$ by starting with an initial interpretation defined by setting $\Delta^{\widehat{\mathcal{I}}} = \mathsf{Ind}(\mathcal{A})$, $r^{\widehat{\mathcal{I}}} = \{(a,b) \mid r(a,b) \in \mathcal{A}\}$, and $A^{\widehat{\mathcal{I}}} = \{a \in \mathsf{Ind}(\mathcal{A}) \mid A(a) \in \Gamma\}$. The rest of the construction of $\widehat{\mathcal{I}}$ is as before. It is not hard to adapt the proof of Lemma 10 to show that $\widehat{\mathcal{I}}$ satisfies all inclusions and assertions in Γ. As in the case of finite satisfiability, it thus remains to prove that $\widehat{\mathcal{I}}$ is a model of $\mathcal{T}_\mathsf{f}$. Fortunately, the proof of goes through without modification.

Apart from providing a basis for practical implementations, our algorithm also yields an EXPTIME upper bound for finite ABox consistency in Horn-$\mathcal{ALCFI}$. This result is known from (Lutz, Sattler, and Tendera 2005), where it is shown that ABox consistency in the non-Horn version of $\mathcal{ALCQI}$ is in EXPTIME. A matching lower bound can be derived from (Baader, Brandt, and Lutz 2008) where an EXPTIME lower bound is established for unrestricted subsumption in (the $\mathcal{ELI}$ fragment of) Horn-$\mathcal{ALCFI}$; the proof can easily be adapted to finite satisfiability.

Theorem 12 *Finite satisfiability and finite ABox consistency in Horn-$\mathcal{ALCQI}$ are* EXPTIME*-complete.*

5 Query Answering in the Finite

In the ontology-based data access (OBDA) paradigm, the central reasoning problem is answering database-style queries over ABoxes in the presence of a DL TBox. In this section, we study the finite model version of this problem, assuming that queries are positive existential queries (PEQs) and that TBoxes are formulated in Horn-$\mathcal{ALCFI}$. We show that, as in the case of ABox consistency, finite PEQ answering can be reduced to unrestricted PEQ answering by reversing finmod cycles in the TBox. This result enables the use of algorithms for unrestricted PEQ answering also in the finite case. It also allows us to show that finite PEQ answering w.r.t. Horn-$\mathcal{ALCFI}$ TBoxes is EXPTIME-complete regarding combined complexity, and PTIME-complete regarding data complexity.

We start with a brief introduction of positive existential queries and of query answering. For simplicity, we concentrate on Boolean queries, that is, queries without answer variables. It is, however, easy to adapt all techniques established in this section to the case of queries with answer variables. A *(Boolean) positive existential query (PEQ)* q takes the form $\exists \mathbf{x}\, \varphi(\mathbf{x})$ where φ is built from atoms of the form $A(x)$ and $r(x,y)$ using conjunction and disjunction, with x, y variables from $\mathbf{x}$, A a concept name, and r a role name. Let $\mathcal{I}$ be an interpretation and $q = \exists \mathbf{x}\, \varphi$ a PEQ. A *match* of q in $\mathcal{I}$ is a mapping $\pi : \mathbf{x} \to \Delta^{\mathcal{I}}$ such that φ evaluates to true unter the valuation that assigns true to an atom $A(x)$ in φ iff $\pi(x) \in A^{\mathcal{I}}$ and true to an atom $r(x,y)$ in φ iff $(\pi(x), \pi(y)) \in r^{\mathcal{I}}$. We write $\mathcal{I} \models q$ if there is a match of q in $\mathcal{I}$. For an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, we write $\mathcal{A}, \mathcal{T} \models q$ (resp. $\mathcal{A}, \mathcal{T} \models_{\mathsf{fin}} q$) if $\mathcal{I} \models q$ for all models (resp. finite models) $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$. We then say that $\mathcal{A}, \mathcal{T}$ *entails* (resp. *finitely entails*) q. The problem that we are interested in is *finite query entailment*, that is, given an ABox $\mathcal{A}$, a TBox $\mathcal{T}$, and a query q, to decide whether $\mathcal{A}, \mathcal{T} \models_{\mathsf{fin}} q$. We will study both the combined complexity and the data complexity of this problem. When studying combined complexity, all of $\mathcal{A}$, $\mathcal{T}$, and q are considered an input. In the case of data complexity, $\mathcal{T}$ and q are assumed to be fixed and q is the only input.

The main result of this section is the following theorem, where $\mathcal{T}_\mathsf{f}$ is the TBox obtained from $\mathcal{T}$ by exhaustively reversing finmod cycles, exactly as in Section 3.

Theorem 13 *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCFI}$ TBox and $\mathcal{A}$ an ABox that is finitely consistent w.r.t. $\mathcal{T}$. For any PEQ q,*

$$\mathcal{A}, \mathcal{T} \models_{\mathsf{fin}} q \text{ iff } \mathcal{A}, \mathcal{T}_\mathsf{f} \models q$$

The proof of the "$\Leftarrow$" direction is trivial. Indeed, if $\mathcal{A}, \mathcal{T} \not\models_{\mathsf{fin}} q$, then there is a finite model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ such that $\mathcal{I} \not\models q$. Since every finite model of $\mathcal{T}$ is also a model of $\mathcal{T}_\mathsf{f}$ by Lemma 4, it follows that $\mathcal{A}, \mathcal{T}_\mathsf{f} \not\models q$.

For the proof of the "$\Rightarrow$" direction, we use a well-known (infinite) canonical model $\mathcal{U}$ of $\mathcal{A}$ and $\mathcal{T}_\mathsf{f}$, constructed by starting with the following initial interpretation

$$\begin{aligned} \Delta^{\mathcal{U}} &= \mathsf{Ind}(\mathcal{A}) \\ A^{\mathcal{U}} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{A}, \mathcal{T}_\mathsf{f} \models A(a)\} \\ r^{\mathcal{U}} &= \{(a,b) \mid r(a,b) \in \mathcal{A}\} \end{aligned}$$

and then exhaustively applying the following completion rule: for all $d \in \Delta^{\mathcal{U}}$ such that $\mathcal{T}_\mathsf{f} \models \mathsf{tp}_{\mathcal{U}}(d) \sqsubseteq \exists r.t'$, where t' is maximal with this property and $d \notin (\exists r.t')^{\mathcal{U}}$, add a fresh element d' to $\Delta^{\mathcal{U}}$, the edge (d,d') to $r^{\mathcal{U}}$, and d' to the interpretation $A^{\mathcal{U}}$ of all concept names $A \in t'$.

The following properties of $\mathcal{U}$ are well-known and the reason for why $\mathcal{U}$ is called canonical (Krisnadhi and Lutz 2007; Eiter et al. 2008; Ortiz, Rudolph, and Šimkus 2011).

Lemma 14

1. *$\mathcal{U}$ is a model of $\mathcal{A}$ and of $\mathcal{T}_\mathsf{f}$;*
2. *For any PEQ q, we have that $\mathcal{A}, \mathcal{T}_\mathsf{f} \models q$ iff $\mathcal{U} \models q$.*

By Point 2 of Lemma 14, we can establish the "⇒" direction of Theorem 13 by showing that $\mathcal{A}, \mathcal{T} \models_{\mathsf{fin}} q$ implies $\mathcal{U} \models q$. The proof makes intense use of homomorphisms. For interpretations $\mathcal{I}_1, \mathcal{I}_2$, a *homomorphism from $\mathcal{I}_1$ to $\mathcal{I}_2$* is a function $h : \Delta^{\mathcal{I}_1} \to \Delta^{\mathcal{I}_2}$ such that

1. $h(a) = a$ for all $a \in \mathsf{N_I}$;
2. $d \in A^{\mathcal{I}_1}$ implies $h(d) \in A^{\mathcal{I}_2}$ for all concept names A;
3. $(d, e) \in r^{\mathcal{I}_1}$ implies $(h(d), h(e)) \in r^{\mathcal{I}_2}$ for all (possibly inverse) roles r.

For $n > 0$, an *n-substructure* of an interpretation $\mathcal{I}$ is an interpretation $\mathcal{I}'$ obtained from $\mathcal{I}$ by selecting a domain $\Delta^{\mathcal{I}'} \subseteq \Delta^{\mathcal{I}}$ with at most n elements and restricting $\mathcal{I}$ to $\Delta^{\mathcal{I}'}$. To show that $\mathcal{A}, \mathcal{T} \models_{\mathsf{fin}} q$ implies $\mathcal{U} \models q$, it suffices to establish the following.

Proposition 15 *For every $n_0 > 0$, there is a finite model $\mathcal{J}_{n_0}$ of $\mathcal{A}$ and $\mathcal{T}$ such that there is a homomorphism from any n_0-substructure of $\mathcal{J}_{n_0}$ to $\mathcal{U}$.*

In fact, $\mathcal{A}, \mathcal{T} \models_{\mathsf{fin}} q$ implies $\mathcal{J}_{n_0} \models q$ and thus there is an n_0-substructure $\mathcal{J}$ of $\mathcal{J}_{n_0}$ with $\mathcal{J} \models q$, where n_0 is the number of variables in q. The latter is witnessed by a match π. By Proposition 15, there is a homomorphism h from $\mathcal{J}$ to $\mathcal{U}$ and thus a match of q in $\mathcal{U}$ can be found by composing π with h.

We construct the model $\mathcal{J}$ from Proposition 15 by modifying the finite model $\mathcal{I}$ constructed in Section 3. For two reasons, the finite model $\mathcal{I}$ constructed in Section 3 need not satisfy the condition formulated for $\mathcal{J}_{n_0}$ in Proposition 15.

1. $\mathcal{I}$ can contain paths of length $\leq n_0$ that do not exist in $\mathcal{U}$.
2. $\mathcal{I}$ can contain cycles that do not exclusively consist of ABox elements, while no such cycles are present in $\mathcal{U}$.

Let us start with Problem 1 above. There are, in turn, two sources for paths in $\mathcal{I}$ that we cannot reproduce in $\mathcal{U}$.

(i) Application of $(\mathbf{c3})$ can generate a path $(d_1, d) \in r^{\mathcal{I}}$, $(d, d_2) \in s^{\mathcal{I}}$ such that $\mathsf{tp}_{\mathcal{I}}(d_1) \to_r \mathsf{tp}_{\mathcal{I}}(d) \ {}_s\!\leftarrow \mathsf{tp}_{\mathcal{I}}(d_2)$ and d is not identified by an ABox element. Such situations are not necessarily reproducible in $\mathcal{U}$. As a concrete example, consider

$$\mathcal{A} = \{\ B_1(a), B_2(b)\ \}$$
$$\mathcal{T} = \{\ B_1 \sqsubseteq \exists r.A,\ B_2 \sqsubseteq \exists r.A\ \}.$$

The problematic path is $(a, d_t) \in r^{\mathcal{I}}, (d_t, b) \in (r^-)^{\mathcal{I}}$ with $t = \{A\}$.

(ii) Application of $(\mathbf{c2})$ can result in similar a situation as above, but where the middle element d is replaced with a sequence of elements $e_0, \dots, e_k$ such that $(e_i, e_{i+1}) \in r_i^{\mathcal{I}}$ for all $i < k$ (for some roles $r_0, \dots, r_{k-1}$) and

$$\mathsf{tp}_{\mathcal{I}}(e_0) \ {}^1\!\leftrightarrow^1_{r_1} \cdots {}^1\!\leftrightarrow^1_{r_{k-1}} \mathsf{tp}_{\mathcal{I}}(e_k). \tag{18}$$

For a very simple example, take

$$\mathcal{A} = \{\ B_1(a),\ B_2(b)\ \}$$

and assume that $\mathcal{T}$ is such that $B_1 \ {}^1\!\leftrightarrow^1_r \ B_2$. Then an application of $(\mathbf{c2})$ will simply add $r(a, b)$, an edge that does not exist in $\mathcal{U}$.

To obtain the desired model $\mathcal{J}_{n_0}$ from Proposition 15, we first solve Problems (i) and (ii) above, and then Problem 2. To make precise what we mean by this, we introduce bounded simulations, a weakening of homomorphisms. A *bounded simulation of $\mathcal{I}_1$ in $\mathcal{I}_2$* is a relation $\rho \subseteq \Delta^{\mathcal{I}_1} \times \mathbb{N} \times \Delta^{\mathcal{I}_2}$ such that for all $(d, i, e) \in \rho$, the following conditions are satisfied:

1. if $d \in A^{\mathcal{I}_1}$, then $e \in A^{\mathcal{I}_2}$;
2. if $i > 0$ and $(d, d') \in r^{\mathcal{I}_1}$ for some (possibly inverse) role r, then there is an $e' \in \Delta^{\mathcal{I}_2}$ with $(e, e') \in r^{\mathcal{I}_2}$ and $(d', i-1, e') \in \rho$.

We write $(\mathcal{I}_1, d) \preceq_k (\mathcal{I}_2, e)$, for $d \in \Delta^{\mathcal{I}_1}$ and $e \in \Delta^{\mathcal{I}_2}$, if there is a bounded simulation of $\mathcal{I}_1$ in $\mathcal{I}_2$ such that $(d, k, e) \in \rho$ and for all $a \in \mathsf{N_I} \cap \Delta^{\mathcal{I}_1}$, we have $(a, k, a) \in \rho$. Then $\mathcal{I}_1 \preceq_k \mathcal{I}_2$ denotes that for every $d \in \Delta^{\mathcal{I}_1}$, there is an $e \in \Delta^{\mathcal{I}_2}$ with $(\mathcal{I}_1, d) \preceq_k (\mathcal{I}_2, e)$. We write $(\mathcal{I}_1, d) \sim_k (\mathcal{I}_2, e)$ if $(\mathcal{I}_1, d) \preceq_k (\mathcal{I}_2, e)$ and vice versa.

With solving Problems (i) and (ii), we mean to establish the following intermediate result.

Proposition 16 *For every $n_0 > 0$, there is a finite model $\mathcal{I}_{n_0}$ of $\mathcal{A}$ and $\mathcal{T}$ such that $\mathcal{I}_{n_0} \preceq_{n_0} \mathcal{U}$.*

To remove the undesired paths illustrated in (i) above, we modify the construction of $\mathcal{I}$ by replacing the elements d_t, $t \in \mathsf{TP}(\mathcal{T}_{\mathsf{f}})$, that are introduced at the beginning of the construction of $\mathcal{I}$ and used as 'targets' for role edges introduced by applications of $(\mathbf{c3})$. In the modified construction, we instead introduce one $(\mathbf{c3})$-target for each *n_0-bounded simulation type*, which is an equivalence class of $\sim_{n_0}$ on the set of all pointed interpretations $(\mathcal{I}_1, d)$. In the example given in (i) above, the result is that the two existential restrictions would no longer be witnessed by the same d_t because the 1-simulation type of the witnesses are different (one has an r-predecessor in B_1, the other in B_2). Since simulations need only to consider symbols that occur in the (fixed) ABox $\mathcal{A}$ and (fixed) TBox $\mathcal{T}$, there are only finitely many n_0-simulation types and thus finiteness of $\mathcal{I}$ is not compromised.

Undesired paths of type (ii) are avoided by modifying the $(\mathbf{c2})$ rule so that the sequences (18) are of length exceeding n_0 and thus the highlighted problem which involves both ends of the sequence is not 'visible' in n_0-substructures. We also include an initial piece of the canonical model $\mathcal{U}$ for $\mathcal{A}$ and $\mathcal{T}_{\mathsf{f}}$ of depth n_0 in the initial version of $\mathcal{I}$ to avoid the undesired 'shortcuts' between ABox elements illustrated by the example given in (ii) above.

The construction is spelled out in full detail in the appendix. We have actually omitted some aspects in the overview above for the sake of a clearer exposition, such as the fact that we first exhaustively apply rules $(\mathbf{c1})$ and $(\mathbf{c2})$, followed by exhaustive application of $(\mathbf{c3})$ (the latter two in their modified versions), and that we actually cannot include in the initial $\mathcal{I}$ *all* n_0-bounded simulation types, but must select only the 'relevant' ones. This finishes the proof of Proposition 16.

To solve Problem 2 above and thus obtain the model $\mathcal{J}_{n_0}$ stipulated by Proposition 15, we have to eliminate all non-ABox-cycles of size at most n_0 in the model $\mathcal{I}_{n_0}$ delivered by Proposition 16. This is achieved by taking the product with a suitable finite group of high girth, a technique championed

by Otto (2012). Details are provided in the appendix. This finishes the proof of Theorem 13.

Apart from enabling the use of algorithms for unrestricted PEQ answering also in the finite case, Theorem 13 yields tight complexity bounds for finite PEQ entailment.

Theorem 17 *Finite PEQ entailment in Horn-$\mathcal{ALCFI}$ is decidable,* EXPTIME*-complete in combined complexity, and* PTIME*-complete in data complexity.*

Proof.(sketch) For the unrestricted case, an EXPTIME lower bound is in (Baader, Brandt, and Lutz 2008) and a PTIME one in (Calvanese et al. 2006). Both results can easily be adapted to the finite case. The upper bounds can be proved using the following straightforward algorithm for PEQ entailment, which resembles existing algorithms such as those presented in (Krisnadhi and Lutz 2007; Eiter et al. 2008; Calì, Gottlob, and Lukasiewicz 2009; Ortiz, Rudolph, and Šimkus 2011). Assume that an input ABox $\mathcal{A}$, TBox $\mathcal{T}$, and PEQ q are given, and let n_0 be the number of variables in q. As a consequence of Theorem 3, finite satisfiability w.r.t. $\mathcal{T}$ coincides with unrestricted satisfiability w.r.t. $\mathcal{T}_\mathsf{f}$. Using our algorithm for computing finite satisfiability in Horn-$\mathcal{ALCFI}$ in EXPTIME, we can thus compute the set $\mathsf{TP}(\mathcal{T}_\mathsf{f})$ of types for $\mathcal{T}_\mathsf{f}$ without computing $\mathcal{T}_\mathsf{f}$ or explicitly reasoning w.r.t. this exponentially large TBox. Let $\mathcal{A}'$ be the extension of $\mathcal{A}$ with assertions $\{A(a_t) \mid A \in t\}$ for each $t \in \mathsf{TP}(\mathcal{A})$. Now compute an initial piece $\mathcal{U}'$ of the canonical model $\mathcal{U}$ of $\mathcal{A}'$ and $\mathcal{T}_\mathsf{f}$, namely its restriction to depth n_0. Similar to the computation of $\mathsf{TP}(\mathcal{T}_\mathsf{f})$ above, we can do this by using finite subsumption w.r.t. $\mathcal{T}$ instead of unrestricted subsumption w.r.t. $\mathcal{T}_\mathsf{f}$. It is not difficult to prove that $\mathcal{U}' \models q$ iff $\mathcal{U} \models q$. To check whether $\mathcal{U}' \models q$ within the desired time bounds, we can simply enumerate all possible maps of variables in q to elements of $\mathcal{U}'$ and check whether any such map is a match. ❑

Note that decidability of PEQ entailment in Horn-$\mathcal{ALCFI}$ was expected given a result by Pratt-Hartmann which states that finite CQ answering for the two-variable guarded fragment of first-order logic extended with counting quantifiers is decidable (Pratt-Hartmann 2009). We assume that his proof can be extended to unions of conjunctive queries (UCQs), thus to PEQs. Pratt-Hartmann also analyses the data complexity of finite CQ answering in his logic, but finds it to be CONP-complete. He does not analyse combined complexity. Theorem 17 suggests that PEQ entailment in Horn-$\mathcal{ALCFI}$ has the same complexity in finite and in unrestricted models. For the unrestricted case, PTIME-completeness in data complexity follows from the results in (Hustadt, Motik, and Sattler 2007), and EXPTIME-completeness in combined complexity is proved in (Eiter et al. 2008) for UCQs. We assume that the techniques in that paper extend to PEQs.

6 From Horn-$\mathcal{ALCFI}$ to Horn-$\mathcal{ALCQI}$

Our results for finite satisfiability and finite subsumption (the reasoning tasks that do not involve ABoxes) extend in a straightforward way from Horn-$\mathcal{ALCFI}$ to Horn-$\mathcal{ALCQI}$. In particular, we can convert a Horn-$\mathcal{ALCQI}$ TBox $\mathcal{T}$ into a Horn-$\mathcal{ALCFI}$ TBox $\mathcal{T}'$ such that finite (un)satisfiability is preserved by replacing each CI $K \sqsubseteq (\geqslant n\ r\ K')$ in $\mathcal{T}$ with the following inclusions, for $1 \leq i < j \leq n$:

$$K \sqsubseteq \exists r.B_i, \quad B_i \sqsubseteq K', \quad B_i \sqcap B_j \sqsubseteq \bot \qquad (*)$$

While an easy unraveling argument can be used to prove that this reduction is correct in the presence of infinite models, more care is required in the finite case (see appendix).

Proposition 18 *$\mathcal{T}$ is finitely satisfiable iff $\mathcal{T}'$ is finitely satisfiable.*

It follows from Proposition 18 and Theorem 3 that a Horn-$\mathcal{ALCQI}$ TBox $\mathcal{T}$ is finitely satisfiable iff $(\mathcal{T}')_\mathsf{f}$ is satisfiable. Actually, it is not hard to see that this is the case iff $\mathcal{T}_\mathsf{f}$ (the result of applying cycle reversion directly to the Horn-$\mathcal{ALCQI}$ TBox, ignoring all inclusions $A \sqsubseteq (\geqslant n\ r\ C)$) is satisfiable because if any of the existential restrictions in $\mathcal{T}' \setminus \mathcal{T}$ is involved in a finmod cycle, then a simple semantic argument shows that both $\mathcal{T}_\mathsf{f}$ and $(\mathcal{T}')_\mathsf{f}$ are unsatisfiable. Proposition 18 also enables the use of our consequence-based procedure for deciding finite satisfiability in Horn-$\mathcal{ALCQI}$.

It is not immediately obious how to extend $(*)$ and Proposition 18 to ABox consistency and instance checking. We believe, though, that it is not too hard to modify the proof of Theorem 3 for Horn-$\mathcal{ALCQI}$, to adapt the consequence-based procedure to allow a direct treatment of Horn-$\mathcal{ALCQI}$ TBoxes without prior reduction to Horn-$\mathcal{ALCFI}$, and to extend all model constructions underlying our results about PEQ entailment to Horn-$\mathcal{ALCQI}$. In particular, such a direct approach should yield EXPTIME/ PTIME upper bounds for PEQ entailment in Horn-$\mathcal{ALCQI}$ even when the numbers in at least restrictions are coded in binary (note that, in this case, the translation $(*)$ incurs an exponential blowup).

7 Future Work

As future research, it would be interesting to extend the results in this paper to Horn-$\mathcal{SHIQ}$, that is, to add role hierarchies and transitive roles. Reducing out role hierarchies does not seem easily possible in the finite,[2] so they would have to be built directly into all constructions. For query entailment, we expect transitive roles to cause significant additional challenges, see for example (Eiter et al. 2009; Mosurovic et al. 2013). In particular, transitive roles result in an additional way in which the finite model property is lost, illustrated by the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.A, \mathsf{trans}(r)\}$ and the conjunctive query $q = \exists x\, r(x,x)$. We have $\{A(a)\}, \mathcal{T} \not\models q$, but $\{A(a)\}, \mathcal{T} \models_\mathsf{fin} q$ although neither counting nor inverse roles are present (the TBox $\mathcal{T}$ is formulated in the DL $\mathcal{EL}_\mathsf{trans}$). Finite model reasoning in versions of Datalog$^\pm$ that extend $\mathcal{EL}_\mathsf{trans}$ has recently been studied in (Gogacz and Marcinkowski 2013b; 2013a).

In this paper, we have not analyzed the size of finite models. It is, however, easy to prove a double exponential lower bound on the size of finite models for satisfiability in Horn-$\mathcal{ALCFI}$ by enforcing a tree of exponential depth in which no two elements can be identical. A matching upper bound follows from Pratt-Hartmann's result that every finitely satisfiable formula in first-order logic with two variables and

[2] In contrast to what we have claimed in the workshop predecessor of this paper (Ibáñez-García, Lutz, and Schneider 2013).

counting quantifiers has a model of at most double exponential size (Pratt-Hartmann 2005). Analyzing the size of finite (counter)models for query entailment is left as future work.

Acknowledgements Carsten Lutz was supported by the SFB/TR8 *Spatial Cognition*.

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI-05*, 364–369.

Baader, F.; Brandt, S.; and Lutz, C. 2008. Pushing the $\mathcal{EL}$ envelope further. In *Proc. OWLED-08 DC*, volume 496 of *CEUR-WS.org*.

Baader, F.; Lutz, C.; and Suntisrivaraporn, B. 2006. CEL – a polynomial-time reasoner for life science ontologies. In *Proc. IJCAR-06*, volume 4130 of *LNCS*, 287–291. Springer.

Bienvenu, M.; Lutz, C.; and Wolter, F. 2013. First-order rewritability of atomic queries in Horn description logics. In *Proc. of IJCAI-13*. IJCAI/AAAI.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2009. A general datalog-based framework for tractable query answering over ontologies. In *PODS*, 77–86.

Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2006. Data complexity of query answering in description logics. In *Proc. KR-06*, 260–270. AAAI Press.

Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning* 39(3):385–429.

Calvanese, D. 1996. Finite model reasoning in description logics. In *Proc. of KR-96*, 292–303. Morgan Kaufmann.

Cosmadakis, S.; Kanellakis, P.; and Vardi, M. 1990. Polynomial-time implication problems for unary inclusion dependencies. *J. ACM* 37(1):15–46.

Eiter, T.; Gottlob, G.; Ortiz, M.; and Simkus, M. 2008. Query answering in the description logic Horn-$\mathcal{SHIQ}$. In *Proc. JELIA-08*, volume 5293 of *LNCS*, 166–179. Springer.

Eiter, T.; Lutz, C.; Ortiz, M.; and Simkus, M. 2009. Query answering in description logics with transitive roles. In *Proc. IJCAI-09*, 759–764.

Eiter, T.; Ortiz, M.; Šimkus, M.; Tran, T.-K.; and Xiao, G. 2012. Towards practical query answering for Horn-$\mathcal{SHIQ}$. In *Proc. DL-12*, volume 846 of *CEUR-WS.org*.

Gogacz, T., and Marcinkowski, J. 2013a. Converging to the chase - a tool for finite controllability. In *Proc. LICS-13*, 540–549. IEEE Computer Society.

Gogacz, T., and Marcinkowski, J. 2013b. On the BDD/FC conjecture. In *Proc. PODS-13*, 127–138. ACM.

Hustadt, U.; Motik, B.; and Sattler, U. 2007. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning* 39(3).

Ibáñez-García, Y. A.; Lutz, C.; and Schneider, T. 2013. Finite model reasoning in Horn-$\mathcal{SHIQ}$. In *Proc. DL-13*, volume 1014 of *CEUR-WS.org*, 234–245.

Kazakov, Y.; Krötzsch, M.; and Simančík, F. 2011a. Unchain my $\mathcal{EL}$ reasoner. In *Proc. DL-11*, volume 745 of *CEUR-WS.org*.

Kazakov, Y.; Krötzsch, M.; and Simančík, F. 2011b. Concurrent classification of el ontologies. In *Proc. ISWC-11 (1)*, volume 7031 of *LNCS*, 305–320. Springer.

Kazakov, Y. 2009. Consequence-driven reasoning for Horn-$\mathcal{SHIQ}$ ontologies. In *Proc. IJCAI-09*, 2040–2045.

Krisnadhi, A., and Lutz, C. 2007. Data complexity in the $\mathcal{EL}$ family of DLs. In *Proc. DL-07*, volume 250 of *CEUR-WS.org*.

Lutz, C., and Wolter, F. 2012. Non-uniform data complexity of query answering in description logics. In *Proc. KR-12*. AAAI Press.

Lutz, C.; Sattler, U.; and Tendera, L. 2005. The complexity of finite model reasoning in description logics. *Information and Computation* 199:132–171.

Mosurovic, M.; Krdzavac, N.; Graves, H.; and Zakharyaschev, M. 2013. A decidable extension of $\mathcal{SROIQ}$ with complex role chains and unions. *J. Artif. Intell. Res. (JAIR)* 47:809–851.

Ortiz, M.; Rudolph, S.; and Šimkus, M. 2011. Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In *Proc. IJCAI-11*, 1039–1044. IJCAI/AAAI.

Otto, M. 2012. Highly acyclic groups, hypergraph covers, and the guarded fragment. *J. ACM* 59(1):5.

Pacholski, L.; Szwast, W.; and Tendera, L. 2000. Complexity results for first-order two-variable logic with counting. *SIAM J. Comput.* 29(4):1083–1117.

Pratt-Hartmann, I. 2005. Complexity of the two-variable fragment with counting quantifiers. *J. of Logic, Language and Information* 14(3):369–395.

Pratt-Hartmann, I. 2009. Data-complexity of the two-variable fragment with counting quantifiers. *Inf. Comput.* 207(8):867–888.

Rosati, R. 2008. Finite model reasoning in *DL-Lite*. In *Proc. ESWC-08*, volume 5021 of *LNCS*, 215–229. Springer.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Exact Learning of Lightweight Description Logic Ontologies

Boris Konev
University of Liverpool
United Kingdom

Carsten Lutz
Universität Bremen
Germany

Ana Ozaki
University of Liverpool
United Kingdom

Frank Wolter
University of Liverpool
United Kingdom

Abstract

We study learning of description logic TBoxes in Angluin et al.'s framework of exact learning via queries. We admit entailment queries ("is a given subsumption entailed by the target TBox?") and equivalence queries ("is a given TBox equivalent to the target TBox?"), assuming that the signature and logic of the target TBox are known. We present three main results: (1) TBoxes formulated in DL-Lite with role inclusions and composite concepts on the right-hand side of concept inclusions can be learned in polynomial time; (2) $\mathcal{EL}$ TBoxes with only concept names on the right-hand side of concept inclusions can be learned in polynomial time; and (3) $\mathcal{EL}$ TBoxes cannot be learned in polynomial time. It follows that non-polynomial time learnability of $\mathcal{EL}$ TBoxes is caused by the interaction between existential restrictions on the right- and left-hand sides of concept inclusions. We also show that neither entailment nor equivalence queries alone are sufficient in cases (1) and (2) above.

Successfully deploying a description logic (DL) in a concrete application requires to carefully capture the relevant domain knowledge in a DL ontology. This is a subtle, error-prone, and time consuming task, which is further hindered by the fact that domain experts are rarely experts in ontology engineering and, conversely, ontology engineers are often not sufficiently familiar with the domain to be modeled. From its beginnings, DL research was driven by the aim to provide various forms of support for ontology engineers, assisting them in the design of high-quality ontologies. Examples include the ubiquitous task of ontology classification (Baader et al. 2003), bootstrapping ontology design from examples, data, or text (Baader and Molitor 2000; Borchmann and Distel 2011; Ma and Distel 2013) and checking the completeness of the modeling in a systematic way (Baader et al. 2007).

Machine learning (ML) techniques are natural candidates to support ontology engineering by partly automatizing the design process, and in fact ML approaches have been proposed for ontology engineering in non-DL-contexts. There, the main focus is on mining the relevant terms of the application domain from bodies of text and semi-structured data with the aim of including them in the ontology. Some approaches also support learning of subsumptions between the identified terms, but not between compound logical expressions based on these terms (Cimiano, Hotho, and Staab 2005; Buitelaar, Cimiano, and Magnini 2005). In the context of DL ontologies, though, the main objective is to develop a detailed modeling of the relevant concepts of the domain using the logical operators provided by the DL at hand (Voelker 2009). In this paper, we thus concentrate on learning the full logical structure of a DL TBox. In particular, we study the foundations of learning ontologies that are formulated in several DLs which underly the three tractable *profiles* of the OWL2 ontology language.

We adopt Angluin et al.'s framework of exact learning via queries (Angluin 1987c). In particular, we are interested in learning a target TBox $\mathcal{T}$ that is formulated in a given DL and whose signature Σ (the concept and role names that occur in $\mathcal{T}$) is also known, by posing queries to an oracle. Intuitively, the oracle can be thought of as a domain expert who interacts with the learning algorithm. We consider two forms of oracle queries:

- *entailment queries*: does $\mathcal{T}$ entail a given Σ concept inclusion $C \sqsubseteq D$? These queries are answered by the oracle with 'yes' or 'no'.
- *equivalence queries*: is a given Σ-TBox $\mathcal{H}$ (the hypothesis) equivalent to $\mathcal{T}$? The oracle answers 'yes' if $\mathcal{H}$ and $\mathcal{T}$ are logically equivalent and 'no' otherwise. It then returns a Σ-inclusion $C \sqsubseteq D$ such that $\mathcal{T} \models C \sqsubseteq D$ and $\mathcal{H} \not\models C \sqsubseteq D$ (a *positive counterexample*), or vice versa (a *negative counterexample*).

We generally assume that the inclusions used in entailment queries and returned by equivalence queries are of the same form as the inclusions allowed in the target TBox. While, in general, answering an equivalence query of the above form can be expected to be rather challenging also for a domain expert and an ontology engineer, we emphasize that the actual algorithms developed in this paper use equivalence queries only in a very restricted way. In particular, our algorithms always ensure that the hypothesis TBox $\mathcal{H}$ that they maintain is a logical consequence of the target TBox $\mathcal{T}$, and then use equivalence queries only to ask 'is the TBox $\mathcal{H}$ learned so far complete and if not, please give me a missing inclusion $C \sqsubseteq D$'.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Following Angluin, our aim is to find (deterministic) learning algorithms that run in polynomial time and consequently also make only polynomially many oracle queries. More precisely, we require that there is a two-variable polynomial $p(n, m)$ such that at every point of the algorithm execution, the time spent up to that point is bounded by $p(n, m)$, where n is the size of the target TBox $\mathcal{T}$ to be learned and m is the size of the largest (positive or negative) counterexample that was returned by the oracle until the considered point of the execution. If such an algorithm exists, then $\mathcal{L}$ TBoxes are *polynomial time learnable*.

Within the setup laid out above, we study three different DLs, which are significant fragments of the three profiles of OWL2 (Motik et al. 2009). The popular description logic $\mathcal{EL}$ can be viewed as a logical core of the OWL2 EL profile of OWL2. It is known that propositional Horn formulas, which correspond to $\mathcal{EL}$ TBoxes without existential restrictions, can be learned in polynomial time when both entailment and equivalence queries are available, but not when one of these types of queries is disallowed (Frazier and Pitt 1993; Angluin, Frazier, and Pitt 1992; Angluin 1987a), see also (Arias and Balcázar 2011). We show that, in contrast, $\mathcal{EL}$ TBoxes are not polynomial time learnable. The results for the other two DLs that we study will shed some light on the reasons for this behaviour.

The DL-Lite family of DLs underlies the OWL2 QL profile of OWL2 (Calvanese et al. 2007). In many versions of DL-Lite such as in DL-Lite$_\mathcal{R}$ (DL-Lite with role inclusions), the number of Σ-concept inclusions is polynomial in the size of Σ. In this case, TBoxes are trivially learnable in polynomial time, even when only entailment queries (but no equivalence queries) are available or vice versa. We thus consider the more interesting DL-Lite$_\mathcal{R}^\exists$ dialect, that is, the extension of DL-Lite$_\mathcal{R}$ that allows any $\mathcal{ELI}$ concept on the right-hand side of concept inclusions. Note that DL-Lite$_\mathcal{R}^\exists$ can be viewed as a logical core of OWL2 QL. Here, the number of Σ-concept inclusions is infinite, but we are nevertheless able to establish that DL-Lite$_\mathcal{R}^\exists$ TBoxes are polynomial time learnable. In practice, many $\mathcal{EL}$ TBoxes actually fall within (the intersection of $\mathcal{EL}$ and) DL-Lite$_\mathcal{R}^\exists$. We also show that DL-Lite$_\mathcal{R}^\exists$ TBoxes cannot be learned in polynomial time with entailment queries alone or with equivalence queries alone.

We then consider the fragment $\mathcal{EL}_\mathsf{lhs}$ of $\mathcal{EL}$ where only concept names (but no compound concepts) are admitted on the right-hand side of concept inclusions. $\mathcal{EL}_\mathsf{lhs}$ is a significant fragment of the OWL2 RL profile of OWL2, and also a fragment of datalog. We prove that $\mathcal{EL}_\mathsf{lhs}$ TBoxes can be learned in polynomial time using a non-trivial extension of Angluin's polynomial time algorithm for learning propositional Horn theories (1992). Note that the symmetric fragment $\mathcal{EL}_\mathsf{rhs}$ that allows only concept names on the *left*-hand side of concept inclusions is a fragment of DL-Lite$_\mathcal{R}^\exists$. Together, our results for DL-Lite$_\mathcal{R}^\exists$ and $\mathcal{EL}_\mathsf{lhs}$ thus show that non-polynomial time learnability of $\mathcal{EL}$ TBoxes is caused by the interaction between existential restrictions on the right- and left-hand sides of concept inclusions.

To improve readability, we first present the upper bounds (polynomial time learnability for DL-Lite$_\mathcal{R}^\exists$ and $\mathcal{EL}_\mathsf{lhs}$) and then the lower bounds. Proof details are provided in an appendix (Konev et al. 2013).

Related Work. In the terminology of the learning literature, entailment queries are a kind of membership queries, which can take many different forms (Raedt 1997). Learning using membership and equivalence queries appears to be the most successful protocol for exact learning in Angluin's framework. Apart from propositional Horn formulas, polynomial time learnable classes for this protocol include regular sets (Angluin 1987b) and monotone DNF (Angluin 1987c). Exploring the possibilities of extending the learning algorithm for propositional Horn formulas to (fragments) of first-order Horn logic has been a major topic in exact learning (Reddy and Tadepalli 1999; Arias and Khardon 2002; Arias, Khardon, and Maloberti 2007; Selman and Fern 2011). Note that $\mathcal{EL}_\mathsf{lhs}$ can be regarded as a fragment of first-order (FO) Horn logic. Existing approaches to polynomial time learning of FO Horn formulas either disallow recursion, bound the number of individual variables per Horn clause, or admit additional queries in the learning protocol. None of this is the case in our setup. Also related to our work is exact learning of schema mappings in data exchange, as recently studied in (ten Cate, Dalmau, and Kolaitis 2012). In this and some other studies mentioned above, membership queries take the form of interpretations ("is a given interpretation a model of the target theory?"). In DL, exact learning has been studied for CLASSIC in (Frazier and Pitt 1996) where it is shown that single CLASSIC concepts (but not TBoxes) can be learned in polynomial time (here membership queries are concepts). Learning single concepts using refinement operators has been studied in (Lehmann and Hitzler 2010).

Preliminaries

Let $\mathsf{N_C}$ be a countably infinite set of *concept names* and $\mathsf{N_R}$ a countably infinite set of *role names*. The dialect DL-Lite$_\mathcal{R}^\exists$ of DL-Lite is defined as follows (Calvanese et al. 2007). A *role* is a role name or an inverse role r^- with $r \in \mathsf{N_R}$. A *role inclusion (RI)* is of the form $r \sqsubseteq s$, where r and s are roles. A *basic concept* is either a concept name or of the form $\exists r.\top$, with r a role. A DL-Lite$_\mathcal{R}^\exists$ concept inclusion (CI) is of the form $B \sqsubseteq C$, where B is a basic concept and C is an $\mathcal{ELI}$ concept, that is, C is formed according to the rule

$$C, D \quad := \quad A \quad | \quad \top \quad | \quad C \sqcap D \quad | \quad \exists r.C \quad | \quad \exists r^-.C$$

where A ranges over $\mathsf{N_C}$ and r ranges over $\mathsf{N_R}$. A DL-Lite$_\mathcal{R}^\exists$ TBox is a finite set of DL-Lite$_\mathcal{R}^\exists$ CIs and RIs.[1]

As usual, an $\mathcal{EL}$ *concept* is an $\mathcal{ELI}$ concept that does not use inverse roles, an $\mathcal{EL}$ *concept inclusion* has the form $C \sqsubseteq D$ with C and D $\mathcal{EL}$ concepts, and a *(general) $\mathcal{EL}$ TBox* is a finite set of $\mathcal{EL}$ concept inclusions (Baader, Brandt, and Lutz 2005). We use $C \equiv D$ as an abbreviation for the inclusions $C \sqsubseteq D$ and $D \sqsubseteq C$. An $\mathcal{EL}$ TBox $\mathcal{T}$ is *acyclic* if it consists of inclusions $A \sqsubseteq C$ and

[1] For simplicity, we consider DL-Lite without disjointness axioms. All our results also hold for the extension of DL-Lite$_\mathcal{R}^\exists$ with disjointness.

$A \equiv C$ such that A is a concept name, no concept names occurs more than once on the left hand side of an inclusion in $\mathcal{T}$, and $\mathcal{T}$ contains no cycles (Baader et al. 2003; Konev et al. 2012). The semantics of concepts and TBoxes is defined as usual in terms of interpretations (Baader et al. 2003). For a TBox $\mathcal{T}$ and concept inclusion $C \sqsubseteq D$, we write $\mathcal{T} \models C \sqsubseteq D$ if every model $\mathcal{I}$ of $\mathcal{T}$ also satisfies $C \sqsubseteq D$. $\mathcal{T}$ is omitted if it is empty, that is, we then simply write $\models C \sqsubseteq D$. We say that concepts C and D are *equivalent w.r.t.* $\mathcal{T}$ and write $C \equiv_{\mathcal{T}} D$ if both $\mathcal{T} \models C \sqsubseteq D$ and $\mathcal{T} \models D \sqsubseteq C$; if r, s are roles, $r \equiv_{\mathcal{T}} s$ is defined analogously. A *signature* Σ is a finite set of concept and role names. The *size* of a concept C, denoted with $|C|$, is the length of the string that represents it, where concept names and role names are considered to be of length one. The *size* of a TBox $\mathcal{T}$, denoted with $|\mathcal{T}|$, is $\sum_{C \sqsubseteq D \in \mathcal{T}} |C| + |D|$.

Learning DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes in Polynomial Time

We prove that DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes can be learned in polynomial time. The signature Σ of the target TBox $\mathcal{T}$ is known to the learner. To simplify presentation, we assume that the target TBox is in *named form*, that is, it contains for each role r a concept name A_r such that $A_r \equiv \exists r.\top \in \mathcal{T}$. This assumption is without loss of generality since any (polynomial time) learning algorithm for TBoxes in named form can be transformed into one for unrestricted TBoxes: the algorithm still uses the concept names A_r in its internal representations (although they are no longer included in the target signature Σ), and replaces each A_r with $\exists r.\top$ in queries to the oracle and when ultimately returning the TBox that it has learned.

In an initial phase, the learning algorithm for DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes determines, using at most $\mathcal{O}(|\Sigma|^2)$ entailment queries, all CIs $B_1 \sqsubseteq B_2$ with $\mathcal{T} \models B_1 \sqsubseteq B_2$ and B_1, B_2 basic concepts and the set of all RIs $r \sqsubseteq s$ with $\mathcal{T} \models r \sqsubseteq s$. A concept or role inclusion is in *reduced form* if all basic concepts that occur in it are concept names. We assume without further notice that all concept inclusions considered by the learner, except those that have been determined in the initial phase, are in reduced form. In particular, counterexamples returned by the oracle are immediately converted into this form.

To formulate the learning algorithm, it is useful to identify each $\mathcal{ELI}$ concept C with a finite tree T_C whose nodes are labeled with sets of concept names and whose edges are labeled with roles. In detail, if A is a concept name, then T_A has a single node d with label $l(d) = \{A\}$; if $C = \exists r.D$, then T_C is obtained from T_D by adding a new root d_0 and an edge from d_0 to the root d of T_D with label $l(d_0, d) = r$ (we then call d an *r-successor* of d_0); if $C = D_1 \sqcap D_2$, then T_C is obtained by identifying the roots of T_{D_1} and T_{D_2}. Conversely, every tree T of the described form gives rise to an $\mathcal{ELI}$ concept C_T in the obvious way. We will not always distinguish explicitly between C and its tree representation T_C which allows us to speak, for example, about the nodes and subtrees of an $\mathcal{ELI}$ concept.

The following notion plays a central role in the learning algorithm. Let $\mathcal{T}$ be a DL-Lite$_{\mathcal{R}}^{\exists}$ TBox. A DL-Lite$_{\mathcal{R}}^{\exists}$ inclusion $A \sqsubseteq C$ is *$\mathcal{T}$-essential* if it is in reduced form, $\mathcal{T} \models A \sqsubseteq C$, and the following conditions are satisfied:

1. $A \sqsubseteq C$ is *concept saturated for* $\mathcal{T}$: if C' results from C by adding a concept name A' to the label of some node, then $\mathcal{T} \not\models A \sqsubseteq C'$.
2. $A \sqsubseteq C$ is *role saturated for* $\mathcal{T}$: if C' results from C by replacing a role r by a role r' with $r' \not\equiv_{\mathcal{T}} r$ and $\mathcal{T} \models r' \sqsubseteq r$, then $\mathcal{T} \not\models A \sqsubseteq C'$.
3. $A \sqsubseteq C$ is *sibling-merged for* $\mathcal{T}$: if C' is the result of identifying in C an r-successor and an s-successor of the same node where $r \equiv_{\mathcal{T}} s$, then $\mathcal{T} \not\models A \sqsubseteq C'$.[2]
4. $A \sqsubseteq C$ is *parent/child-merged for* $\mathcal{T}$: if d' is an r-successor of d in C and d'' is an s-successor of d' with $r^- \equiv_{\mathcal{T}} s$, then $\mathcal{T} \not\models A \sqsubseteq C'$ where C' results from C by identifying d and d''.
5. $A \sqsubseteq C$ is *minimal for* $\mathcal{T}$: if d' is an r-successor of d in C and A' is in the node label of d, then $\mathcal{T} \not\models A' \sqsubseteq \exists r.C'$ where C' corresponds to the subtree rooted at d' (where $A' \not\equiv_{\mathcal{T}} A$ if d is the root of C).

Note that Points 1 to 5 carefully mix conditions that 'maximize' (Points 1 and 2) and 'minimize' (Point 5) the concept C. Points 3 and 4 are a form of maximization.

Algorithm 1 The learning algorithm for DL-Lite$_{\mathcal{R}}^{\exists}$

```
1: Compute                                              (entailment queries)
     H_basic = {r ⊑ s | T ⊨ r ⊑ s} ∪
               {B1 ⊑ B2 | T ⊨ B1 ⊑ B2, B1, B2 basic}
2: Set H_add = ∅
3: while Equivalent (H_basic ∪ H_add)? returns "no" do
4:     Let A ⊑ C be the returned positive counterexample
           for T relative to H_basic ∪ H_add
5:     Find a T-essential inclusion A' ⊑ C' with
           H_basic ∪ H_add ⊭ A' ⊑ C'                    (entailment queries)
6:     if there is A' ⊑ C'' ∈ H_add then
7:         Find T-essential inclusion A' ⊑ C* such that
           H_basic ⊨ C* ⊑ C'' ⊓ C'                      (entailment queries)
8:         Replace A' ⊑ C'' by A' ⊑ C* in H_add
9:     else
10:        add A' ⊑ C' to H_add
11:    end if
12: end while
13: Set H = H_basic ∪ H_add.
```

The algorithm for learning DL-Lite$_{\mathcal{R}}^{\exists}$ TBoxes in named form is given as Algorithm 1, where the implementation of the lines marked with "(entailment queries)" requires the execution of entailment queries as detailed below. Observe that, in Line 4, the assumption that a positive counterexample is returned by the oracle (i.e., CI entailed by $\mathcal{T}$ but not by $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$) is justified by the construction of $\mathcal{H}_{basic}$ and $\mathcal{H}_{add}$, that is, we only include inclusions entailed by the target TBox $\mathcal{T}$ in our hypothesis TBox $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$ and thus, at all times, $\mathcal{T} \models \mathcal{H}_{basic} \cup \mathcal{H}_{add}$. We now show in

[2] The node that results from the identification is then either an r-successor or an s-successor of its parent, which is equivalent.

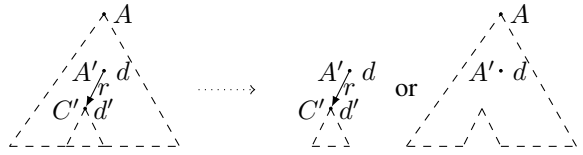


Figure 1: Minimization.

more detail how Lines 5 and 7 can be implemented, and that only polynomially many entailment queries are required. The next lemma addresses Line 5.

Lemma 1 *Given a positive counterexample $A \sqsubseteq C$ for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$, one can construct a $\mathcal{T}$-essential such counterexample using only polynomially many entailment queries in $|C| + |\mathcal{T}|$.*

Proof. Let $A \sqsubseteq C$ be a positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$. We may assume that $A \sqsubseteq C$ is in reduced form and exhaustively apply the following rules, which rely on posing entailment queries to the oracle:

(Concept saturation) if $\mathcal{T} \models A \sqsubseteq C'$ and C' results from C by adding a concept name A' to the label of some node, then replace $A \sqsubseteq C$ by $A \sqsubseteq C'$.

(Role saturation) if $\mathcal{T} \models A \sqsubseteq C'$ and C' results from C by replacing a role r by a r' with $r \not\equiv_{\mathcal{T}} r'$ and $\mathcal{T} \models r' \sqsubseteq r$, then replace $A \sqsubseteq C$ by $A \sqsubseteq C'$.

(Sibling merging) if $\mathcal{T} \models A \sqsubseteq C'$ and C' is the result of identifying in C an r-successor and an s-successor of the same node where $r \equiv_{\mathcal{T}} s$, then replace $A \sqsubseteq C$ by $A \sqsubseteq C'$.

(Parent/child merging) if d' is an r-successor of d in C, d'' is an s-successor of d' with $r^- \equiv_{\mathcal{T}} s$, and $\mathcal{T} \models A \sqsubseteq C'$ where C' results from C by identifying d and d'', then replace $A \sqsubseteq C$ by $A \sqsubseteq C'$.

(Minimization) if d' is an r-successor of d in C, A' is in the node label of d, and C' corresponds to the subtree rooted at d', $\mathcal{T} \models A' \sqsubseteq \exists r.C'$ (plus $A' \not\equiv_{\mathcal{T}} A$ if d is the root of C), then replace $A \sqsubseteq C$ by

(a) $A' \sqsubseteq \exists r.C'$ if $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\models A' \sqsubseteq \exists r.C'$; or

(b) $A \sqsubseteq C|^-_{d'\downarrow}$, where $C|^-_{d'\downarrow}$ is obtained from C by removing the subtree generated by d' from C, otherwise.

The last rule is illustrated in Figure 1. It follows directly from the definition of the rules that the final concept inclusion is $\mathcal{T}$-essential. Let us verify that it is also a positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$. It suffices to show that the CI resulting from each rule application is entailed by $\mathcal{T}$, but not by $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$. The former is straightforward for all five rules. Regarding the latter, in the first four rules we have $\mathcal{H}_{basic} \models C' \sqsubseteq C$ if $A \sqsubseteq C$ is replaced by $A \sqsubseteq C'$. Hence $\mathcal{H}_{basic} \cup \mathcal{H}_{add} \not\models A \sqsubseteq C'$. In the last rule, we have $\{A \sqsubseteq C|^-_{d'\downarrow}, A' \sqsubseteq \exists r.C'\} \models A \sqsubseteq C$. Thus, one of the inclusions $A \sqsubseteq C|^-_{d'\downarrow}$ and $A' \sqsubseteq \exists r.C'$ is not entailed by $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$, and this is the CI resulting from the rule application.

It is not hard to see that the number of rule applications is bounded by $|C|^2 \cdot |\Sigma|$. Note that termination of the algorithm requires the condition in the last rule that $A' \not\equiv_{\mathcal{T}} A$ if d is the root of C. Otherwise, this rule could 'replace' $A \sqsubseteq C$ by any $A' \sqsubseteq C$ with A equivalent to A' w.r.t. $\mathcal{T}$. With the mentioned condition and since all CIs are in reduced form, each time when d is the root of C and $A \sqsubseteq C$ is replaced by $A' \sqsubseteq \exists r.C'$, then we strictly descend in subsumption order, that is, $\mathcal{T} \models A \sqsubseteq A'$ and $\mathcal{T} \not\models A' \sqsubseteq A$. ❑

The following lemma addresses Line 7.

Lemma 2 *Assume that $A \sqsubseteq C_1$ and $A \sqsubseteq C_2$ are $\mathcal{T}$-essential. Then one can construct a $\mathcal{T}$-essential $A \sqsubseteq C$ such that $\mathcal{H}_{basic} \models C \sqsubseteq C_1 \sqcap C_2$ using polynomially many entailment queries in $|C_1| + |C_2|$.*

Proof. Starting with $A \sqsubseteq C_1 \sqcap C_2$, we exhaustively apply the rule for sibling merging from Lemma 1 and denote the result by $A \sqsubseteq C$. Observe that since $A \sqsubseteq C_1$ and $A \sqsubseteq C_2$ are both $\mathcal{T}$-essential, the concept inclusion $A \sqsubseteq C_1 \sqcap C_2$ is (i) concept-saturated for $\mathcal{T}$, (ii) role saturated for $\mathcal{T}$, (iii) parent/child-merged for $\mathcal{T}$, and (iv) minimal for $\mathcal{T}$. The only property of $\mathcal{T}$-essential inclusions that can fail for $A \sqsubseteq C_1 \sqcap C_2$ is being sibling-merged for $\mathcal{T}$. Now one can show that an inclusion with properties (i)–(iv) still has those properties after applying the rule (sibling merging) above. Thus $A \sqsubseteq C$ is $\mathcal{T}$-essential, as required. ❑

Example 3 *Suppose the target TBox $\mathcal{T}$ is*[3]

$$\begin{aligned} \mathsf{Prof} &\sqsubseteq \mathsf{Academic} \sqcap \\ &\quad \exists \mathsf{worksFor}.(\exists \mathsf{supports.Teaching} \sqcap \\ &\qquad \exists \mathsf{supports.Research}) \\ \mathsf{Academic} &\sqsubseteq \exists \mathsf{advisor.Academic} \end{aligned}$$

The algorithm first computes $\mathcal{H}_{\mathsf{basic}}$ as

$$\begin{aligned} \mathsf{Prof} &\sqsubseteq \mathsf{Academic} & \mathsf{Prof} &\sqsubseteq \exists \mathsf{worksFor}.\top \\ \mathsf{Prof} &\sqsubseteq \exists \mathsf{advisor}.\top & \mathsf{Academic} &\sqsubseteq \exists \mathsf{advisor}.\top \end{aligned}$$

and then poses $\mathcal{H}_{\mathsf{basic}}$ as the first equivalence query. Assume that the oracle returns the positive counterexample

$$\mathsf{Prof} \sqsubseteq \exists \mathsf{advisor.Academic}.$$

which is not concept saturated for $\mathcal{T}$; saturating it yields

$$\mathsf{Prof} \sqsubseteq \mathsf{Academic} \sqcap \exists \mathsf{advisor.Academic}$$

which is not minimal for $\mathcal{T}$; minimizing it yields

$$\mathsf{Academic} \sqsubseteq \exists \mathsf{advisor.Academic}.$$

The above CI constitutes the new $\mathcal{H}_{\mathsf{add}}$ and the algorithm asks $\mathcal{H}_{\mathsf{basic}} \cup \mathcal{H}_{\mathsf{add}}$ as an equivalence query. Assume the oracle returns the positive counterexample

$$\begin{aligned} \mathsf{Prof} &\sqsubseteq \exists \mathsf{worksFor}.(\exists \mathsf{supports.Teaching} \sqcap \\ &\qquad \exists \mathsf{worksFor}^-.\mathsf{Academic}) \end{aligned}$$

[3]To facilitate presentation, we use a TBox $\mathcal{T}$ that is not in named form; for this concrete example, however, named form does not make any difference.

which is not parent/child-merged for $\mathcal{T}$; merging it results in the following CI, which is added to $\mathcal{H}_{\mathsf{add}}$:

$$\mathsf{Prof} \sqsubseteq \mathsf{Academic} \sqcap \exists \mathsf{worksFor}.\exists \mathsf{supports}.\mathsf{Teaching}.$$

For the next equivalence query, the oracle returns the positive counterexample

$$\mathsf{Prof} \sqsubseteq \exists \mathsf{worksFor}.\exists \mathsf{supports}.\mathsf{Research}.$$

Following Lines 6 and 7 of Algorithm 1, we conjoin the right-hand side of this CI with that of the existing CI for Prof *in $\mathcal{H}_{\mathsf{add}}$. Applying sibling merging yields the first CI from $\mathcal{T}$, and the algorithm has succeeded to learn $\mathcal{T}$.*

If Algorithm 1 terminates, then it obviously has found a TBox $\mathcal{H}_{basic} \cup \mathcal{H}_{add}$ that is logically equivalent to $\mathcal{T}$. It thus remains to show that the algorithm terminates in polynomially time. Observe that $\mathcal{H}_{add}$ contains at most one concept inclusion $A \sqsubseteq C$ for each concept name A. At each step in the while loop, either some $A \sqsubseteq C$ is added to $\mathcal{H}_{add}$ such that no inclusion with A on the left-hand side existed in $\mathcal{H}_{add}$ before or an existing inclusion $A \sqsubseteq C$ in $\mathcal{H}_{add}$ is replaced by a fresh $A \sqsubseteq C'$ with $\models C' \sqsubseteq C$ and $\not\models C \sqsubseteq C'$. It is thus not hard to see that we can prove termination in polynomial time by establishing the following lemma.

Lemma 4 *The number of replacements of an existing CI $A \sqsubseteq C$ in $\mathcal{H}_{add}$ is bounded polynomially in $|\mathcal{T}|$.*

Proof.(sketch) We first observe that, as a consequence of the fact that $A \sqsubseteq C$ and its replacement $A \sqsubseteq C'$ are both $\mathcal{T}$-essential, the tree that corresponds to C is obtained from the tree that corresponds to C' by removing subtrees (and possibly replacing roles). Let n_C denote the number of nodes in the tree representation of C and let $A^{\mathcal{T}} = \{B \mid A \sqsubseteq B \in \mathcal{H}_{basic}\} \cup \{D \mid A \equiv_{\mathcal{T}} B, B \sqsubseteq D \in \mathcal{T}\}$. Then it suffices to prove the following

Claim. If $A \sqsubseteq C$ is $\mathcal{T}$-essential, then $n_C \leq \sum_{D \in A^{\mathcal{T}}} n_D$.

The claim is proved in the appendix using canonical models for DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes. ❑

We have obtained the following main result of this section.

Theorem 5 *DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes are polynomial time learnable using entailment and equivalence queries.*

In the appendix, we provide additional examples showing that if any of the five conditions for being $\mathcal{T}$-essential is removed, then Algorithm 1 no longer runs in polynomial time or does not even terminate. As an example, assume that minimality is omitted from the definition of being $\mathcal{T}$-essential and let $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq \exists r.B\}$. Then the oracle can provide for the n-th equivalence query the positive counterexample $A \sqsubseteq \exists r^n.B$, $n \geq 1$. The algorithm does not terminate.

Learning $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes in Polynomial Time

We consider the restriction $\mathcal{EL}_{\mathsf{lhs}}$ of general $\mathcal{EL}$ TBoxes where only concept names are allowed on the right-hand side of concept inclusions. We assume that CIs used in entailment queries and in equivalence queries and those returned as counterexamples are also of this restricted form. Our learning algorithm is a non-trivial extension of the polynomial time algorithm for learning propositional Horn theories presented in (Angluin, Frazier, and Pitt 1992; Arias and Balcázar 2011).

We first introduce some notation. An interpretation $\mathcal{I}$ is a *tree interpretation* if the directed graph $(\Delta^{\mathcal{I}}, \bigcup_{r \in \mathsf{N_R}} r^{\mathcal{I}})$ is a tree and $r^{\mathcal{I}} \cap s^{\mathcal{I}} = \emptyset$ for all distinct $r, s \in \mathsf{N_R}$. We generally denote the root of a tree interpretation $\mathcal{I}$ with $\rho_{\mathcal{I}}$. The *product* of two interpretations $\mathcal{I}$ and $\mathcal{J}$ is the interpretation $\mathcal{I} \times \mathcal{J}$ with $\Delta^{\mathcal{I}\times\mathcal{J}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$, $(d, e) \in A^{\mathcal{I}\times\mathcal{J}}$ if $d \in A^{\mathcal{I}}$ and $e \in A^{\mathcal{J}}$, and $((d, e), (d', e')) \in r^{\mathcal{I}\times\mathcal{J}}$ if $(d, d') \in r^{\mathcal{I}}$ and $(e, e') \in r^{\mathcal{J}}$, for any concept name A and role name r. Products preserve the truth of $\mathcal{EL}$ concept inclusions (Lutz, Piro, and Wolter 2011):

Lemma 6 *For all $\mathcal{EL}$ concepts C: $d \in C^{\mathcal{I}}$ and $e \in C^{\mathcal{J}}$ iff $(d, e) \in C^{\mathcal{I}\times\mathcal{J}}$.*

One can show that the product of tree interpretations is a disjoint union of tree interpretations. If $\mathcal{I}$ and $\mathcal{J}$ are tree interpretations, we denote by $\mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ the tree interpretation that is contained in $\mathcal{I} \times \mathcal{J}$ with root $(\rho_{\mathcal{I}}, \rho_{\mathcal{J}})$.

Let $\mathcal{I}, \mathcal{J}$ be interpretations, $d \in \Delta^{\mathcal{I}}$ and $e \in \Delta^{\mathcal{J}}$. A relation $\sim \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ is a *simulation from $(\mathcal{I}, d)$ to $(\mathcal{J}, e)$* if the following conditions are satisfied: (i) $d \sim e$; (ii) $d \in A^{\mathcal{I}}$ and $d \sim e$ implies $e \in A^{\mathcal{J}}$; (iii) $(d, d') \in r^{\mathcal{I}}$ and $d \sim e$ implies $d' \sim e'$ for some $e' \in \Delta^{\mathcal{J}}$ with $(e, e') \in r^{\mathcal{J}}$. We write $(\mathcal{I}, d) \Rightarrow (\mathcal{J}, e)$ if there is a simulation from $(\mathcal{I}, d)$ to $(\mathcal{J}, e)$ and if $\mathcal{I}$ and $\mathcal{J}$ are tree interpretations then we write $\mathcal{I} \Rightarrow \mathcal{J}$ as a shorthand for $(\mathcal{I}, \rho_{\mathcal{I}}) \Rightarrow (\mathcal{J}, \rho_{\mathcal{J}})$. Simulations preserve the membership of $\mathcal{EL}$-concepts (Lutz, Piro, and Wolter 2011):

Lemma 7 *For all $\mathcal{EL}$ concepts C: if $d \in C^{\mathcal{I}}$ and $(\mathcal{I}, d) \Rightarrow (\mathcal{J}, e)$, then $e \in C^{\mathcal{J}}$.*

For an $\mathcal{EL}$ concept C, we write $\mathcal{I}_C$ to denote the tree interpretation that is obtained by viewing C as an interpretation in the natural way, that is, $\mathcal{I}_C$ is the tree T_C introduced in the context of our learning algorithm for DL-Lite$^{\exists}_{\mathcal{R}}$, represented as an interpretation. Conversely, every tree interpretation $\mathcal{I}$ can be viewed as an $\mathcal{EL}$ concept $C_{\mathcal{I}}$ in a straightforward way.

Let $\mathcal{T}$ be the target TBox to be learned, and assume again that its signature Σ is known to the learner. An interpretation $\mathcal{I}$ is a *$\mathcal{T}$-countermodel* if $\mathcal{I} \not\models \mathcal{T}$. We will now describe a class of $\mathcal{T}$-countermodels that are in a sense minimal and central to our learning algorithm. For a tree interpretation $\mathcal{I}$, we use $\mathcal{I}|^{-}_{\mathsf{root}}$ to denote the interpretation obtained from $\mathcal{I}$ by removing the root $\rho_{\mathcal{I}}$ of $\mathcal{I}$. For any element d of $\Delta^{\mathcal{I}}$, we use $\mathcal{I}|^{-}_{d\downarrow}$ to denote $\mathcal{I}$ with the subtree rooted at d removed. A $\mathcal{T}$-countermodel is *essential* if the following conditions are satisfied:

Algorithm 2 The learning algorithm for $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes

1: Set $\mathfrak{I}$ to the empty sequence (of tree interpretations)
2: Set $\mathcal{H} = \emptyset$
3: **while** Equivalent($\mathcal{H}$)? returns "no" **do**
4: Let $C \sqsubseteq A$ be the returned positive counterexample for $\mathcal{T}$ relative to $\mathcal{H}$
5: Find an essential $\mathcal{T}$-countermodel $\mathcal{I}$ with $\mathcal{I} \models \mathcal{H}$ (entailment queries)
6: **if** there is a $\mathcal{J} \in \mathfrak{I}$ such that $\mathcal{J} \not\to (\mathcal{I} \times_{\mathsf{r}} \mathcal{J})$ and $\mathcal{I} \times_{\mathsf{r}} \mathcal{J} \not\models \mathcal{T}$ **then**
7: Let $\mathcal{J}$ be the first such element of $\mathfrak{I}$
8: Find an essential $\mathcal{T}$-countermodel $\mathcal{J}' \subseteq \mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ (entailment queries)
9: replace $\mathcal{J}$ in $\mathfrak{I}$ with $\mathcal{J}'$
10: **else**
11: append $\mathcal{I}$ to $\mathfrak{I}$
12: **end if**
13: Construct $\mathcal{H} = \{C_{\mathcal{I}} \sqsubseteq A \mid \mathcal{I} \in \mathfrak{I}, \mathcal{T} \models C_{\mathcal{I}} \sqsubseteq A\}$ (entailment queries)
14: **end while**

1. $\mathcal{I}|^{-}_{\mathsf{root}} \models \mathcal{T}$;
2. $\mathcal{I}|^{-}_{d\downarrow} \models \mathcal{T}$ for all $d \in \Delta^{\mathcal{I}} \setminus \{\rho_{\mathcal{I}}\}$.

Intuitively, Condition 1 states that $\mathcal{I}$ contradicts $\mathcal{T}$ only at the root, that is, the only reason for why $\mathcal{I}$ does not satisfy $\mathcal{T}$ is that for at least one CI $C \sqsubseteq A \in \mathcal{T}$, we have that $\rho_{\mathcal{I}} \in C^{\mathcal{I}}$ and $\rho_{\mathcal{I}} \notin A^{\mathcal{I}}$. Condition 2 is a minimality condition which states that for any such $C \sqsubseteq A$ (which needs not be unique), $\rho_{\mathcal{I}}$ is no longer in $C^{\mathcal{I}}$ if we remove any node from $\mathcal{I}$.

The algorithm for learning $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes is given as Algorithm 2, where the implementation of the lines marked with "(entailment queries)" requires the execution of entailment queries as detailed below. It maintains a sequence $\mathfrak{I}$ of tree interpretations that intuitively represents the TBox $\mathcal{H}$ constructed in Line 13; however, tree interpretations are easier to work with for our purposes than the $\mathcal{EL}$ concepts that they represent. In Line 8, we write $\mathcal{J}' \subseteq \mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ as shorthand for: $\mathcal{J}'$ is a subinterpretation of $\mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ that is obtained from $\mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ by removing subtrees. Note that the assumption in Line 4 that a *positive* counterexample is returned is justified by the construction of $\mathcal{H}$ in Lines 2 and 13, which ensures that, at all times, $\mathcal{T} \models \mathcal{H}$.

We now provide additional details on how to realize the three lines marked with "(entailment queries)". Line 13 is easiest: We simply use entailment queries to find all CIs $C_{\mathcal{I}} \sqsubseteq A$ with $\mathcal{I} \in \mathfrak{I}$ and A a concept name from Σ. We will later show that the length of $\mathfrak{I}$ is bounded polynomially in $|\mathcal{T}|$, therefore polynomially many entailment queries suffice. Lines 5 and 8 are addressed by Lemmas 8 and 9 below.

Lemma 8 *Given a positive counterexample $C \sqsubseteq A$ for $\mathcal{T}$ relative to $\mathcal{H}$, one can construct an essential $\mathcal{T}$-countermodel $\mathcal{I}$ with $\mathcal{I} \models \mathcal{H}$ using only polynomially many entailment queries in $|\mathcal{T}| + |C|$.*

Proof. $\mathcal{I}$ is constructed by applying the following rules to $\mathcal{I} := \mathcal{I}_C$.

1. Saturate $\mathcal{I}$ by exhaustively applying the CIs from $\mathcal{H}$ as rules: if $D \sqsubseteq B \in \mathcal{H}$ and $d \in D^{\mathcal{I}}$, then add d to $B^{\mathcal{I}}$.
2. Replace $\mathcal{I}$ by a minimal subtree of $\mathcal{I}$ refuting $\mathcal{T}$ to address Condition 1 of essential $\mathcal{T}$-countermodels: replace $\mathcal{I}$ by $\mathcal{I}|_d$ if $\mathcal{I}|_d$ is minimal with $\mathcal{I}|_d \not\models \mathcal{T}$ (checked using entailment queries), where $\mathcal{I}|_d$ denotes the subtree of $\mathcal{I}$ rooted at d.
3. Exhaustively remove subtrees from $\mathcal{I}$ until Condition 2 of essential $\mathcal{T}$-countermodels is also satisfied: if $\mathcal{I}|^{-}_{d\downarrow} \not\models \mathcal{T}$ (checked using entailment queries), then replace $\mathcal{I}$ by $\mathcal{I}|^{-}_{d\downarrow}$.

The resulting interpretation $\mathcal{I}$ is as required. Details are presented in the appendix. ❏

Lemma 9 *Given essential $\mathcal{T}$-countermodels $\mathcal{I}$ and $\mathcal{J}$ with $\mathcal{I} \times \mathcal{J} \not\models \mathcal{T}$, one can construct an essential $\mathcal{T}$-countermodel $\mathcal{J}' \subseteq \mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ using only polynomially many entailment queries in $|\mathcal{T}| + |\mathcal{I}| + |\mathcal{J}|$.*

Proof. Let $\mathcal{I}$ and $\mathcal{J}$ be essential $\mathcal{T}$-countermodels with $\mathcal{I} \times_{\mathsf{r}} \mathcal{J} \not\models \mathcal{T}$. Set $\mathcal{J}' = \mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ and then exhaustively apply Rule 3 from the proof of Lemma 8 to $\mathcal{J}'$. Clearly, the new $\mathcal{J}'$ is a $\mathcal{T}$-countermodel. Moreover, it is essential:

(1) $\mathcal{J}'|^{-}_{\mathsf{root}} \models \mathcal{T}$: by Lemma 6, $\mathcal{I}|^{-}_{\mathsf{root}} \models \mathcal{T}$ and $\mathcal{J}|^{-}_{\mathsf{root}} \models \mathcal{T}$ imply $\mathcal{I}|^{-}_{\mathsf{root}} \times \mathcal{J}|^{-}_{\mathsf{root}} \models \mathcal{T}$. Now $\mathcal{J}'|^{-}_{\mathsf{root}}$ can be obtained from $\mathcal{I}|^{-}_{\mathsf{root}} \times \mathcal{J}|^{-}_{\mathsf{root}}$ by removing subtrees, and removing subtrees clearly preserves being a model of an $\mathcal{EL}_{\mathsf{lhs}}$ TBox.

(2) $\mathcal{J}'|^{-}_{d\downarrow} \models \mathcal{T}$ for all $d \in \Delta^{\mathcal{J}'} \setminus \{\rho_{\mathcal{J}'}\}$: otherwise, the subtree rooted at d would have been removed during the construction of $\mathcal{J}'$. ❏

Example 10 *Suppose that the target TBox is*

$$\exists \mathsf{parent}.\mathsf{Mouse} \sqsubseteq \mathsf{Mouse}$$
$$\exists \mathsf{parent}.\mathsf{Rabbit} \sqsubseteq \mathsf{Rabbit}$$
$$\exists \mathsf{parent}.\top \sqsubseteq \mathsf{Living}$$

and assume that the oracle returns to the first equivalence query the positive counterexample $C \sqsubseteq \mathsf{Living}$ with

$$C = \exists \mathsf{parent}.\mathsf{Mouse} \sqcap \exists \mathsf{parent}.\mathsf{Rabbit}$$

$\mathcal{I}_C$ is a $\mathcal{T}$-countermodel that violates Condition 2 of being essential which leads to its rewriting into $\mathcal{I}_{C'}$ with

$$C' = \exists \mathsf{parent}.\mathsf{Mouse}$$

(Alternatively, it could also lead to $C' = \exists \mathsf{parent}.\mathsf{Rabbit}$.) The algorithm inserts $\mathcal{I}_{C'}$ into $\mathfrak{I}$ and asks the next equivalence query, to which the oracle returns $D \sqsubseteq \mathsf{Living}$ with

$$D = \exists \mathsf{parent}.\mathsf{Rabbit}.$$

$\mathcal{I}_D$ is an essential $\mathcal{T}$-countermodel and we have $\mathcal{I}_{C'} \not\to (\mathcal{I}_D \times_{\mathsf{r}} \mathcal{I}_{C'})$, where $\mathcal{I}_D \times_{\mathsf{r}} \mathcal{I}_{C'}$ corresponds to the concept

$$D' = \exists \mathsf{parent}.\top.$$

$\mathcal{I}_{D'}$ is $\mathcal{T}$-essential and thus it is inserted into $\mathfrak{I}$ in place of $\mathcal{I}_{C'}$. At this point, the algorithm has learnt $\mathcal{H} = \{\exists \mathsf{parent}.\top \sqsubseteq \mathsf{Living}\}$. The algorithm asks the next equivalence query, to which the oracle returns $E \sqsubseteq \mathsf{Rabbit}$ with

$$E = \exists \mathsf{parent}.\exists \mathsf{parent}.\mathsf{Rabbit}$$

$\mathcal{I}_E$ violates Conditions 1 and 2 of being essential, which leads to its rewriting into $\mathcal{I}_{E'}$ with

$$E' = \mathsf{Living} \sqcap \exists \mathsf{parent}.\mathsf{Rabbit}.$$

We have $\mathcal{I}_{D'} \Rightarrow \mathcal{I}_{E'} \times_{\mathsf{r}} \mathcal{I}_{D'}$ and so $\mathcal{I}_{E'}$ is appended to the list $\mathfrak{I}$. At this point, the algorithm has learnt $\mathcal{H} = \{\mathsf{Living} \sqcap \exists \mathsf{parent}.\mathsf{Rabbit} \sqsubseteq \mathsf{Rabbit}, \exists \mathsf{parent}.\top \sqsubseteq \mathsf{Living}\}$, which is equivalent to the TBox that consists of the second and third CI in $\mathcal{T}$. The algorithm can now proceed to learn $\mathsf{Living} \sqcap \exists \mathsf{parent}.\mathsf{Mouse} \sqsubseteq \mathsf{Mouse}$.

If the algorithm terminates, then it obviously returns a TBox $\mathcal{H}$ that is equivalent to the target TBox $\mathcal{T}$. It thus remains to prove that the algorithm terminates after polynomially many steps, which is a consequence of the following lemma.

Lemma 11 *Let $\mathfrak{I}$ be a sequence computed at some point of an execution of Algorithm 2. Then (i) the length of $\mathfrak{I}$ is bounded by the number of CIs in $\mathcal{T}$ and (ii) each interpretation in each position of $\mathfrak{I}$ is replaced only $|\mathcal{T}| + |\mathcal{T}|^2$ often with a new interpretation.*

Proof.(sketch) Assume that at each point of the execution of the algorithm, $\mathfrak{I}$ has the form $\mathcal{I}_0, \dots, \mathcal{I}_k$ for some $k \geq 0$. To establish Point (i), we closely follow (Angluin, Frazier, and Pitt 1992) and show that (iii) for every $\mathcal{I}_i$, there is a $D_i \sqsubseteq A_i \in \mathcal{T}$ with $\mathcal{I}_i \not\models D_i \sqsubseteq A_i$ and (iv) if $i \neq j$, then $D_i \sqsubseteq A_i$ and $D_j \sqsubseteq A_j$ are not identical.

The proof of Points (iii) and (iv), in turn requires a careful analysis of the concept inclusions in the target TBox that are violated at the root of the $\mathcal{T}$-countermodels $\mathcal{I}_0, \dots, \mathcal{I}_k$.

For the proof of Point (ii), we show that $|\Delta^{\mathcal{I}}| \leq |\mathcal{T}|$ for any essential $\mathcal{T}$-countermodel $\mathcal{I}$ and analyze the relationship between the essential $\mathcal{T}$-countermodels produced from $\mathcal{I} \times_{\mathsf{r}} \mathcal{J}$ when $\mathcal{J} \not\Rightarrow (\mathcal{I} \times_{\mathsf{r}} \mathcal{J})$ (using Lemma 6 and 7). ❑

We have thus established the main result of this section.

Theorem 12 *$\mathcal{EL}_{\mathsf{lhs}}$ TBoxes are polynomial time learnable using entailment and equivalence queries.*

It can again be shown that all adopted conditions are indeed necessary. Consider for example the first condition of being $\mathcal{T}$-essential. Without it, for the target TBox $\mathcal{T} = \{\exists r.A \sqsubseteq A\}$ the oracle can return the infinite set of positive counterexamples $\exists r^n.A \sqsubseteq A$, n a prime number. Then the learning algorithm does not terminate.

Limits of Polynomial Time Learnability

The main result of this section is that $\mathcal{EL}$ TBoxes cannot be learned in polynomial time using entailment and equivalence queries. We also show that DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes cannot be learned in polynomial time using entailment or equivalence queries alone. This holds for $\mathcal{EL}_{\mathsf{lhs}}$ TBoxes as well and follows from the fact that propositional Horn theories cannot be learned in polynomial time using entailment or equivalence queries alone (Frazier and Pitt 1993; Angluin, Frazier, and Pitt 1992; Angluin 1987a).

We start by proving the non-learnability result for $\mathcal{EL}$ TBoxes. Our proof shows that even *acyclic* target TBoxes cannot be learned in polynomial time when general $\mathcal{EL}$ concept inclusions are admitted in entailment queries, as counterexamples returned by the oracle, and in TBoxes used as equivalence queries. On our way, we also prove non-learnability of DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes using entailment queries only. The proof is inspired by Angluin's lower bound for the following abstract learning problem (1987c): a learner aims to identify one of N distinct sets $L_1, \dots, L_N$ which have the property that there exists a set $L_\cap$ for which $L_i \cap L_j = L_\cap$, for any $i \neq j$. It is assumed that $L_\cap$ is not a valid argument to an equivalence query. The learner can pose membership queries "$x \in L$?" and equivalence queries "$H = L$?". Then in the worst case it takes at least $N - 1$ membership and equivalence queries to exactly identify a hypothesis L_i from $L_1, \dots, L_N$. The proof proceeds as follows. At every stage of computation, the oracle (which here should be viewed as an adversary) maintains a set of hypotheses S, which the learner is not able to distinguish based on the answers given so far. Initially, $S = \{L_1, \dots, L_N\}$. When the learner asks a membership query x, the oracle returns 'Yes' if $x \in L_\cap$ and 'No' otherwise. In the latter case, the (unique) L_i such that $x \in L_i$ is removed from S. When the learner asks an equivalence query H, the oracle returns 'No' and a counterexample $x \in L_\cap \oplus H$ (the symmetric difference of $L_\cap$ and H). This always exists as $L_\cap$ is not a valid query. If the counterexample x is not a member of $L_\cap$, (at most one) $L_i \in S$ such that $x \in L_i$ is eliminated from S. In the worst case, the learner has to reduce the cardinality of S to one to exactly identify a hypothesis, which takes $N - 1$ queries.

Similarly to the method outlined above, in our proof we maintain a set of acyclic $\mathcal{EL}$ TBoxes S whose members the learning algorithm is not able to distinguish based on the answers obtained so far. We start with $S = \{\mathcal{T}_1, \dots, \mathcal{T}_N\}$, where N is superpolynomial in the size of every TBox $\mathcal{T}_i$, and describe an oracle that responds to entailment and equivalence queries. For didactic purposes, we first present a set of acyclic TBoxes $\mathcal{T}_1, \dots, \mathcal{T}_N$, for which the oracle can respond to entailment queries in the way described above but which is polynomial time learnable when equivalence queries are also allowed. We then show how the TBoxes can be modified to obtain a family of acyclic TBoxes that is not polynomial time learnable using entailment and equivalence queries.

To present the TBoxes in S, fix two role names r and s. For any sequence $\sigma = \sigma^1\sigma^2 \dots \sigma^n$ with $\sigma^i \in \{r, s\}$, the expression $\exists \boldsymbol{\sigma}.C$ stands for $\exists \sigma^1.\exists \sigma^2 \dots \exists \sigma^n.C$. For every such sequence $\boldsymbol{\sigma}$, of which there are $N = 2^n$ many, consider the acyclic $\mathcal{EL}$ TBox $\mathcal{T}_\sigma$ defined as

$$\begin{aligned} \mathcal{T}_{\boldsymbol{\sigma}} &= \{A \sqsubseteq \exists \boldsymbol{\sigma}.M \sqcap X_0\} \cup \mathcal{T}_0 \text{ with} \\ \mathcal{T}_0 &= \{X_i \sqsubseteq \exists r.X_{i+1} \sqcap \exists s.X_{i+1} \mid 0 \leq i < n\} \end{aligned}$$

where $\mathcal{T}_0$ generates a full binary tree whose edges are la-

belled with the role names r and s and with X_0 at the root, X_1 at level 1 and so on. M is a concept name that 'marks' a particular path in this tree given by the sequence $\boldsymbol{\sigma}$. One can use Angluin's strategy to show that TBoxes from the set S of all such TBoxes $\mathcal{T}_{\boldsymbol{\sigma}}$ cannot be learned in polynomial time using entailment queries only: notice that for no sequence $\boldsymbol{\sigma}' \neq \boldsymbol{\sigma}$ of length n, we have $\mathcal{T}_{\boldsymbol{\sigma}} \models A \sqsubseteq \exists \boldsymbol{\sigma}'.M$. Thus an entailment query of the form $A \sqsubseteq \exists \boldsymbol{\sigma}.M$ eliminates at most one TBox from the set of TBoxes that the learner cannot distinguish. This observation can be generalized to arbitrary entailment queries $C \sqsubseteq D$ in $\mathcal{EL}$ since one can prove, similarly to the proof of Lemma 14 below, that for any $\mathcal{EL}$ concept inclusion $C \sqsubseteq D$ either $\{A \sqsubseteq X_0\} \cup \mathcal{T}_0 \models C \sqsubseteq D$ (thus $C \sqsubseteq D$ is entailed by all TBoxes in S) or at most one TBox from S entails $C \sqsubseteq D$. It follows that acyclic $\mathcal{EL}$ TBoxes are not polynomial time learnable using entailment queries, only. Observe that the TBoxes $\mathcal{T}_{\boldsymbol{\sigma}}$ are actually formulated in DL-Lite$^{\exists}_{\mathcal{R}}$. We show in the appendix that all arguments above are true also when entailment queries are formulated in DL-Lite$^{\exists}_{\mathcal{R}}$ (which include inverse roles) and thus obtain the following result.

Theorem 13 *DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes are not polynomial time learnable using entailment queries, only.*

Interestingly, a single equivalence query is sufficient to learn any TBox from S in two steps: given the equivalence query $\{A \sqsubseteq X_0\} \cup \mathcal{T}_0$, the oracle has no other option but to reveal the target TBox $\mathcal{T}_{\boldsymbol{\sigma}}$ as $A \sqsubseteq \exists \boldsymbol{\sigma}.M$ can be found 'inside' every counterexample. Our strategy to rule out this option for the oracle is to modify $\mathcal{T}_1, \dots, \mathcal{T}_N$ in such a way that although a TBox $\mathcal{T}_\cap$ axiomatizing the intersection over the set of consequences of each $\mathcal{T}_i$, $i \leq N$, exists, its size is superpolynomial and so cannot be used as an equivalence query by a polynomial time learning algorithm.

For every $n > 0$ and every n-tuple $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$, where every $\boldsymbol{\sigma}_i$ is a role sequence of length n as above, we define an acyclic $\mathcal{EL}$ TBox $\mathcal{T}_L$ as the union of $\mathcal{T}_0$ and the following inclusions:[4]

$$\begin{array}{ccc} A_1 \sqsubseteq \exists \boldsymbol{\sigma}_1.M \sqcap X_0 & & A_n \sqsubseteq \exists \boldsymbol{\sigma}_n.M \sqcap X_0 \\ B_1 \sqsubseteq \exists \boldsymbol{\sigma}_1.M \sqcap X_0 & \cdots & B_n \sqsubseteq \exists \boldsymbol{\sigma}_n.M \sqcap X_0 \end{array}$$

$$A \equiv X_0 \sqcap \exists \boldsymbol{\sigma}_1.M \sqcap \dots \sqcap \exists \boldsymbol{\sigma}_n.M.$$

Let $\mathfrak{L}_n$ be a set of n-tuples such that for $1 \leq i \leq n$ and every $L, L' \in \mathfrak{L}_n$ with $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$, $L' = (\boldsymbol{\sigma}'_1, \dots, \boldsymbol{\sigma}'_n)$, if $\boldsymbol{\sigma}_i = \boldsymbol{\sigma}'_j$ then $L = L'$ and $i = j$. Then for any sequence $\boldsymbol{\sigma}$ of length n there exists at most one $L \in \mathfrak{L}_n$ and at most one $i \leq n$ such that $\mathcal{T}_L \models A_i \sqsubseteq \exists \boldsymbol{\sigma}.M$ and $\mathcal{T}_L \models B_i \sqsubseteq \exists \boldsymbol{\sigma}.M$. We can choose $\mathfrak{L}_n$ such that there are $N = \lfloor 2^n/n \rfloor$ different tuples in $\mathfrak{L}_n$. Notice that the size of each $\mathcal{T}_L$ with $L \in \mathfrak{L}_n$ is polynomial in n and so N is superpolynomial in the size of each $\mathcal{T}_L$ with $L \in \mathfrak{L}_n$.

Every $\mathcal{T}_L$, for $L \in \mathfrak{L}_n$, entails, among other inclusions, $\bigsqcap_{i=1}^n C_i \sqsubseteq A$, where every C_i is either A_i or B_i. There are 2^n different such inclusions, which indicates that every representation of the 'intersection TBox' requires superpolynomially many axioms. It follows from Lemma 15 below that this is indeed the case.

The following lemma (proved in the appendix) enables us to respond to entailment queries without eliminating too many TBoxes from the list S of TBoxes that the learner cannot distinguish. We use Σ_n to denote the signature of $\mathcal{T}_L$.

Lemma 14 *For all $\mathcal{EL}$ concept inclusions $C \sqsubseteq D$ over Σ_n:*

- *either $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$ or*
- *the number of different $L \in \mathfrak{L}_n$ such that $\mathcal{T}_L \models C \sqsubseteq D$ does not exceed $|C|$.*

To illustrate the result, consider two TBoxes $\mathcal{T}_L$ and $\mathcal{T}_{L'}$, where $L = (\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_n)$ and $L' = (\boldsymbol{\sigma}'_1, \dots, \boldsymbol{\sigma}'_n)$. Then the inclusion $X_0 \sqcap \exists \boldsymbol{\sigma}_1.M \sqcap \exists \boldsymbol{\sigma}'_1.M \sqcap A_2 \sqcap \dots \sqcap A_n \sqsubseteq A$ is entailed by both $\mathcal{T}_L$ and $\mathcal{T}_{L'}$ but not by any other $\mathcal{T}_{L''}$ with $L \in \mathfrak{L}_n$.

We now show how the oracle can answer equivalence queries, aiming to show that for any polynomial size equivalence query $\mathcal{H}$, the oracle can return a counterexample $C \sqsubseteq D$ such that either (i) $\mathcal{H} \models C \sqsubseteq D$ and $\mathcal{T}_L \models C \sqsubseteq D$ for at most one $L \in \mathfrak{L}_n$ or (ii) $\mathcal{H} \not\models C \sqsubseteq D$ and for every $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L \models C \sqsubseteq D$. Thus, such a counterexample eliminates at most one $\mathcal{T}_L$ from the set S of TBoxes that the learner cannot distinguish. In addition, however, we have to take extra care of the size of counterexamples as the learning algorithm is allowed to run in time polynomial not only in the size of the target TBox but also in the size of the counterexamples returned by the oracle. For instance, if the hypothesis TBox $\mathcal{H}$ contains an inclusion $C \sqsubseteq D$ which is not entailed by any $\mathcal{T}_L$, one cannot simply return $C \sqsubseteq D$ as a counterexample since the learner will be able to 'pump up' its running time by asking for equivalence of the target TBox a sequence of hypotheses $\mathcal{H}_i = \{C_i \sqsubseteq D_i\}$ such that the size of $C_{i+1} \sqsubseteq D_{i+1}$ is twice the size of $C_i \sqsubseteq D_i$. Then at every stage in a run of the learning algorithm, the running time will be polynomial in the size of the input and the size of the largest counterexample received so far, but the overall running time will be exponential in the size of input. The following lemma addresses this issue.

Lemma 15 *For any $n > 1$ and any $\mathcal{EL}$ TBox $\mathcal{H}$ in Σ_n with $|\mathcal{H}| < 2^n$, there exists an $\mathcal{EL}$ CI $C \sqsubseteq D$ over Σ_n such that (i) the size of $C \sqsubseteq D$ does not exceed $6n$ and (ii) if $\mathcal{H} \models C \sqsubseteq D$ then $\mathcal{T}_L \models C \sqsubseteq D$ for at most one $L \in \mathfrak{L}_n$ and if $\mathcal{H} \not\models C \sqsubseteq D$ then for every $L \in \mathfrak{L}_n$ we have $\mathcal{T}_L \models C \sqsubseteq D$.*

Then we have the following.

Theorem 16 *$\mathcal{EL}$ TBoxes are not polynomial time learnable using entailment and equivalence queries.*

[4] In fact, to prove non-polynomial learnability, it suffices to consider $\exists \boldsymbol{\sigma}_1.M \sqcap \dots \sqcap \exists \boldsymbol{\sigma}_n.M \sqsubseteq A$ in place of the concept equality; however, inclusions of this form are not allowed in acyclic TBoxes.

Proof. Assume that TBoxes are polynomial time learnable. Then there exists a learning algorithm whose running time is bounded at any stage by a polynomial $p(n, m)$. Choose n such that $\lfloor 2^n/n \rfloor > (p(n, 6n))^2$ and let $S = \{\mathcal{T}_L \mid L \in \mathfrak{L}_n\}$. We follow Angluin's strategy of letting the oracle remove TBoxes from S in such a way that the learner cannot distinguish between any of the remaining TBoxes. Given an entailment query $C \sqsubseteq D$, if $\mathcal{T}_L \models C \sqsubseteq D$ for every $L \in \mathfrak{L}_n$, then the answer is 'yes'; otherwise the answer is 'no' and all $\mathcal{T}_L$ with $\mathcal{T}_L \models C \sqsubseteq D$ are removed from S (by Lemma 14, there are at most $|C|$ such TBoxes). Given an equivalence query $\mathcal{H}$, the answer is 'no', a counterexample $C \sqsubseteq D$ guaranteed by Lemma 15 is produced, and (at most one) $\mathcal{T}_L$ such that $\mathcal{T}_L \models C \sqsubseteq D$ is removed from S.

As all counterexamples produced are smaller than $6n$, the overall running time of the algorithm is bounded by $p(n, 6n)$. Hence, the learner asks no more than $p(n, 6n)$ queries and the size of every query does not exceed $p(n, 6n)$. By Lemmas 14 and 15, at most $(p(n, 6n))^2$ TBoxes are removed from S during the run of the algorithm. But then, the algorithm cannot distinguish between any remaining TBoxes and we have derived a contradiction. ❑

We now show that DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes cannot be learnt in polynomial time using equivalence queries only. We use the following result on non-learnability of monotone DNF formulas using equivalence queries due to Angluin (1990). Here, equivalence queries take a hypothesis ψ in the form of a monotone DNF formula and return as a counterexample either a truth assignment that satisfies ψ but not the target formula ϕ or vice versa. Let $M(n, t, s)$ denote the set of all monotone DNF formulas whose variables are $x_1, \dots, x_n$, that have exactly t conjunctions, and where each conjunction contains exactly s variables.

Theorem 17 (Angluin) *For any polynomial $q(\cdot)$ there exist constants t_0 and s_0 and a strategy for the oracle $\mathfrak{O}$ to answer equivalence queries posed by a learning algorithm in such a way that for sufficiently large n any learning algorithm that asks at most $q(n)$ equivalence queries, each bounded in size by $q(n)$, cannot exactly identify elements of $M(n, t_0, s_0)$.*

To employ Theorem 17, we associate with every monotone DNF formula $\phi = \bigvee_{i=1}^{t}(x^i_1 \wedge \dots \wedge x^i_{s^i})$ with variables $x_1, \dots, x_n$ a DL-Lite$^{\exists}_{\mathcal{R}}$ TBox $\mathcal{T}_\phi$ as follows. With each conjunct $x^i_1 \wedge \dots \wedge x^i_{s^i}$ we associate a concept $C_i := \exists\rho^i_1.\exists\rho^i_2.\dots.\exists\rho^i_n.\top$ where $\rho^i_j = r$ if x_j occurs in $x^i_1 \wedge \dots \wedge x^i_{s^i}$ and $\rho^i_j = \bar{r}$ otherwise (r and $\bar{r}$ are role names). Let A be a concept name and set

$$\mathcal{T}_\phi = \{A \sqsubseteq \textstyle\bigsqcap_{i=1}^{t} C_i, \quad \bar{r} \sqsubseteq r\}.$$

For example, for $n = 4$ and $\phi = (x_1 \wedge x_4) \vee x_2$ we have

$$\mathcal{T}_\phi = \{A \sqsubseteq \exists r.\exists\bar{r}.\exists\bar{r}.\exists r.\top,\ A \sqsubseteq \exists\bar{r}.\exists r.\exists\bar{r}.\exists\bar{r}.\top,\ \bar{r} \sqsubseteq r\}.$$

A truth assignment I (for the variables $x_1 \dots, x_n$) also corresponds to a concept $C_I := \exists\rho^i_1.\exists\rho^i_2.\dots.\exists\rho^i_n.\top$, where $\rho^i_j = r$ if I makes x_j true and $\rho^i_j = \bar{r}$ otherwise. Then $I \models \phi$ iff $\mathcal{T}_\phi \models A \sqsubseteq C_I$ holds for all truth assignments I.

Note that $\bar{r}$ represents that a variable is false and r that a variable is true. Thus, the role inclusion $\bar{r} \sqsubseteq r$ captures the monotonicity of the DNF formulas considered. For any fixed values n, s and t, we set $T(n, t, s) = \{\mathcal{T}_\phi \mid \phi \in M(n, t, s)\}$.

Theorem 18 *The class of DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes is not polynomial time learnable using equivalence queries.*

Proof. We sketch the proof for the case when the TBoxes $\mathcal{H}$ from any equivalence query represent a monotone DNF formula in the variables $x_1, \dots, x_n$, that is, if all equivalence queries $\mathcal{H}$ are of the form

$$\{A \sqsubseteq \textstyle\bigsqcap_{\rho_1 \cdots \rho_n \in \Gamma} \exists\rho_1.\exists\rho_2.\dots.\exists\rho_n.\top, \quad \bar{r} \sqsubseteq r\},$$

for some $\Gamma \subseteq \{r, \bar{r}\}^n$. Arbitrary DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes in equivalence queries are considered in the appendix.

For a proof by contradiction, suppose that the running time of a learning algorithm $\mathfrak{A}$ for DL-Lite$^{\exists}_{\mathcal{R}}$ TBoxes in $\Sigma = \{A, r, \bar{r}\}$ is bounded at every stage of computation by a polynomial $p(x, y)$, where x is the size of the target TBox, and y is the maximal size of a counterexample returned by the oracle up to the current stage of computation. Let $q(n) = p(n^2, 4n + 2)$, and let t_0 and s_0 be the constants from the strategy of the oracle in Theorem 17. Let n be sufficiently large so that the claim of Theorem 17 holds.

Consider the oracle $\mathfrak{O}'$ that responds to every equivalence query $\mathcal{H}$ that represents a monotone DNF-formula ψ in n variables by returning the counterexample $A \sqsubseteq C_I$ corresponding to the truth assignment I that $\mathfrak{O}$ would return for the equivalence query ψ. We show that $\mathfrak{A}$ cannot distinguish between certain TBoxes in $T(n, t_0, s_0)$ and thus obtain a contradiction.

The largest counterexample returned by $\mathfrak{O}'$ is of the form $A \sqsubseteq \exists\rho_1.\dots\exists\rho_n.\top$, so for sufficiently large n the maximal size of any counterexample returned by $\mathfrak{O}'$ in any run of the algorithm is bounded by $4n + 2$. Similarly, the size of every potential target TBox $\mathcal{T}_\phi \in T(n, t_0, s_0)$ does not exceed $t_0 \cdot (4n + 2)$ and, as t_0 is a constant, for sufficiently large n it is bounded by n^2. Thus, for sufficiently large n the total running time of $\mathfrak{A}$ for any target TBox in $T(n, t_0, s_0)$ is bounded by $p(n^2, 4n + 2)$. So, the size of a monotone DNF equivalence query forwarded to the strategy $\mathfrak{O}$ is bounded by $q(n)$, and there will be at most $q(n)$ queries forwarded. But then $\mathfrak{O}$ returns answers such that some ϕ and ψ from $M(n, t_0, s_0)$ are not distinguished. It remains to observe that $\mathfrak{A}$ does not distinguish $\mathcal{T}_\phi$ and $\mathcal{T}_\psi$. ❑

Future Work

We have presented the first study of learnability of DL TBoxes in Angluin et al's framework of learning via queries. Many research questions remain to be explored. An immediate question is whether acyclic $\mathcal{EL}$ TBoxes can be learned in polynomial time using queries and counterexamples of the form $A \equiv C$ and $A \sqsubseteq C$ only. Note that our non-learnability result for acyclic $\mathcal{EL}$ TBoxes relies heavily on counterexamples that are not of this form. Another immediate question is whether the extension of $\mathcal{EL}_{\mathsf{lhs}}$ with inverse roles (which is a better approximation of OWL2 RL than $\mathcal{EL}_{\mathsf{lhs}}$ itself) can still

be learned in polynomial time. Other interesting research directions are non-polynomial time learning algorithms for $\mathcal{EL}$ TBoxes and the admission of different types of membership queries and counterexamples in the learning protocol. For example, one could replace concept inclusions as counterexamples with interpretations. In an OBDA context, one could allow membership queries that speak about certain answers to queries over an ABox and relative to the target TBox. Our results provide a good starting point for studying such variations.

Acknowledgements Konev and Wolter were supported by the EPSRC project EP/H043594/1. Lutz was supported by the DFG project Prob-DL (LU1417/1-1). Ozaki is supported by the Science without Borders scholarship programme.

References

Angluin, D.; Frazier, M.; and Pitt, L. 1992. Learning conjunctions of Horn clauses. *Machine Learning* 9:147–164.

Angluin, D. 1987a. Learning propositional Horn sentences with hints. Technical report, Yale University.

Angluin, D. 1987b. Learning regular sets from queries and counterexamples. *Inf. Comput.* 75(2):87–106.

Angluin, D. 1987c. Queries and concept learning. *Machine Learning* 2(4):319–342.

Angluin, D. 1990. Negative results for equivalence queries. *Machine Learning* 5:121–150.

Arias, M., and Balcázar, J. L. 2011. Construction and learnability of canonical Horn formulas. *Machine Learning* 85(3):273–297.

Arias, M., and Khardon, R. 2002. Learning closed Horn expressions. *Inf. Comput.* 178(1):214–240.

Arias, M.; Khardon, R.; and Maloberti, J. 2007. Learning Horn expressions with LOGAN-H. *Journal of Machine Learning Research* 8:549–587.

Baader, F., and Molitor, R. 2000. Building and structuring description logic knowledge bases using least common subsumers and concept analysis. In *ICCS*, 292–305.

Baader, F.; Calvanese, D.; McGuiness, D.; Nardi, D.; and Patel-Schneider, P. 2003. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press.

Baader, F.; Ganter, B.; Sertkaya, B.; and Sattler, U. 2007. Completing description logic knowledge bases using formal concept analysis. In *IJCAI*, 230–235.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *IJCAI*, 364–369. Professional Book Center.

Borchmann, D., and Distel, F. 2011. Mining of $\mathcal{EL}$-GCIs. In *The 11th IEEE International Conference on Data Mining Workshops*. Vancouver, Canada: IEEE Computer Society.

Buitelaar, P.; Cimiano, P.; and Magnini, B., eds. 2005. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *Journal of Automated reasoning* 39(3):385–429.

Cimiano, P.; Hotho, A.; and Staab, S. 2005. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res. (JAIR)* 24:305–339.

Frazier, M., and Pitt, L. 1993. Learning from entailment: An application to propositional horn sentences. In *ICML*, 120–127.

Frazier, M., and Pitt, L. 1996. Classic learning. *Machine Learning* 25(2-3):151–193.

Konev, B.; Ludwig, M.; Walther, D.; and Wolter, F. 2012. The logical difference for the lightweight description logic EL. *J. Artif. Intell. Res. (JAIR)* 44:633–708.

Konev, B.; Lutz, C.; Ozaki, A.; and Wolter, F. 2013. Appendix of: Exact learning of lightweight description logic ontologies. http://cgi.csc.liv.ac.uk/~frank/publ/publ.html.

Lehmann, J., and Hitzler, P. 2010. Concept learning in description logics using refinement operators. *Machine Learning* 78(1-2):203–250.

Lutz, C.; Piro, R.; and Wolter, F. 2011. Description logic TBoxes: Model-theoretic characterizations and rewritability. In *IJCAI*, 983–988.

Ma, Y., and Distel, F. 2013. Learning formal definitions for snomed ct from text. In *AIME*, 73–77.

Motik, B.; Grau, B. C.; Horrocks, I.; Wu, Z.; Fokoue, A.; and Lutz, C. 2009. OWL 2 web ontology language: Profiles. W3C Recommendation.

Raedt, L. D. 1997. Logical settings for concept-learning. *Artif. Intell.* 95(1):187–201.

Reddy, C., and Tadepalli, P. 1999. Learning Horn definitions: Theory and an application to planning. *New Generation Comput.* 17(1):77–98.

Selman, J., and Fern, A. 2011. Learning first-order definite theories via object-based queries. In *ECML/PKDD (3)*, 159–174.

ten Cate, B.; Dalmau, V.; and Kolaitis, P. G. 2012. Learning schema mappings. In *ICDT*, 182–195.

Voelker, J. 2009. *Learning Expressive Ontologies*. Ph.D. Dissertation, Universitaet Karlsruhe.

Nominal Schemas in Description Logics: Complexities Clarified

Markus Krötzsch and **Sebastian Rudolph**
Technische Universität Dresden, Germany
{*markus.kroetzsch,sebastian.rudolph*}*@tu-dresden.de*

Abstract

Nominal schemas extend description logics (DLs) with a restricted form of variables, thus integrating rule-like expressive power into standard DLs. They are also one of the most recently introduced DL features, and in spite of many works on algorithms and implementations, almost nothing is known about their computational complexity and expressivity. We close this gap by providing a comprehensive analysis of the reasoning complexities of a wide range of DLs—from $\mathcal{EL}$ to $\mathcal{SROIQ}$—extended with nominal schemas. Both combined and data complexities increase by one exponential in most cases, with the one previously known case of $\mathcal{SROIQ}$ being the main exception. Our proofs employ general modeling techniques that exploit the power of nominal schemas to succinctly represent many axioms, and which can also be applied to study DLs beyond those we consider. To further improve our understanding of nominal schemas, we also investigate their semantics, traditionally based on finite grounding, and show that it can be extended to infinite sets of individuals without affecting reasoning complexities. We argue that this might be a more suitable semantics when considering entailments of axioms with nominal schemas.

1 Introduction

The fruitful integration of reasoning on both schema and instance level poses a continued challenge to knowledge representation and reasoning. While description logics (DLs) excel at the former task, rule-based formalisms are often more adequate for the latter. An established and highly productive strand of research therefore continues to investigate ways of reconciling both paradigms.

A practical breakthrough in this area was the discovery of *DL-safe rules*, which ensure decidability of reasoning by restricting the applicability of rules to a finite set of elements that are denoted by an individual name (Motik, Sattler, and Studer 2005). As of today, DL-safe rules are the most widely used DL-rule extension, supported by several mainstream reasoners (Kolovski, Parsia, and Sirin 2006; Motik, Shearer, and Horrocks 2009).

More recently, *nominal schemas* have been proposed as an even tighter integration of "DL-safe" instance reasoning with DL schema reasoning (Krötzsch et al. 2011). A *nominal* is a DL concept expression $\{a\}$ that represents a singleton set containing only (the individual denoted by) a. Nominal schemas replace a by a variable x that ranges over all individual names, so that it might represent arbitrary nominals $\{a\}$, where all occurrences of $\{x\}$ in one axiom represent the same nominal. For example,

$$\exists \texttt{hasFather}.\{x\} \sqcap \exists \texttt{hasMother}.(\{y\} \sqcap \exists \texttt{married}.\{x\})$$

represents the set of all individuals whose father (x) and mother (y) are married to each other, where the parents must be represented by individual names. No standard DL can express this in such a concise way. The interplay with other DL features also makes nominal schemas more expressive than the combination of DLs and DL-safe rules (cf. Sect. 8).

Nominal schemas have thus caused significant research interest, and several reasoning algorithms that exploit this succinct representation have been proposed (Krisnadhi and Hitzler 2012; Wang and Hitzler 2013; Steigmiller, Glimm, and Liebig 2013; Martínez, Wang, and Hitzler 2013). Most recently, Steigmiller, Glimm, and Liebig (2013) demonstrated that such algorithms can even outperform other systems for reasoning with DL-safe rules.

Surprisingly and in sharp contrast to these successes, many basic questions about the expressivity and complexity of nominal schemas have remained unanswered. A naive reasoning approach is based on replacing nominal schemas by nominals in all possible ways, which leads to complexity upper bounds one exponential above the underlying DL. The only tight complexity result so far is that the N2ExpTime combined complexity of reasoning in the DL $\mathcal{SROIQ}$ is not affected by nominal schemas—a result that reveals almost nothing about the computational or expressive impact of nominal schemas in general (Krötzsch et al. 2011). Beyond this singular result, it is only known that nominal schemas can simulate Datalog rules of any arity using $\exists$, $\sqcap$, and the universal role U (Knorr, Hitzler, and Maier 2012).

In this paper, we give a comprehensive account of the reasoning complexities of a wide range of DLs, considering both combined complexities (w.r.t. the size of the given knowledge base) and data complexities (w.r.t. the size of the ABox only). Figure 1 summarizes our results for combined complexities for DLs with nominal schemas (right; marked by the letter $\mathcal{V}$) in comparison with known complexities of DLs with nominals (left). It turns out that $\mathcal{SROIQ}$ is an exception, while most other DLs experience exponential complexity increases due to nominal schemas.

The effects on the data complexity are even more striking. The data complexity of standard DLs is either in P (for $\mathcal{EL}$ and Horn-DLs, which restrict the use of $\sqcup$ and $\neg$) or in NP. In contrast, the data complexities for all nominal-schema DLs in Fig. 1 are only one exponential below their combined complexity, i.e., ExpTime or NExpTime for most cases.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

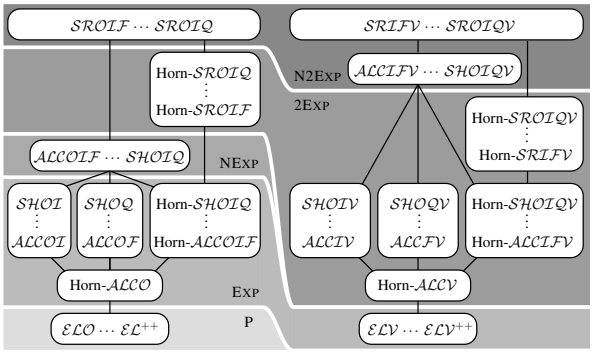


Figure 1: Combined complexities for DLs with nominals compared to DLs with nominal schemas

To obtain these results, we identify general modeling techniques that use nominal schemas to express complex schema information very succinctly. Two fundamental techniques provide the basis for most of our hardness proofs:

- **TBox-to-ABox Internalization** A TBox is replaced by a small set of "template axioms" with nominal schemas, and the original TBox is expressed with ABox assertions.
- **GCI Iterators** Templates of TBox axioms (general concept inclusions, short GCIs) are instantiated by replacing placeholder concepts by concepts from an exponentially long list of "indexed" concept names.

TBox-to-ABox internalization explains why the data complexity of most DLs with nominal schemas agrees with the combined complexity of their underlying standard DL. $\mathcal{SROIQV}$ is a noteworthy exception where the internalization is not possible. GCI iterators are a kind of generalized TBox axiom that can be used to encode exponentially large TBoxes polynomially using nominal schemas. This technique can be applied to TBoxes from known hardness proofs to boost complexities by one exponential. Both techniques provide concrete illustrations for the expressive power of nominal schemas and outline ways to obtain results for DLs that we did not consider.

After establishing these results, we revisit the formal semantics of nominal schemas. Normally, nominal schemas are considered to represent a finite set of nominals, based on individuals that either occur in the knowledge base or are part of some finite signature. This can lead to unintuitive effects, since entailments may become invalid when adding more individuals. We thus study the semantics obtained when using an infinite set of individual names instead. This makes it impossible to replace nominal schemas by nominals in all possible ways to decide entailment. Surprisingly, reasoning is still decidable with the same complexity results. Indeed, the consequences of both approaches turn out to agree under some mild assumptions.

2 Standard Description Logics

The DLs we consider are based on the well-known DL $\mathcal{SROIQ}$ (Horrocks, Kutz, and Sattler 2006). The precise syntax and semantics that we adhere to is explained in detail in the freely available DL Primer (Krötzsch, Simančík, and Horrocks 2012), so we will not repeat it here. We also consider Horn-DLs, which, in essence, forbid $\sqcup$ and $\neg$, and restrict the use of $\forall$ (Krötzsch, Rudolph, and Hitzler 2013).

Our results apply to many DLs. The smallest DL we consider is $\mathcal{EL}$, allowing only $\sqcap$, $\exists$, and $\top$. We do not consider variants of DL-Lite, since these DLs typically exclude nominals, and are therefore no natural candidates for extension with nominal schemas. We also exclude lightweight approaches like DLP, pD*, and OWL RL, since the rule-like expressiveness of these DLs is subsumed completely by nominal schemas (see Section 4). We thus use the term *standard description logic* to mean a DL $\mathcal{L}$ with $\mathcal{EL} \subseteq \mathcal{L} \subseteq \mathcal{SROIQ}$. Our complexity results refer to standard reasoning tasks, and this is what the term *reasoning* will refer to. Formally, a *standard reasoning problem* of $\mathcal{L}$ is any problem that can be polynomially reduced to checking if an $\mathcal{L}$ knowledge base KB entails a fact $A(a)$ (note that $\mathcal{EL}$ consistency checking is trivial). Nondeterminstic complexity classes may occur in two versions, e.g., NExpTime and co-NExpTime, depending on the exact reasoning task.

Two knowledge bases are semantically equivalent if they have the same models. This is often too restrictive, since many "essentially equivalent" transformations make use of auxiliary symbols that affect the set of models. A useful way to describe that a knowledge base faithfully captures the semantics of another one is to use *conservative extensions*.

Definition 1. *Consider a knowledge base* KB *over a signature* Σ*, and a knowledge base* KB$'$ *over a signature* Σ' *that extends* Σ*. Then* KB$'$ *is a* (model-)conservative extension *of* KB *if (a) every model of* KB$'$ *is a model of* KB*, and (b) every model of* KB *can be extended to a model of* KB$'$ *by adding suitable interpretations for additional signature symbols. Note that we do not require* KB $\subseteq$ KB$'$.

Essentially all mainstream DLs allow axioms to be normalized. We exploit this in some of our proofs.

Definition 2. *A DL* $\mathcal{L}$ *is called normalizable, if there is a finite set* $\{\alpha_1, \ldots, \alpha_k\}$ *of* $\mathcal{L}$ *axioms and a polynomially computable function* τ *that maps* $\mathcal{L}$ *knowledge bases to* $\mathcal{L}$ *knowledge bases such that for every* $\mathcal{L}$ *knowledge base* KB

- τ(KB) *is a conservative extension of* KB*, and*
- *every axiom of* τ(KB) *can be obtained from some* $\alpha \in \{\alpha_1, \ldots, \alpha_k\}$ *by renaming signature symbols.*

3 Nominal Schemas: Syntax and Semantics

Recall that a $\mathcal{SROIQ}$ knowledge base is based on a signature consisting of finite sets N_I of individual names, N_C of concept names, and N_R of role names. To define nominal schemas, we extend signatures by an additional finite set N_V of variable names. A nominal schema is a concept of the form $\{x\}$ with $x \in \mathsf{N}_V$, which can be used in axioms and complex concept expressions like any other $\mathcal{SROIQ}$ concept. DLs with nominal schemas are denoted by their standard name extended by $\mathcal{V}$ as in $\mathcal{SROIQV}$ or $\mathcal{ELV}$.

The semantics of nominal schemas are defined by *grounding*. For a $\mathcal{SROIQV}$ axiom α, ground(α) is the set of all $\mathcal{SROIQ}$ axioms obtained from α by uniformly replacing all variables in α by individual names in N_I in all

possible ways. The grounding of a knowledge base KB is $\mathsf{ground}(\mathrm{KB}) := \bigcup_{\alpha \in \mathrm{KB}} \mathsf{ground}(\alpha)$. Note that $\mathsf{ground}(\mathrm{KB})$ is finite, since N_I is finite. An interpretation satisfies a $\mathcal{SROIQV}$ knowledge base KB if it satisfies $\mathsf{ground}(\mathrm{KB})$, and satisfiability and entailment are defined accordingly.

The semantics of nominal schemas depends on the set N_I of individual names in the given signature. For example,

$$\{a\} \sqsubseteq \{b\} \models \{a\} \sqsubseteq \{x\} \tag{1}$$

is a valid entailment in a signature with $\mathsf{N}_I = \{a,b\}$, but not in one with $\mathsf{N}_I = \{a,b,c\}$. However, the choice of N_I (as long as it is non-empty) is only relevant when asking if axioms with nominal schemas are entailed but not for standard reasoning problems, such as instance or subsumption checking. Hence we often do not mention N_I explicitly, though, in general, always assume that the signature is given. Moreover, instances of entailment problems are always considered in the context of a single signature that covers all involved axioms.

One could take other approaches, which are worth mentioning. First, instead of using N_I, one could define nominal schemas to refer to all individual names that occur in the knowledge base. This raises questions when considering entailment problems (Which of the knowledge bases? Both?), and it leads to a kind of non-monotonicity, where $\{a\} \sqsubseteq \{x\}$ follows from $\{a\} \sqsubseteq \{b\}$, while the logically weaker axiom $\{a\} \sqsubseteq \{x\} \sqcup \{c\}$ does not (since it extends the range of $\{x\}$). Secondly, to overcome this problem, one could restrict the range of each nominal schema explicitly by requiring additional syntax. The original formulation of DL-safe rules, e.g., requires each rule variable x to appear in an atom of the form $O(x)$ in the premise, and O was defined in the ABox (Motik, Sattler, and Studer 2005). For DL axioms, however, such restrictions would unduly clutter the syntax. Finally, instead of considering N_I to be variable and finite, one could consider an infinite signature used by all knowledge bases. Grounding is no longer possible in this approach, and it is not the semantics that was assumed in previous works. We study this approach in Section 8, and find that it leads to mostly the same results as the traditional formulation.

Regarding reasoning complexity, it is important that we do not assume N_I to be fixed. Rather, the signature is part of the problem instance of an entailment problem. Therefore, the *size of a problem* always includes the size of the signature. Analogously, *data complexity* refers to the complexity of a problem under the assumption that the size of TBox and RBox is bounded by a constant, while the ABox *and* the signature can vary. When speaking of the size of a knowledge base or of an ABox, we always assume that the size of the signature is included.

Since the size of $\mathsf{ground}(\mathrm{KB})$ is exponential in the size of KB and polynomial in the size of the ABox (where the number of variables per TBox axiom is bounded), we obtain some general upper bounds for the complexity of reasoning.

Theorem 1. *Consider a DL $\mathcal{L}$. The combined complexity of $\mathcal{LV}$ is at most exponentially higher than that of $\mathcal{LO}$. If the combined complexity of $\mathcal{LO}$ is in a complexity class* C *that subsumes* P*, then the data complexity of $\mathcal{LV}$ is in* C *as well.*

4 Expressiveness of Nominal Schemas

We now take a closer look at the expressive power of nominal schemas, and draw some first conclusions about the complexity of reasoning. It was immediately observed that nominal schemas can be used to encode DL-safe rules (Krötzsch et al. 2011), and this result has later been extended to show that they can express extended DL-safe rules with predicates of any arity (Knorr, Hitzler, and Maier 2012). The universal role U was used in both encodings. However, it is easy to see that we can remove this requirement. To represent a first-order logic atom $p(t_1,\ldots,t_n)$, where $t_i \in \mathsf{N}_I \cup \mathsf{N}_V$, we use a fresh concept A_p and auxiliary roles $\mathtt{atom}$, $\mathtt{arg}_1$, …, $\mathtt{arg}_n$. We define $\mathsf{enc}(p(t_1,\ldots,t_n))$ to be the formula

$$\exists \mathtt{atom}.(A_p \sqcap \exists \mathtt{arg}_1.\{t_1\} \sqcap \ldots \sqcap \exists \mathtt{arg}_n.\{t_n\}). \tag{2}$$

A rule $B_1 \wedge \ldots \wedge B_\ell \rightarrow H$, where $B_1,\ldots,B_\ell,H$ are first-order atoms, can now be encoded by the DL axiom

$$\mathsf{enc}(B_1) \sqcap \ldots \sqcap \mathsf{enc}(B_\ell) \sqsubseteq \mathsf{enc}(H) \tag{3}$$

(as usual, the left-hand side is $\top$ if $\ell = 0$). It is easy to see that this captures the semantics of an arbitrary set of function-free Horn rules (a.k.a. Datalog). Hence, the simple DL $\mathcal{ELV}$ can already capture Datalog and thus inherits its ExpTime complexity (Dantsin et al. 2001). The matching upper bound follows from Theorem 1 and the polynomial combined complexity of $\mathcal{EL}$ (Baader, Brandt, and Lutz 2005).

Theorem 2. *Standard reasoning in $\mathcal{ELV}$ is* ExpTime*-complete for combined complexity.*

Note that the encoding of rules does not provide any interaction between unary and binary rule predicates and DL concepts and roles, respectively. We could add this using axioms like $\exists \mathtt{atom}.(A_r \sqcap \exists \mathtt{arg}_1.\{x\} \sqcap \exists \mathtt{arg}_2.\{y\}) \sqsubseteq \exists \mathtt{aux}.(\{x\} \sqcap \exists r.\{y\})$. However, some DL knowledge bases may interfere with the correctness of the encoding if they restrict the overall domain size. We address this issue in the next section.

In addition to the above, it is intuitively clear that nominal schemas are at least as expressive as nominals. The next definition provides a suitable construction. As before, we only need features of $\mathcal{EL}$.

Definition 3. *Let* KB *be a knowledge base. The knowledge base* $\mathsf{nomelim}(\mathrm{KB})$ *is obtained from* KB *as follows:*

- *replace all occurrences of $\{a\}$ in* KB *by O_a, where O_a is a fresh concept name,*
- *add an ABox axiom $O_a(a)$ for all $a \in \mathsf{N}_I$ in* KB*,*
- *add the axiom $\top \sqsubseteq \exists \mathtt{somenom}.\{x\}$, where* somenom *is a fresh role name, and*
- *add the axiom $O_a \sqcap \exists \mathtt{somenom}.(\{x\} \sqcap O_a) \sqsubseteq \{x\}$ for each of the fresh O_a.*

Essentially this procedure installs atomic concepts O_a which are axiomatized to have the same extension as the nominals they substitute. It is clear that one can compute $\mathsf{nomelim}(\mathrm{KB})$ in polynomial time. Moreover, $\mathsf{nomelim}(\mathrm{KB})$ is a conservative extension of KB. Part (a) of Definition 1 is straightforward since $\{a\} \equiv O_a$ is a direct consequence of $\mathsf{nomelim}(\mathrm{KB})$ for every nominal $\{a\}$ of KB. For (b) we enrich any model $\mathcal{I}$ of KB by $O_a^{\mathcal{I}} = \{a^{\mathcal{I}}\}$ and $\mathtt{somenom}^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \{a^{\mathcal{I}} \mid a \text{ individual in KB}\}$ to find a model of $\mathsf{nomelim}(\mathrm{KB})$. This leads to the following general result.

Theorem 3. *For any DL $\mathcal{L}$ with $\mathcal{EL} \subseteq \mathcal{L}$, the combined and data complexities of standard reasoning in $\mathcal{LO}$ are lower bounds for the respective complexities of $\mathcal{LV}$.*

Nominal schemas thus generalize the expressiveness of nominals, DL-safe rules, and (DL-safe) Datalog. Nevertheless, using grounding, all of these can be expressed with nominals already, though an exponential loss in efficiency is unavoidable in general. DL-safe rules, in contrast, cannot even express nominals. The Datalog encoding (3) illustrates a type of expression that is not expressible in DLs extended with DL-safe rules. Indeed, if we consider all rules and axioms as first-order logic formulae, then the existential quantifiers in the conclusion of (3) occur in the scope of (i.e., "depend on") an arbitrary number of nominal schema assignments. In contrast, existential quantifiers in DLs (with or without DL-safe rules) occur only in axioms that share exactly one variable between the left and right formula. In other words, using Skolemization to eliminate quantifiers, one only needs unary Skolem functions in standard DLs, but Skolem functions of arbitrary arity with nominal schemas.

A related property of DLs with nominal schemas is that they cannot be normalized in the sense of Definition 2. Our complexity results allow us to show the following theorem at the end of Section 7; the case of $\mathcal{SROIQ}$ remains open.

Theorem 4. *No DL $\mathcal{LV}$ with $\mathcal{ELV} \subseteq \mathcal{LV} \subseteq \mathcal{SHOIQV}$ is normalizable.*

5 Domain Size Matters

Several of our results are based on transforming a knowledge base by using fresh individuals to represent classes. For example, for a concept `Car`, we can introduce an individual $c_{\texttt{Car}}$ and express a fact $\texttt{Car}(a)$ as $\texttt{type}(a, c_{\texttt{Car}})$. However, DLs that support nominals can interfere with this encoding.

Example 1. *Consider a knowledge base* KB *containing the TBox axiom $\top \sqsubseteq \{a,b\}$ and the ABox axioms $A(a), \neg B(a), \neg A(b), B(b), C(a), C(b)$. Clearly,* KB *is satisfiable. If we represent classes by individuals as above, we obtain the knowledge base* KB$'$ *with ABox axioms $\texttt{type}(a,c_A), \neg\texttt{type}(a,c_B), \neg\texttt{type}(b,c_A), \texttt{type}(b,c_B), \texttt{type}(a,c_C), \texttt{type}(b,c_C)$. Since every model $\mathcal{I}$ of* KB$'$ *has to map any of c_A, c_B, c_C to either $a^{\mathcal{I}}$ or $b^{\mathcal{I}}$,* KB$'$ *is unsatisfiable.*

To address this obstacle, we show how an $\mathcal{L}$ reasoning problem can be transformed in such a way that it is legitimate to focus attention on models with unbounded domains. The following notion will come handy for this.

Definition 4. *We call a knowledge base* KB unbounded *if, for every model $\mathcal{I}$ of* KB*, there is a model $\mathcal{J}$ of* KB *with infinite domain such that $\mathcal{I} \models \alpha$ iff $\mathcal{J} \models \alpha$ for every atomic concept assertion α.*

Unboundedness is guaranteed in some situations. For example, every $\mathcal{SRIQ}$ knowledge base not using the universal role is unbounded (Mehdi and Rudolph 2011, Lemma 1), and every Horn-$\mathcal{SROIQ}$ knowledge base that has a model with a domain size > 1 is unbounded as well. In the remainder of this section, we show how to transform bounded knowledge bases into unbounded ones. We can then focus on unbounded knowledge bases in the rest of the paper.

Concept expressions of the form $\forall U.C$ complicate the conversion of a knowledge base into an unbounded one, especially if the availability of other modeling features cannot be assumed. However, one can remove such expressions in a way that (next to using the universal role U which is present anyway) only requires features of $\mathcal{EL}$.

Definition 5. *Let* $\mathsf{remuniv}(\text{KB})$ *be the knowledge base obtained from* KB *by performing the following steps for each concept expression of the form $\forall U.C$ contained in* KB*, where $A_{\forall U.C}$ and $Aux_{\forall U.C}$ are fresh concept names:*

- *substitute all occurrences of $\forall U.C$ by $A_{\forall U.C}$*
- *add the following axioms to* KB*:*

$$\exists U.A_{\forall U.C} \sqsubseteq C \tag{4}$$

$$\top \sqsubseteq \exists U.Aux_{\forall U.C} \tag{5}$$

$$\exists U.(Aux_{\forall U.C} \sqcap C) \sqsubseteq A_{\forall U.C} \tag{6}$$

Lemma 5. $\mathsf{remuniv}(\text{KB})$ *is a conservative extension of* KB.

Proof. First, we show that every model of $\mathsf{remuniv}(\text{KB})$ is a model of KB by observing that $\mathsf{remuniv}(\text{KB}) \models A_{\forall U.C} \equiv \forall U.C$ for all the replaced expressions $\forall U.C$. To this end, we make a detour via first-order logic. We start with the Axiom (4) and obtain $\exists U.A_{\forall U.C} \sqsubseteq C \Rightarrow \forall x.(\exists y.(A_{\forall U.C}(y)) \rightarrow C(x)) \Rightarrow (\exists y.(A_{\forall U.C}(y)) \rightarrow \forall x.C(x)) \Rightarrow \forall y.(A_{\forall U.C}(y) \rightarrow \forall x.C(x)) \Rightarrow A_{\forall U.C} \sqsubseteq \forall U.C$.

For the other direction, we start from Axiom (6), $\exists U.(Aux_{\forall U.C} \sqcap C) \sqsubseteq A_{\forall U.C}$, translate it into first-order logic: $\forall x.(\exists y.(Aux_{\forall U.C}(y) \wedge C(y)) \rightarrow A_{\forall U.C}(x))$, transform it into $\forall x.(\forall y.(\neg Aux_{\forall U.C}(y) \vee \neg C(y)) \vee A_{\forall U.C}(x))$, and further into $\forall y.(\neg Aux_{\forall U.C}(y) \vee \neg C(y) \vee \forall x.(A_{\forall U.C}(x)))$ to finally arrive at $\forall y.(Aux_{\forall U.C}(y) \rightarrow (\neg C(y) \vee \forall x.(A_{\forall U.C}(x))))$. Since the FOL version of Axiom (5) is simply $\exists y.Aux_{\forall U.C}(y)$, we can use this to instantiate the premise of the last rule and thus obtain: $\exists y.(\neg C(y) \vee \forall x.(A_{\forall U.C}(x))) \Rightarrow \forall x.(\exists y.(\neg C(y)) \vee A_{\forall U.C}(x)) \Rightarrow \forall x.(\neg \forall y.(C(y)) \vee A_{\forall U.C}(x)) \Rightarrow \forall x.(\forall y.(C(y)) \rightarrow A_{\forall U.C}(x)) \Rightarrow \forall U.C \sqsubseteq A_{\forall U.C}$.

Next, we have to show that every model $\mathcal{I}$ of KB can be extended to a model $\mathcal{J}$ of $\mathsf{remuniv}(\text{KB})$ by providing appropriate extensions for the fresh concept names. This can be achieved via $A^{\mathcal{J}}_{\forall U.C} := (\forall U.C)^{\mathcal{I}}$ and by letting $Aux^{\mathcal{J}}_{\forall U.C} := \top^{\mathcal{I}}$ if $(\neg C)^{\mathcal{I}} = \emptyset$ and $Aux^{\mathcal{J}}_{\forall U.C} := (\neg C)^{\mathcal{I}}$ otherwise. □

Since remuniv is clearly polynomial, we do not have to consider expressions $\forall U.C$ below. Our next transformation to produce an unbounded knowledge base is reminiscent of a technique used to allow the introduction of new individuals for metamodeling (Glimm, Rudolph, and Völker 2010).

Definition 6. *Let* KB *be an $\mathcal{L}$ knowledge base for a standard DL $\mathcal{L}$, without expressions of the form $\forall U.C$. We define* $\mathsf{unbound}(\text{KB})$ *as follows, where* `Top` *is a fresh concept:*

1. *Recursively, outside-in, we replace all concept subexpressions C in all axioms by $\texttt{Top} \sqcap C$.*
2. *If $\mathcal{L}$ allows for universal quantification, we add a GCI $\top \sqsubseteq \forall r.\texttt{Top}$ for every (possibly inverse) role $r \neq U$ in* KB*.*
3. *For every individual name a in* KB*, we add $\texttt{Top}(a)$; if no individual occurs, we introduce a fresh one a_{new}.*

Example 2. *Consider our knowledge base* KB *from Example 1;* $\mathsf{unbound}(\mathrm{KB})$ *contains the axioms* $\mathtt{Top} \sqsubseteq \mathtt{Top} \sqcap \{a,b\}$, $\mathtt{Top}(a)$, $\mathtt{Top}(b)$, $\mathtt{Top} \sqcap A(a)$, $\mathtt{Top} \sqcap \neg(\mathtt{Top} \sqcap B)(a)$, $\mathtt{Top} \sqcap \neg(\mathtt{Top} \sqcap A)(b)$, $\mathtt{Top} \sqcap B(b)$, $\mathtt{Top} \sqcap C(a)$, $\mathtt{Top} \sqcap C(b)$.

Obviously, the described transformation can be computed in polynomial time and outputs a knowledge base of polynomial size. Moreover, for any standard DL $\mathcal{L}$, and any $\mathcal{L}$ knowledge base, $\mathsf{unbound}(\mathrm{KB})$ is an $\mathcal{L}$ knowledge base.

Lemma 6 (Properties of $\mathsf{unbound}$**).**

1. *For any knowledge base* KB *and any model* $\mathcal{J}$ *of* $\mathsf{unbound}(\mathrm{KB})$*, there is a model* $\mathcal{I}$ *of* KB *such that* $\mathcal{I} \models A(a)$ *if and only if* $\mathcal{J} \models \mathtt{Top} \sqcap A(a)$ *for every atomic concept assertion* $A(a)$.
2. *For any model* $\mathcal{I}$ *of a knowledge base* KB*, there is a model* $\mathcal{J}$ *of* $\mathsf{unbound}(\mathrm{KB})$ *with an infinite domain such that* $\mathcal{I} \models A(a)$ *if and only if* $\mathcal{J} \models \mathtt{Top} \sqcap A(a)$ *for every atomic concept assertion* $A(a)$.
3. $\mathsf{unbound}(\mathrm{KB})$ *is unbounded.*
4. $\mathrm{KB} \models A(a)$ *if and only if* $\mathsf{unbound}(\mathrm{KB}) \models \mathtt{Top} \sqcap A(a)$.

Proof. Claim (1). We provide a construction of $\mathcal{I}$. Given a model $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ of $\mathsf{unbound}(\mathrm{KB})$ we define $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ as follows: $\Delta^{\mathcal{I}} := \mathtt{Top}^{\mathcal{J}}$ (note that nonemptyness is guaranteed since $\mathtt{Top}^{\mathcal{J}}$ contains at least one named individual by definition of $\mathsf{unbound}$), $a^{\mathcal{I}} := a^{\mathcal{J}}$ for all $a \in \mathsf{N}_I$, $A^{\mathcal{I}} := A^{\mathcal{J}} \cap \Delta^{\mathcal{I}}$ for $A \in \mathsf{N}_C$, and $r^{\mathcal{I}} := r^{\mathcal{J}} \cap (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$ for $r \in \mathsf{N}_R$. We can now show by induction on the structure of the concept expressions that $C^{\mathcal{I}} = \mathsf{topbound}(C)^{\mathcal{J}}$ for any concept expression C occurring in KB, where $\mathsf{topbound}(C)$ denotes the result of replacing, recursively and outside-in, all concept subexpressions D of C (including C itself) by $\mathtt{Top} \sqcap D$. Base cases and induction steps are straightforward for all cases except for universal quantification.

Thus, consider $C = \forall r.D$, with $\mathsf{topbound}(C) = \mathtt{Top} \sqcap \forall r.\mathsf{topbound}(D)$ and assume $D^{\mathcal{I}} = \mathsf{topbound}(D)^{\mathcal{J}}$ holds by induction hypothesis. Since C occurs in KB, the latter must be formulated in a DL supporting universal quantification, thus $\mathsf{unbound}(\mathrm{KB})$ must by definition contain the axiom $\top \sqsubseteq \forall r.\mathtt{Top}$. Hence $y \in \mathtt{Top}^{\mathcal{J}}$ for all $\langle x,y\rangle \in r^{\mathcal{J}}$. Now we observe equivalence of the following statements: $x \in C^{\mathcal{I}} \iff x \in (\forall r.D)^{\mathcal{I}} \iff x \in \Delta^{\mathcal{I}}$ and $\forall y$ with $\langle x,y\rangle \in r^{\mathcal{I}}$ holds $y \in D^{\mathcal{I}} \iff x \in \mathtt{Top}^{\mathcal{J}}$ and $\forall y$ with $\langle x,y\rangle \in r^{\mathcal{J}} \cap (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$ holds $y \in \mathsf{topbound}(D)^{\mathcal{J}} \iff x \in \mathtt{Top}^{\mathcal{J}}$ and $\forall y$ with $\langle x,y\rangle \in r^{\mathcal{J}} \cap (\mathtt{Top}^{\mathcal{J}} \times \mathtt{Top}^{\mathcal{J}})$ holds $y \in \mathsf{topbound}(D)^{\mathcal{J}} \iff x \in \mathtt{Top}^{\mathcal{J}}$ and $\forall y$ with $\langle x,y\rangle \in r^{\mathcal{J}}$ holds $y \in \mathsf{topbound}(D)^{\mathcal{J}} \iff x \in (\mathtt{Top} \sqcap \forall r.\mathsf{topbound}(D))^{\mathcal{J}} \iff x \in \mathsf{topbound}(C)^{\mathcal{J}}$.

Having established that $C^{\mathcal{I}} = \mathsf{topbound}(C)^{\mathcal{J}}$, we can show that satisfaction of all axioms from $\mathsf{unbound}(\mathrm{KB})$ in $\mathcal{J}$ directly ensures satisfaction of all axioms from KB in $\mathcal{I}$. We also find, for any individual name a and concept name A, that: $\mathcal{I} \models A(a) \iff a^{\mathcal{I}} \in A^{\mathcal{I}} \iff a^{\mathcal{I}} \in A^{\mathcal{J}} \cap \Delta^{\mathcal{I}} \iff a^{\mathcal{J}} \in A^{\mathcal{J}} \cap \mathtt{Top}^{\mathcal{J}} \iff \mathcal{J} \models \mathtt{Top} \sqcap A(a)$ as claimed.

Claim (2). We assume a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of KB and construct $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ as follows: $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}} \cup \mathbb{N}$; $a^{\mathcal{J}} := a^{\mathcal{I}}$ for all $a \in \mathsf{N}_I$, where $a^{\mathcal{J}}_{\text{new}} \in \Delta^{\mathcal{I}}$ if a_{new} has been introduced in $\mathsf{unbound}(\mathrm{KB})$; $A^{\mathcal{J}} := A^{\mathcal{I}}$ for all $A \in \mathsf{N}_C$ interpreted by $\mathcal{I}$; $\mathtt{Top}^{\mathcal{J}} := \Delta^{\mathcal{I}}$; and $r^{\mathcal{J}} := r^{\mathcal{I}}$ for $r \in \mathsf{N}_R$. Again, we can use induction to show that the correspondence $C^{\mathcal{I}} = \mathsf{topbound}(C)^{\mathcal{J}}$ holds and conclude that all axioms from $\mathsf{unbound}(\mathrm{KB})$ are valid in $\mathcal{J}$. Clearly, $\mathcal{J}$ has an infinite domain. We also obtain for any concept name A and individual name a the following equivalence: $\mathcal{J} \models \mathtt{Top} \sqcap A(a) \iff a^{\mathcal{J}} \in A^{\mathcal{J}} \cap \mathtt{Top}^{\mathcal{J}} \iff a^{\mathcal{I}} \in A^{\mathcal{I}} \cap \Delta^{\mathcal{I}} \iff \mathcal{I} \models A(a)$.

Claim (3). This is a direct consequence of the preceding two. For an arbitrary model $\mathcal{J}$ of $\mathsf{unbound}(\mathrm{KB})$ we create the model $\mathcal{I}$ of KB as in the proof of Claim (1). From $\mathcal{I}$, we then construct a model $\mathcal{J}'$ of $\mathsf{unbound}(\mathrm{KB})$ as in the proof of Claim (2). Then we know that $\mathcal{J}$ and $\mathcal{J}'$ satisfy the same atomic concept assertions and $\mathcal{J}'$ has an infinite domain.

Claim (4). This is equivalent to the statement that $\mathrm{KB} \not\models A(a)$ iff $\mathsf{unbound}(\mathrm{KB}) \not\models \mathtt{Top} \sqcap A(a)$. This can be easily shown by seeing that any model of KB not satisfying $A(a)$ can be turned into a model of $\mathsf{unbound}(\mathrm{KB})$ not satisfying $\mathtt{Top} \sqcap A(a)$ (by Claim (2)) and vice versa (by Claim (1)). □

Example 3. *Referring to Example 2, we find that* $\mathsf{unbound}(\mathrm{KB})$ *is satisfiable and, unlike* KB*, it even has a model* $\mathcal{J}$ *with infinite domain* $\Delta^{\mathcal{J}} = \{a^{\mathcal{J}}, b^{\mathcal{J}}\} \cup \mathbb{N}$*. Moreover we find, e.g., the correspondence between* KB *entailing* $A \sqsubseteq C$ *and* $\mathsf{unbound}(\mathrm{KB})$ *entailing* $\mathtt{Top} \sqcap A \sqsubseteq \mathtt{Top} \sqcap C$.

6 Data Complexities

From Theorem 1 and the known complexity results for various DLs, we obtain the following upper bounds for the data complexity: P for $\mathcal{EL}\mathcal{V}^{++}$ (Baader, Brandt, and Lutz 2005), ExpTime for $\mathcal{SHOIV}$, $\mathcal{SHOQV}$, and Horn-$\mathcal{SHOIQV}$ (Hladik 2004; Glimm, Horrocks, and Sattler 2008; Ortiz, Rudolph, and Simkus 2010), NExpTime for $\mathcal{SHOIQV}$ (Tobies 2001), 2ExpTime for Horn-$\mathcal{SROIQV}$ (Ortiz, Rudolph, and Simkus 2010), and N2ExpTime for $\mathcal{SROIQV}$ (Kazakov 2008).[1] We can obtain matching hardness results in many cases, even for much simpler DLs.

Theorem 7. *1. The data complexity of* $\mathcal{ELV}$ *is* P*-hard.*

2. *The data complexity of Horn-*$\mathcal{ALCV}$ *is* ExpTime*-hard.*
3. *The data complexity of* $\mathcal{ALCIFV}$ *is* (co)NExpTime*-hard.*

For $\mathcal{ELV}$, this already follows from the hardness of data complexity for $\mathcal{EL}$. To prove the other cases, we apply a generic method of expressing TBoxes as ABoxes.

Definition 7. *Consider a DL* $\mathcal{L}$ *with* $\mathcal{EL} \subseteq \mathcal{L} \subseteq \mathcal{SROIQ}$ *and an* $\mathcal{L}$ *TBox axiom* $\alpha = C \sqsubseteq D$*. The* template *for* α*, denoted* $\mathsf{tmpl}(\alpha)$*, is defined as follows. Let* $\sigma_1, \ldots, \sigma_n$ *be a list of all individual names and concept names in* α*. Let* A_α *be a fresh concept name, and let* $\mathtt{gci}$, $\mathtt{type}$*, and* $\mathtt{symb}_i$ $(1 \le i \le n)$ *be fresh role names. Then* $\mathsf{tmpl}(\alpha)$ *is the* $\mathcal{LV}$ *axiom*

$$\exists \mathtt{gci}.(A_\alpha \sqcap \exists \mathtt{symb}_1.\{x_1\} \sqcap \ldots \sqcap \mathtt{symb}_n.\{x_n\}) \sqcap C' \sqsubseteq D'$$

where C' *and* D' *are obtained from* C *and* D*, respectively, by replacing each concept name* σ_i *by* $\exists \mathtt{type}.\{x_i\}$ *and each individual name* σ_j *by* x_j.

[1] Most complexity upper bounds for DLs above $\mathcal{ALCOIN}$ are only known to hold when assuming unary coding of numbers; our results inherit this restriction.

The template instance for α, *denoted* $\mathsf{tins}(\alpha)$, *is the following set of ABox assertions:*

$$\{A_\alpha(c_\alpha), \mathtt{symb}_1(c_\alpha, c_{\sigma_1}), \ldots, \mathtt{symb}_n(c_\alpha, c_{\sigma_n})\}$$

where c_α *and* $c_{\sigma_1}, \ldots, c_{\sigma_n}$ *are fresh individual names.*

We extend temp *and* tins *to sets of axioms by taking the union of the results, and to ABox axioms* $A(c)$ *and* $r(c,d)$ *by expressing them as TBox axioms* $\{c\} \sqsubseteq A$ *and* $\{c\} \sqsubseteq \exists r.\{d\}$, *respectively. For an unbounded knowledge base* KB *with ABox* $\mathcal{A}$, *TBox* $\mathcal{T}$, *and RBox* $\mathcal{R}$, *we define* $\mathsf{tt}(\mathrm{KB})$ *as the set* $\{\top \sqsubseteq \exists \mathtt{gci}.\{x\}\} \cup \mathsf{temp}(\mathcal{A} \cup \mathcal{T}) \cup \mathsf{tins}(\mathcal{A} \cup \mathcal{T}) \cup \mathcal{R}$, *where we assume that the same individuals* c_σ *for representing a symbol* σ *in any translation.*

It is easy to verify the following correctness property, where the axiom $\top \sqsubseteq \exists \mathtt{gci}.\{x\}$ ensures that the template instantiation individuals c_α are universally reachable via gci.

Lemma 8. $\mathrm{KB} \models A(c)$ *iff* $\mathsf{tt}(\mathrm{KB}) \models \exists \mathtt{type}.\{c_A\}(c)$.

This yields a polynomial reduction from one reasoning problem to another. We can improve this to obtain a reduction where the size of the TBox of $\mathsf{tt}(\mathrm{KB})$ is not affected by the size of N_I, N_C, or the TBox of KB. Indeed, since $\mathsf{tmpl}(\alpha)$ contains only role names and variable names, we only need one template for each variant of α obtained by uniformly renaming concepts and individuals. In particular, if the given DL admits a finite number of normal forms, then it is enough to include one template for each axioms that can be obtained by uniform renaming of roles in some normal form. If N_R is fixed, this yields a fixed TBox and also a fixed RBox (since a fixed N_R only allows for a bounded number of RBox axioms in normal form). Summing up, all standard reasoning problems for a normalizable DL $\mathcal{L}$ that can be expressed with a fixed, finite set of role names N_R can be expressed in $\mathcal{LV}$ using a fixed, finite TBox and RBox.

Lemma 9. *Let* $\mathcal{L}[\mathsf{N}_R]$ *be the DL* $\mathcal{L}$ *restricted to signatures that use a fixed, finite set* N_R *of role names. If* $\mathcal{L}[\mathsf{N}_R]$ *is normalizable using only roles from* N_R, *and its combined complexity is hard for a complexity class* C *that includes* P, *then so is the data complexity of* $\mathcal{LV}$.

This allows us to show Theorem 7. Indeed, the known hardness proofs for Horn-$\mathcal{ALC}$ (Krötzsch, Rudolph, and Hitzler 2013) and $\mathcal{ALCOIF}$ (Tobies 2000) are based on constructing a sequence of knowledge bases that use a fixed, finite set of role names. Hence, the respective complexities are the claimed lower bounds for the data complexities of Horn-$\mathcal{ALCV}$ and $\mathcal{ALCIFV}$.

Interestingly, Lemma 9 is not applicable to $\mathcal{SROIQV}$, since the known N2ExpTime-hardness proof for $\mathcal{SROIQ}$ relies on an unbounded set of role names (Kazakov 2008). Indeed, it turns out that our N2ExpTime upper bound for the data complexity of $\mathcal{SROIQV}$ can be improved to NExpTime, which is tight according to Theorem 7. With a very similar argument one can establish an ExpTime upper bound for data complexity of Horn-$\mathcal{SROIQV}$.

Theorem 10. *The data complexity of* $\mathcal{SROIQV}$ *is in* NExpTime, *assuming unary coding of numbers. The data complexity of Horn-*$\mathcal{SROIQV}$ *is in* ExpTime.

Proof. In order to show our claim, we deploy a two-step reduction and analyze its effects on the knowledge base. Starting from the $\mathcal{SROIQV}$ knowledge base $\mathrm{KB} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$, we obtain the $\mathcal{SROIQ}$ knowledge base $\mathrm{KB}' = (\mathcal{A}, \mathcal{T}', \mathcal{R})$ by grounding KB. Next, we obtain the $\mathcal{ALCHOIQ}$ knowledge base $\mathrm{KB}'' = (\mathcal{A}, \mathcal{T}'', \emptyset)$ by applying the RBox removal transformation introduced by Kazakov (2008). Analyzing the grounding step, we find that $size(\mathcal{T}') \leq size(\mathcal{T}) + size(\mathcal{A})^{size(\mathcal{T})}$. Moreover, for the RBox removal step, we obtain $size(\mathcal{T}'') \leq 2^{size(\mathcal{R})} \cdot size(\mathcal{T}')$. Therefore we get $size(\mathrm{KB}'') = size(\mathcal{A}) + size(\mathcal{T}'') \leq size(\mathcal{A}) + 2^{size(\mathcal{R})} \cdot size(\mathcal{T}') \leq size(\mathcal{A}) + 2^{size(\mathcal{R})} \cdot (size(\mathcal{T}) + size(\mathcal{A})^{size(\mathcal{T})})$. Since $\mathcal{T}$ and $\mathcal{R}$ are fixed, we find that KB'' is polynomial in the size of $\mathcal{A}$. Since the combined complexity of standard reasoning in $\mathcal{ALCHOIQ}$ is in NExpTime for unary coding of numbers (Tobies 2001), we have proven our claim. The same size argument as above holds for the transformation of Horn-$\mathcal{SROIQV}$ into Horn-$\mathcal{SROIQ}$ (via grounding) and further into Horn-$\mathcal{ALCHOIF}b$, which has a combined complexity of ExpTime (Ortiz, Rudolph, and Simkus 2010). This shows the claim for Horn-$\mathcal{SROIQ}$. □

7 Combined Complexities

We already know the combined complexities of $\mathcal{ELV}$ (ExpTime, cf. Section 4) and $\mathcal{SROIQV}$ (N2ExpTime). The N2ExpTime argument for $\mathcal{SROIQV}$ (Krötzsch et al. 2011) can be directly applied to Horn-$\mathcal{SROIQV}$ (Ortiz, Rudolph, and Simkus 2010) to establish 2ExpTime combined complexity. From Theorem 1 and the known complexity results we further obtain that the upper bound for the combined complexity of $\mathcal{SHOIV}$, and $\mathcal{SHOQV}$ is 2ExpTime (Hladik 2004; Glimm, Horrocks, and Sattler 2008). The upper bounds Theorem 1 yields for $\mathcal{SHOIQV}$ and Horn-$\mathcal{SHOIQV}$ coincide with those of $\mathcal{SROIQV}$ and Horn-$\mathcal{SROIQV}$, respectively. It turns out that these bounds are tight: reasoning in DLs above Horn-$\mathcal{ALCV}$ is 2ExpTime-hard, and reasoning in DLs above $\mathcal{ALCIFV}$ is (co)N2ExpTime-hard.

To show these results, we employ a general technique of encoding "axiom templates" that can encode an exponential number of concept inclusion axioms in polynomial space. We call this expressive feature *GCI iterator* since it iterates over a list of indexed concept names to produce concrete axioms. For example, the GCI iterator $A[\mathtt{i}] \sqsubseteq A[\mathtt{i}+1]$ $[\mathtt{i} = 1, \ldots, n]$ encodes n axioms of the form $A_i \sqsubseteq A_{i+1}$. We will show that GCI iterators can be expressed polynomially using nominal schemas, even when assuming binary coding of the number n. This allows us to encode exponentially large knowledge bases using polynomially many axioms with nominal schemas. In particular, we will do this for the specific knowledge bases that have been used to show the known hardness results for Horn-$\mathcal{ALC}$ and $\mathcal{ALCOIF}$, thus boosting the complexity by one exponential.

Definition 8. *Consider a DL signature* $\langle \mathsf{N}_I, \mathsf{N}_C, \mathsf{N}_R \rangle$. *A* GCI iterator *over this signature is an expression*

$$C \sqsubseteq D \; [\mathtt{i} = 1, \ldots, n]$$

where $n \geq 1$ *and* $C \sqsubseteq D$ *is a general concept inclusion over*

$\langle \mathsf{N}_I, \mathsf{N}_C \cup \{A[1],\dots,A[n+1],A[\mathtt{i}],A[\mathtt{i}+1] \mid A \in \mathsf{N}_C\}, \mathsf{N}_R\rangle$. *Note that* $\mathtt{i}$ *is a literal part of the syntax, not a placeholder for a specific number. The additional concept names* $A[\dots]$ *are assumed to be distinct from all concepts in* N_C.

The expansion *of a GCI iterator is the set of GCIs over* $\langle \mathsf{N}_I, \mathsf{N}_C \cup \{A[1],\dots,A[n+1] \mid A \in \mathsf{N}_C\}, \mathsf{N}_R\rangle$ *obtained by replacing, for each* $i \in \{1,\dots,n\}$*, all concepts* $A[\mathtt{i}]$ *by* $A[i]$*, and all concepts* $A[\mathtt{i}+1]$ *by* $A[i+1]$.

For a DL $\mathcal{L}$*, we let* $\mathcal{L}_{\mathrm{GI}}$ *be* $\mathcal{L}$ *extended by GCI iterators as axioms. The semantics of an* $\mathcal{L}_{\mathrm{GI}}$ *knowledge base* KB *is given by the translation into* $\mathcal{L}$ *through replacing all GCI iterators by their expansions, denoted* $\mathsf{expand}(\mathrm{KB})$.

Note that, assuming binary encoding of the value n, GCI iterators allow for an exponentially more succinct representation of their expansion.

Next, we will show that standard reasoning in a DL $\mathcal{L}_{\mathrm{GI}}$ without nominal schemas can be polynomially reduced to standard reasoning in $\mathcal{LV}$. We restrict attention to the case where the upper bound n in all GCI iterators is of the form 2^ℓ for some $\ell \geq 1$. Arbitrary n could be supported with some additional effort, but this is not relevant for our proofs.

Definition 9. *Consider* $\ell \geq 1$*, role names* $\mathtt{bit}_1,\dots,\mathtt{bit}_\ell$*, and individual names* 0 *and* 1*. For a list* $b_1,\dots,b_\ell$ *with* $b_i \in \{0,1\} \cup \mathsf{N}_V$*, we define* $\mathit{Bits}(b_1,\dots,b_\ell) := \prod_{i=1}^{\ell} \exists \mathtt{bit}_i.\{b_i\}$*. Consider additional role names* $\mathtt{num}$*,* $\mathtt{this}$*, and* $\mathtt{next}$*.* $\mathrm{KB}_{\mathrm{num}}$ *consists of the following axioms, for all* $1 \leq i \leq \ell$*:*

$$\top \sqsubseteq \exists \mathtt{num}.\exists \mathtt{this}.\mathit{Bits}(0,\dots,0)$$
$$\exists \mathtt{this}.\mathit{Bits}(x_1,\dots,x_i,0,1,\dots,1) \equiv \exists \mathtt{next}.\mathit{Bits}(x_1,\dots,x_i,1,0,\dots,0)$$
$$\exists \mathtt{num}.\exists \mathtt{next}.\mathit{Bits}(x_1,\dots,x_\ell) \equiv \exists \mathtt{num}.\exists \mathtt{this}.\mathit{Bits}(x_1,\dots,x_\ell)$$

Given a GCI iterator γ *of the form* $C \sqsubseteq D \;\; [\mathtt{i}=1,\dots,2^\ell]$*, we define* $\mathsf{vexpand}(\gamma)$ *to be the axiom* $N \sqcap C' \sqsubseteq D'$*, where* $N = \exists \mathtt{num}.(\exists \mathtt{this}.\mathit{Bits}(x_1,\dots,x_\ell) \sqcap \exists \mathtt{next}.\mathit{Bits}(y_1,\dots,y_\ell))$ *and* C' *and* D' *are obtained from* C *and* D*, respectively, by the following replacements:*

- $A[\mathtt{i}]$ *is replaced by* $\exists \mathtt{type}.(A' \sqcap \mathit{Bits}(x_1,\dots,x_\ell))$*,*
- $A[\mathtt{i}+1]$ *is replaced by* $\exists \mathtt{type}.(A' \sqcap \mathit{Bits}(y_1,\dots,y_\ell))$*,*
- $A[k]$ *is replaced by* $\exists \mathtt{type}.(A' \sqcap \mathit{Bits}(b_1,\dots,b_\ell))$ *where* $b_1,\dots,b_\ell$ *is the binary encoding of* k*,*

where A' *is a fresh concept name for each* $A \in \mathsf{N}_C$.

For an unbounded $\mathcal{L}_{\mathrm{GI}}$ *knowledge base* $\mathrm{KB} = \langle \mathcal{A}, \mathcal{T}, \mathcal{R}\rangle$ *where all GCI iterators range from* 1 *to* 2^ℓ*, the* $\mathcal{LV}$ *knowledge base* $\mathsf{itov}(\mathrm{KB})$ *is obtained by adding the axioms* $\mathrm{KB}_{\mathrm{num}}$ *and replacing each GCI iterator* $\gamma \in \mathrm{KB}$ *with* $\mathsf{vexpand}(\gamma)$.

Theorem 11. *For any DL* $\mathcal{L}$*,* itov *provides a polytime reduction from standard reasoning in* $\mathcal{L}_{\mathrm{GI}}$ *to standard reasoning in* $\mathcal{LV}$*: for every unbounded* $\mathcal{L}_{\mathrm{GI}}$ *knowledge base* KB *and* $\mathcal{L}$ *axiom* $A(c)$*, we have* $\mathrm{KB} \models A(c)$ *iff* $\mathsf{itov}(\mathrm{KB}) \models \mathsf{itov}(A(c))$.

Proof. We write $\mathsf{bin}(k)$ for the list of ℓ bits of a number $k < 2^\ell$ in binary encoding. The minimal model for $\mathrm{KB}_{\mathrm{num}}$ is one where every individual has one $\mathtt{num}$-successor in concept $\exists \mathtt{this}.\mathit{Bits}(\mathsf{bin}(k))$ for each number $0 \leq k < 2^\ell$. Moreover, if $k < 2^\ell - 1$ (i.e., $\mathsf{bin}(k)$ contains a 0), the corresponding successor also is in the class $\exists \mathtt{next}.\mathit{Bits}(\mathsf{bin}(k+1))$.

Now every model $\mathcal{I} \models \mathsf{expand}(\mathrm{KB})$ with $\mathcal{I} \not\models A(c)$ can be transformed into a model $\mathcal{I}' \models \mathsf{itov}(\mathrm{KB})$ with $\mathcal{I}' \not\models \mathsf{itov}(A(c))$. By unboundedness, we can assume that the domain of $\mathcal{I}$ contains infinitely many individuals. We can thus find distinct individuals to represent bits and numbers, and we interpret $\mathtt{num}$, $\mathtt{this}$, $\mathtt{next}$, and $\mathtt{bit}_i$ according to the minimal model. Moreover, for all $A \in \mathsf{N}_C$ and $0 \leq k < 2^\ell$, we add an element $\delta_{A[k]}$ with $\delta_A \in (A' \sqcap \mathit{Bits}(\mathsf{bin}(k)))^{\mathcal{I}'}$, and we set $\langle \delta, \delta_{A[k]}\rangle \in \mathtt{type}$ iff $\delta \in A[k]^{\mathcal{I}}$. Roles and concepts that are not of the form $B[k]$ are interpreted as in $\mathcal{I}$. It is easy to see that $\mathcal{I}'$ has the claimed properties. For the converse, note that the minimal model of $\mathrm{KB}_{\mathrm{num}}$ is indeed found in every model of $\mathsf{itov}(\mathrm{KB})$. Thus, if $\mathcal{I}' \models \mathsf{itov}(\mathrm{KB})$ with $\mathcal{I}' \not\models \mathsf{itov}(A(c))$, then there is more general interpretation $\mathcal{I}'' \models \mathsf{itov}(\mathrm{KB})$ with $\mathcal{I}'' \not\models \mathsf{itov}(A(c))$ where $\mathrm{KB}_{\mathrm{num}}$ is interpreted as in the minimal model. Using similar correspondences as before, we can construct a suitable model $\mathcal{I} \models \mathsf{expand}(\mathrm{KB})$ with $\mathcal{I} \not\models A(c)$. □

We now use Theorem 11 to show additional hardness results by using GCI iterators to re-encode known hardness proofs for exponentially large knowledge bases.

Theorem 12. *The combined complexity of Horn-*$\mathcal{ALCV}$ *is* 2ExpTime*-hard.*

Proof. The known ExpTime-hardness result for Horn-$\mathcal{ALC}$ is based on reducing the problem of deciding if a word w is accepted by an alternating Turing machine (ATM) in polynomial space to the problem if a Horn-$\mathcal{ALC}$ knowledge base entails a GCI, where both knowledge base and GCI are of polynomial size w.r.t. $|w|$ (Krötzsch, Rudolph, and Hitzler 2013). One can generalize this construction to arbitrary (possibly non-polynomial) space bounds s, leading to knowledge base of size polynomial in s. We modify the encoding to be able to express this knowledge base in Horn-$\mathcal{ALC}_{\mathrm{GI}}$, so that the resulting knowledge base is logarithmic in the size of s. Thus, we can simulate exponential space ATM runs with a polynomial Horn-$\mathcal{ALC}_{\mathrm{GI}}$ knowledge base.

Let s be an arbitrary space bound. To encode an s-space ATM $\langle Q, \Sigma, \Delta, q_0\rangle$, the original encoding uses the following concept names: A_q for $q \in Q$ (ATM is in state q), H_i for $i = 1,\dots,s$ (ATM head is at position i), $C_{\sigma,i}$ with $\sigma \in \Sigma$ and $i = 1,\dots,s$ (tape position i contains symbol σ), and A (ATM accepts this configuration). Accordingly, we use concepts A_q, $H[i]$, $C_\sigma[i]$, and A. Roles S_δ encode δ-successor configurations. The original encoding of ATM transitions $\delta = \langle q, \sigma, q', \sigma', r\rangle$ can then be expressed as follows:

$$A_q \sqcap H[\mathtt{i}] \sqcap C_\sigma[\mathtt{i}] \sqsubseteq \exists S_\delta.(A_{q'} \sqcap H[\mathtt{i}+1] \sqcap C'_\sigma[\mathtt{i}]) \; [\mathtt{i}=1,\dots,s]$$

Transitions with leftwards head movement are expressed analogously. To apply Theorem 11, we use the same range for all GCI iterators. This has the side effect that the tape has cells $1,\dots,s+1$, while $\mathtt{i}$ ranges over $1,\dots,s$ and $\mathtt{i}+1$ ranges over $2,\dots,s+1$; thus, when using $\mathtt{i}$ in axioms that refer to all tape positions, we generally assume that there is also one one more axiom for covering $s+1$ as well.

The accepting conditions are modelled by $A_q \sqcap \exists S_\delta.A \sqsubseteq A$ (for existential states q) and $A_q \sqcap H[\mathtt{i}] \sqcap C_\sigma[\mathtt{i}] \sqcap \bigsqcap_{\delta \in \bar{\Delta}} (\exists S_\delta.A) \sqsubseteq A$ $[\mathtt{i}=1,\dots,s]$ (for universal states q with

possible transitions $\tilde{\Delta}$ for symbol σ). To ensure preservation of the ATM memory between configurations, the original proof uses axioms $H_j \sqcap C_{\sigma,i} \sqsubseteq \forall S_\delta.C_{\sigma,i}$ for all $i \neq j$. This cannot be expressed with GCI iterators. Instead, we introduce new concepts $L[i]$ (cell i is left of the head) and $R[i]$ (cell i is right of the head), which we can axiomatize using GCI iterators $H[\mathtt{i}] \sqsubseteq R[\mathtt{i}+1]$, $R[\mathtt{i}] \sqsubseteq R[\mathtt{i}+1]$, $H[\mathtt{i}+1] \sqsubseteq L[\mathtt{i}]$, and $L[\mathtt{i}+1] \sqsubseteq L[\mathtt{i}]$, all with the same range as before. The rules of memory preservation can now be expressed with the GCI iterators $L[\mathtt{i}] \sqcap C_\sigma[\mathtt{i}] \sqsubseteq \forall S_\delta.C_\sigma[\mathtt{i}]$ and $R[\mathtt{i}] \sqcap C_\sigma[\mathtt{i}] \sqsubseteq \forall S_\delta.C_\sigma[\mathtt{i}]$, for all $\delta \in \Delta$ and $\sigma \in \Sigma$.

Finally, the initialization of the tape in the original proof simply defines the tape contents for each position. To initialize the $s-|w|$ many blank cells after the input word, we use concepts $B[i]$ and axioms $B[\mathtt{i}] \sqsubseteq B[\mathtt{i}+1]$ and $B[\mathtt{i}] \sqsubseteq C_\square[\mathtt{i}]$. The initial configuration is now described with the concept

$$I_w := A_{q_0} \sqcap H_1 \sqcap C_{\sigma_1}[1] \sqcap \ldots \sqcap C_{\sigma_{|w|}}[|w|] \sqcap B[|w|+1].$$

Let $\mathrm{KB}_{\mathcal{M},w}$ be the knowledge base that contains all of the above GCIs. It is a direct consequence from (Krötzsch, Rudolph, and Hitzler 2013) that $\mathrm{KB}_{\mathcal{M},w} \models I_w \sqsubseteq A$ if the ATM $\mathcal{M}$ accepts w in space s. Since GCI iterators can encode the range $[\mathtt{i}{=}1,\ldots,s]$ in space logarithmic in s, reasoning with a polynomially large $\mathrm{KB}_{\mathcal{M},w}$ can solve the word problem of exponential space ATMs, which is 2ExpTime-hard. The claim thus follows from Theorem 11. □

Theorem 13. *The combined complexity of $\mathcal{ALCIFV}$ is* (co)N2ExpTime*-hard.*

Proof. We proceed with the case of $\mathcal{ALCOIF}$. We build on the proof of Tobies (2000), which reduced the tiling problem for exponential grids to $\mathcal{ALCOIF}$ consistency checking. To construct an exponential grid, one encodes coordinates of tiles in binary numbers, represented by concept names $X_1,\ldots,X_\ell$ (horizontal position) and $Y_1,\ldots,Y_\ell$ (vertical position), respectively. We will express these by indexed concepts $X[i]$ and $Y[i]$. Horizontal neighbours are connected by a role h, vertical ones by a role v. Functionality restrictions and inverse functionality restrictions on h and v ensure uniqueness of neighbours. The concept with the maximal coordinates (right upper corner) is subsumed by a nominal, ensuring that the potentially diverging structure of h and v successors collapses into one grid.

We need to express that the encoded position of the right (and upper) neighbour is the successor. For numbers of linear length, this can be done as in Definition 9; to allow for numbers of exponential length, we use auxiliary concepts $\hat{X}[i]$ to express that all bits up to position $i-1$ are set to true, i.e., that the conjunction $\prod_{j=1}^{i-1} X[j]$ holds. $\hat{Y}[i]$ plays the same role for vertical positions. The propagation of X-coordinates can now be expressed with the following GCI iterators, all of which we assume to have range $[\mathtt{i}=1,\ldots,\ell]$ for some ℓ:

$$\top \sqsubseteq \exists h.\top \sqcap \leqslant 1\, h.\top \sqcap \leqslant 1\, h^-.\top \tag{7}$$

$$\top \sqsubseteq \hat{X}[1] \qquad \hat{X}[\mathtt{i}] \sqcap X[\mathtt{i}] \equiv \hat{X}[\mathtt{i}+1] \tag{8}$$

$$\hat{X}[\mathtt{i}] \sqsubseteq (X[\mathtt{i}] \sqcap \forall h.\neg X[\mathtt{i}]) \sqcup (\neg X[\mathtt{i}] \sqcap \forall h.X[\mathtt{i}]) \tag{9}$$

$$\neg\hat{X}[\mathtt{i}] \sqsubseteq (X[\mathtt{i}] \sqcap \forall h.X[\mathtt{i}]) \sqcup (\neg X[\mathtt{i}] \sqcap \forall h.\neg X[\mathtt{i}]) \tag{10}$$

$$X[\mathtt{i}] \sqsubseteq \forall v.X[\mathtt{i}] \qquad \neg X[\mathtt{i}] \sqsubseteq \forall v.\neg X[\mathtt{i}] \tag{11}$$

Axiom (7) ensures a chain of h successors. Axioms (8) express the relation of $\hat{X}[i]$ and $X[i]$ explained above. Axioms (9) and (10) characterize the binary increment of numbers. Axioms (11) ensure that horizontal positions are preserved among vertical neighbours. We assume the same formulation for the vertical propagation, swapping h with v and X with Y, respectively. Note that $\mathtt{i}+1$ only occurs in (8); indeed, the bits $X[\ell+1]$ and $Y[\ell+1]$, which exist due to our uniform choice of range, have no significance in our modelling. To "start" the grid, we assert that there is a lower left corner: $\{a\} \sqsubseteq \neg X[i] \sqcap \neg Y[i]$. To "close" the grid, we add a nominal to the upper right corner: $\hat{X}[\ell+1] \sqcap \hat{Y}[\ell+1] \sqsubseteq \{o\}$. Note that the grid will continue to have successors beyond this point, due to axiom (7), but these do not need to form a grid and will not affect whether the tiling problem has a solution (one can always ensure that there are tiles to continue in non-grid structures only).

Formally, a tiling problem is given by a set of tiles $\mathcal{D} = \{D_1,\ldots,D_n\}$ together with horizontal and vertical compatibility relations $V,H \subseteq \mathcal{D} \times \mathcal{D}$. Using concept names D_i (which are not iterated over), we can axiomatize correct tilings by the axioms $\top \sqsubseteq \bigsqcup_{1\leq i\leq d} \left(D_i \sqcap \prod_{j\neq i} \neg D_j\right)$, $D_i \sqsubseteq \forall v. \bigsqcup_{\langle D_i,D_j\rangle \in V} D_j$, and $D_i \sqsubseteq \forall h. \bigsqcup_{\langle D_i,D_j\rangle \in H} D_j$. It is easy to see that this construction reduces the tiling problem on a grid of size $2^\ell \times 2^\ell$ to checking the consistency of an $\mathcal{ALCOIF}_{\mathrm{GI}}$ knowledge base of size polynomial in $\log(\ell)$ (since ℓ is encoded in binary in GCI iterators). Since the double-exponential tiling problem is N2ExpTime-hard, the claim follows from Theorem 11. □

Proof of Theorem 4. As the number of nominal schemas in normalized axioms is bounded, a normalized knowledge base can be grounded polynomially. Thus, the combined complexity of any normalizable DL $\mathcal{LV}$ coincides with the combined complexity of $\mathcal{L}$. It is known that P $\subsetneq$ ExpTime $\subsetneq$ 2ExpTime and that NExpTime $\subsetneq$ N2ExpTime. Together with our previous results, this shows the claim. □

8 An Alternative Semantics

As discussed in Section 3, our semantics of nominal schemas depends on the given set N_I of individual names. If we allow N_I to be empty, then this can even affect the entailment of standard DL axioms. For example, $\top \sqsubseteq \exists r.\{x\} \not\models \top \sqsubseteq \exists r.\top$ if $\mathsf{N}_I = \emptyset$. However, in most cases, only entailments of axioms with nominal schemas are affected, as discussed for (1) in Section 3. Such a dependence on the set of "known" symbols might be considered to be in conflict with the open-world assumption. The ontology language OWL considers a potentially infinite signature of URIs (W3C OWL Working Group 27 October 2009).[2]

To address this problem, one could modify the semantics of nominal schemas by assuming the set N_I to be infinite and fixed. A severe practical consequence of this suggestion is that $\mathsf{ground}(\alpha)$ and $\mathsf{ground}(\mathrm{KB})$ become infinite if they contain any nominal schemas. While this does not pose problems for any of the definitions as such, we seem to lose

[2] On the other hand, OWL also provides ways to explicitly declare a finite set of signature symbols in an ontology.

the upper complexity bounds of Theorem 1. However, we will show that the alternative semantics does not actually affect reasoning complexities, and in fact agrees with the standard semantics in many cases.

In the rest of this section, we assume N_I to be infinite, and use $\bar{\mathsf{N}}$ for a set $\bar{\mathsf{N}} \subseteq \mathsf{N}_I$ that corresponds to our earlier finite signature. We use $\models_{\bar{\mathsf{N}}}$ to denote standard entailment w.r.t. $\bar{\mathsf{N}}$, and $\models_\infty$ to denote entailment w.r.t. the infinite N_I.

Theorem 14. *For any standard DL $\mathcal{L}$, any $\mathcal{LV}$ knowledge base* KB*, any $\mathcal{LV}$ axiom and any finite set* $\bar{\mathsf{N}}$ *which contains more individual names than the sum of the number of individual names used in* $\mathrm{KB} \cup \alpha$ *and the number of variable names in α, we have* $\mathrm{KB} \models_{\bar{\mathsf{N}}} \alpha$ *iff* $\mathrm{KB} \models_\infty \alpha$.

Proof. "If." Assume $\mathrm{KB} \models_\infty \alpha$. We pick a function $\psi : \mathsf{N}_I \to \bar{\mathsf{N}}$ such that $\psi(a) = a$ for all $a \in \bar{\mathsf{N}}$. Note that such a ψ always exists since $\bar{\mathsf{N}}$ is nonempty by assumption. Lifting ψ to concept expressions, axioms and sets of axioms in the usual way, we observe that $\psi(\mathsf{ground}_\infty(\mathrm{KB})) = \mathsf{ground}_{\bar{\mathsf{N}}}(\mathrm{KB})$, as well as $\psi(\mathsf{ground}_\infty(\alpha)) = \mathsf{ground}_{\bar{\mathsf{N}}}(\alpha)$. Moreover, from $\mathsf{ground}_\infty(\mathrm{KB}) \models \beta$ follows that $\psi(\mathsf{ground}_\infty(\mathrm{KB})) \models \psi(\beta)$ for all $\beta \in \psi(\mathsf{ground}_\infty(\alpha))$. Hence we obtain $\mathsf{ground}_{\bar{\mathsf{N}}}(\mathrm{KB}) \models \mathsf{ground}_{\bar{\mathsf{N}}}(\alpha)$ and therefore $\mathrm{KB} \models_{\bar{\mathsf{N}}} \alpha$.

"Only if." Assume $\mathrm{KB} \models_{\bar{\mathsf{N}}} \alpha$. Let $V = \{x_1, \ldots x_n\}$ be the set of nominal variables occurring in α. Then, by assumption, we find n distinct individuals $\{a_1, \ldots, a_n\} \subseteq \bar{\mathsf{N}}$ not occurring in $\mathrm{KB} \cup \alpha$. Let $\alpha' = \alpha[x_1/a_1, \ldots, x_n/a_n]$. Clearly $\mathrm{KB} \models_{\bar{\mathsf{N}}} \alpha'$. Then also $\mathrm{KB} \models_\infty \alpha'$, since the grounding of KB with individuals from $\bar{\mathsf{N}}$ is contained in the grounding of KB with all individuals. In order to show $\mathsf{ground}_\infty(\mathrm{KB}) \models \alpha$, we will show that $\mathsf{ground}_\infty(\mathrm{KB}) \models \beta$ for all $\beta \in \mathsf{ground}_\infty(\alpha)$. So, given such a β and the variable assignment *val* of $V = \{x_1, \ldots x_n\}$ to N_I that produces β from α, we pick a surjective mapping $\varphi : \mathsf{N}_I \to \mathsf{N}_I$ such that $\varphi(a) = a$ for every individual name mentioned in $\mathrm{KB} \cup \alpha$ and $\varphi(a_i) = \mathit{val}(x_i)$ (this is possible since N_I is infinite). Lifting φ to concept expressions, axioms and sets of axioms in the usual way, it is obvious that $\mathrm{KB} \models_\infty \alpha' \implies \mathsf{ground}_\infty(\mathrm{KB}) \models \alpha' \implies \varphi(\mathsf{ground}_\infty(\mathrm{KB})) \models \varphi(\alpha') \implies \mathsf{ground}_\infty(\mathrm{KB}) \models \varphi(\alpha') \implies \mathsf{ground}_\infty(\mathrm{KB}) \models \beta \implies \mathrm{KB} \models_\infty \beta$. Note that $\varphi(\mathsf{ground}_\infty(\mathrm{KB})) = \mathsf{ground}_\infty(\mathrm{KB})$ by construction. □

The previous proof requires that the unique name assumption (UNA) is not made, as is standard for all DLs considered in this paper. Moreover, imposing the unique name assumption in our setting would enforce all models to have an infinite domain, which makes domain-restricting axioms such as $\top \sqsubseteq \{a, b, c\}$ unsatisfiable.

Theorem 14 asserts that the two semantics coincide whenever the finite individual set is "large enough" for the considered knowledge base and tested consequence. For the special case of entailment of axioms without nominal schemas, it suffices if there is at least one individual name.

Corollary 15. *For any standard DL $\mathcal{L}$, $\mathcal{LV}$-reasoning complexities (both data and combined) for $\models$ and $\models_\infty$ coincide.*

We have shown that entailment checking w.r.t. $\models_\infty$ can be reduced to entailment checking w.r.t. $\models_{\bar{\mathsf{N}}}$ for a suitable $\bar{\mathsf{N}}$. On the other hand, any such reasoning task w.r.t. $\models_{\bar{\mathsf{N}}}$ can be reduced to reasoning w.r.t. $\models_\infty$ if disjunction is available, since $\mathrm{KB} \models_{\bar{\mathsf{N}}} \alpha$ iff $\mathrm{KB} \cup \{\{x\} \sqsubseteq \bigsqcup_{a \in \bar{\mathsf{N}}} \{a\}\} \models_\infty \alpha$.

The alternative semantics also comes handy when analyzing expressivity, since one cannot finitely express nominal schemas using grounding. We can use this to show that $\mathcal{LV}$ is generally more expressive than $\mathcal{L}$ extended with DL-safe rules. Consider the axiom

$$\{x\} \sqsubseteq \exists r.\exists r.\exists r.\{x\}. \tag{12}$$

Now, suppose there exists a $\mathcal{SROIQ}$ knowledge base KB and a set RB of DL-safe rules (where w.l.o.g. all used concept expressions are concept names) such that $\mathrm{KB} \cup \mathrm{RB}$ is a conservative extension of the above axiom. Now we pick an individual name a not mentioned in $\mathrm{KB} \cup \mathrm{RB}$ and define two interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ as follows: $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}} = \{b_1, b_2, b_3 \mid b \in \mathsf{N}_I\} \cup \mathbb{N}$, $b^{\mathcal{I}} = b^{\mathcal{J}} = b_1$ for all $b \in \mathsf{N}_I$, $r^{\mathcal{I}} = \{(b_1, b_2), (b_2, b_3), (b_3, b_1) \mid b \in \mathsf{N}_I\}$, and $r^{\mathcal{J}} = r^{\mathcal{I}} \setminus \{(a_1, a_2), (a_2, a_3), (a_3, a_1)\}$. Note that $\mathcal{I}$ is a model of the above axiom while $\mathcal{J}$ is not. Consequently, by assumption there must be an extension $\mathcal{I}'$ of $\mathcal{I}$ satisfying $\mathrm{KB} \cup \mathrm{RB} \models \mathcal{I}'$ but any extension $\mathcal{J}'$ of $\mathcal{J}$ must not be a model of $\mathrm{KB} \cup \mathrm{RB}$. Now we pick a bijective function $\varphi : \mathsf{N}_I \setminus \{a\} \to \mathsf{N}_I$ with $\varphi(b) = b$ for all b mentioned in $\mathrm{KB} \cup \mathrm{RB}$ and use it to define a renaming ρ of $\Delta^{\mathcal{I}'}$ as follows: $\rho(b_i) := c_i$ with $c = \varphi(b)$ for all $b \in \mathsf{N}_I \setminus \{a\}$ and $i \in \{1, 2, 3\}$, $\rho(a_i) := i$ for all $i \in \{1, 2, 3\}$ and $\rho(i) := i + 3$ for all $i \in \mathbb{N}$. Now let $\mathcal{K}$ be the interpretation obtained from $\mathcal{I}'$ via the renaming ρ. It is easy to check that $\mathcal{K}$ is an extension of $\mathcal{J}$. We also find that $\mathcal{K}$ must be a model of KB since $\mathcal{K}$ and $\mathcal{I}'$ are isomorphic as far as the used signature is concerned. Moreover, $\mathcal{K}$ is a model of RB since assuming, for a rule $R \in \mathrm{RB}$, a violating variable assignment *var*, $\varphi^- \circ \mathit{var}$ would be a violating assignment for $\mathcal{I}'$. Thus we have found an extension of $\mathcal{J}$ that is a model of $\mathrm{KB} \cup \mathrm{RB}$, which contradicts our initial assumption. Therefore, the above axiom cannot be expressed by $\mathcal{SROIQ}$ extended by DL-safe rules.

9 Conclusion and Future Work

We have clarified the data and combined complexities for nominal-schema-extensions of all DLs between $\mathcal{EL}$ and $\mathcal{SROIQ}$. The techniques developed for this purpose (TBox-to-ABox internalization and GCI iterators) are generic and directly applicable to DLs outside this range. For example, it is an obvious consequence of our findings and the results from (Rudolph, Krötzsch, and Hitzler 2008b) that adding arbitrary Boolean role constructors on simple roles to any DL above $\mathcal{ALCIFV}$ does not increase data or combined complexity; the same holds if conjunctions on simple roles are added to $\mathcal{ELV}^{++}$. Adding the concept product $\times$ (Rudolph, Krötzsch, and Hitzler 2008a) does not impact complexities either. Likewise, it is trivial to establish ExpTime completeness (data) and 2ExpTime completeness (combined) for $\mathcal{ZOQV}$ and $\mathcal{ZOIV}$ (Calvanese, Eiter, and Ortiz 2009).

The most obvious avenue for future work is the investigation of the complexities for conjunctive query answering.

Acknowledgement This work was supported by the DFG in project DIAMOND (Emmy Noether grant KR 4381/1-1).

References

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In Kaelbling, L., and Saffiotti, A., eds., *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, 364–369. Professional Book Center.

Brewka, G., and Lang, J., eds. 2008. *Proc. 11th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'08)*. AAAI Press.

Calvanese, D.; Eiter, T.; and Ortiz, M. 2009. Regular path queries in expressive description logics with nominals. In Boutilier, C., ed., *Proc. 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09)*, 714–720. IJCAI.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys* 33(3):374–425.

Glimm, B.; Horrocks, I.; and Sattler, U. 2008. Unions of conjunctive queries in $\mathcal{SHOQ}$. In Brewka and Lang (2008), 252–262.

Glimm, B.; Rudolph, S.; and Völker, J. 2010. Integrated metamodeling and diagnosis in OWL 2. In Patel-Schneider, P. F.; Pan, Y.; Glimm, B.; Hitzler, P.; Mika, P.; Pan, J.; and Horrocks, I., eds., *Proc. 9th Int. Semantic Web Conf. (ISWC'10)*, volume 6496 of *LNCS*, 257–272. Springer.

Hladik, J. 2004. A tableau system for the description logic $\mathcal{SHIO}$. In Sattler, U., ed., *Contributions to the Doctoral Programme of the 2ndInt. Joint Conf. on Automated Reasoning (IJCAR'04)*, volume 106 of *CEUR WS Proceedings*. CEUR-WS.org.

Horrocks, I.; Kutz, O.; and Sattler, U. 2006. The even more irresistible $\mathcal{SROIQ}$. In Doherty, P.; Mylopoulos, J.; and Welty, C. A., eds., *Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06)*, 57–67. AAAI Press.

Kazakov, Y. 2008. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In Brewka and Lang (2008), 274–284.

Knorr, M.; Hitzler, P.; and Maier, F. 2012. Reconciling OWL and non-monotonic rules for the Semantic Web. In Raedt, L. D.; Bessière, C.; Dubois, D.; Doherty, P.; Frasconi, P.; Heintz, F.; and Lucas, P. J. F., eds., *Proc. 20th European Conf. on Artificial Intelligence (ECAI'12)*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, 474–479. IOS Press.

Kolovski, V.; Parsia, B.; and Sirin, E. 2006. Extending the $\mathcal{SHOIQ}(d)$ tableaux with DL-safe rules: First results. In Parsia, B.; Sattler, U.; and Toman, D., eds., *Proc. 19th Int. Workshop on Description Logics (DL'06)*, volume 198 of *CEUR WS Proceedings*. CEUR-WS.org.

Krisnadhi, A., and Hitzler, P. 2012. A tableau algorithm for description logics with nominal schema. In Krötzsch, M., and Straccia, U., eds., *Proc. 6th Int. Conf. on Web Reasoning and Rule Systems (RR 2012)*, volume 7497 of *LNCS*, 234–237. Springer.

Krötzsch, M.; Maier, F.; Krisnadhi, A. A.; and Hitzler, P. 2011. A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In *Proc. 20th Int. Conf. on World Wide Web (WWW'11)*, 645–654. ACM.

Krötzsch, M.; Rudolph, S.; and Hitzler, P. 2013. Complexities of Horn description logics. *ACM Trans. Comput. Logic* 14(1):2:1–2:36.

Krötzsch, M.; Simančík, F.; and Horrocks, I. 2012. A description logic primer. *CoRR* abs/1201.4089.

Martínez, D. C.; Wang, C.; and Hitzler, P. 2013. Towards an efficient algorithm to reason over description logics extended with nominal schemas. In Faber, W., and Lembo, D., eds., *Proc. 7th Int. Conf. on Web Reasoning and Rule Systems (RR 2013)*, volume 7994 of *LNCS*, 65–79. Springer.

Mehdi, A., and Rudolph, S. 2011. Revisiting semantics for epistemic extensions of description logics. In Burgard, W., and Roth, D., eds., *Proc. 25th AAAI Conf. on Artificial Intelligence (AAAI'11)*. AAAI Press.

Motik, B.; Sattler, U.; and Studer, R. 2005. Query answering for OWL DL with rules. *J. of Web Semantics* 3(1):41–60.

Motik, B.; Shearer, R.; and Horrocks, I. 2009. Hypertableau reasoning for description logics. *J. of Artificial Intelligence Research* 36:165–228.

Ortiz, M.; Rudolph, S.; and Simkus, M. 2010. Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In Lin, F.; Sattler, U.; and Truszczynski, M., eds., *Proc. 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'10)*, 269–279. AAAI Press.

Rudolph, S.; Krötzsch, M.; and Hitzler, P. 2008a. All elephants are bigger than all mice. In Baader, F.; Lutz, C.; and Motik, B., eds., *Proc. 21st Int. Workshop on Description Logics (DL'08)*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Rudolph, S.; Krötzsch, M.; and Hitzler, P. 2008b. Cheap Boolean role constructors for description logics. In Hölldobler, S.; Lutz, C.; and Wansing, H., eds., *Proc. 11th European Conf. on Logics in Artificial Intelligence (JELIA'08)*, volume 5293 of *LNAI*, 362–374. Springer.

Steigmiller, A.; Glimm, B.; and Liebig, T. 2013. Nominal schema absorption. In Rossi, F., ed., *Proc. 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI'13)*, 1104–1110. AAAI Press/IJCAI.

Tobies, S. 2000. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research* 12:199–217.

Tobies, S. 2001. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. Ph.D. Dissertation, RWTH Aachen, Germany.

W3C OWL Working Group. 27 October 2009. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation. Available at http://www.w3.org/TR/owl2-overview/.

Wang, C., and Hitzler, P. 2013. A resolution procedure for description logics with nominal schemas. In Takeda, H.; Qu, Y.; Mizoguchi, R.; and Kitamura, Y., eds., *Proc. 2nd Joint Int. Conf. on Semantic Technology (JIST'12)*, volume 7774 of *LNCS*, 1–16. Springer.

Practical Uniform Interpolation and Forgetting for $\mathcal{ALC}$ TBoxes with Applications to Logical Difference *

Michel Ludwig
Institute for Theoretical Computer Science
TU Dresden, Germany
michel@tcs.inf.tu-dresden.de

Boris Konev
Department of Computer Science
University of Liverpool, United Kingdom
konev@liverpool.ac.uk

Abstract

We develop a clausal resolution-based approach for computing uniform interpolants of TBoxes formulated in the description logic $\mathcal{ALC}$ when such uniform interpolants exist. We also present an experimental evaluation of our approach and of its application to the logical difference problem for real-life $\mathcal{ALC}$ ontologies. Our results indicate that in many practical cases uniform interpolants exist and that they can be computed with the presented algorithm.

Introduction

Ontologies or *TBoxes* expressed in Description Logics (DL) provide a common vocabulary for a domain of interest together with a description of the meaning of the terms built from the vocabulary and of the relationships between them. Modern applications of ontologies, especially in the biological, medical, or healthcare domain, often demand large and complex ontologies; for example, the National Cancer Institute ontology (NCI) consists of more than 60 000 term definitions. For developing, maintaining, and deploying such large-scale ontologies it can be advantageous for ontology engineers to concentrate on specific parts of an ontology and ignore or *forget* the rest. Ignoring parts of an ontology can be formalised with the help of *predicate forgetting* and its dual *uniform interpolation*, which have both been extensively studied in the AI and DL literature (ten Cate et al. 2006; Eiter et al. 2006; Herzig and Mengin 2008; Konev, Walther, and Wolter 2009; Wang et al. 2008; 2010; 2012; Lutz and Wolter 2011).

Forgetting parts of an ontology can be used, for example, in the following practical scenarios. *Exhibiting hidden relations*: in addition to the explicitly stated connections between terms, additional relations can also be derived from ontologies with the help of reasoners. Such inferred connections are often harder to understand or debug. By forgetting everything but a handful of terms of interest, it then becomes possible to exhibit inferred relations that were hidden initially, potentially simplifying the understanding of the ontology structure. *Ontology obfuscation*: in software engineering, obfuscation (Collberg, Thomborson, and Low 1998) transforms a given program into a functionally equivalent one that is more difficult to read and understand for humans for the purpose of preventing reverse engineering. Forgetting can provide a similar function in the context of ontology engineering. Terms are often defined with the help of auxiliary terms which give structure to TBox inclusions. However, such a structure might be considered proprietary knowledge that should not be exposed, or it could simply be of little interest for ontology users. By forgetting these intermediate auxiliary terms, we obtain an ontology that is functionally equivalent, yet harder to read, understand, and modify by humans. Further applications of forgetting can be found in (Konev, Walther, and Wolter 2009; Lutz, Seylan, and Wolter 2012).

A promising and important application area of forgetting is the computation of the *logical difference* between ontology versions. Determining whether two versions of a document have differences is a standard task in information technology, and finding differences is particularly relevant for text processing and software development. Already in these areas, it is important to be able to identify which changes are *significant* and which are not (e.g., a software developer might want to ignore changes in the formatting style of the code such as the number of indentation spaces). Detecting significant changes is even more important in the setting of Knowledge Representation, where differences in the knowledge captured by ontologies are often more relevant than syntactic changes. Arguably, one of the most important concerns of an ontology engineer when modifying an existing ontology is to ensure that the introduced changes do not interfere with the meaning of the terms outside the fragment under consideration. Notice that neither the version comparison based on the syntactic form of the documents representing ontologies (Conradi and Westfechtel 1998) nor methods based on the structural transformations of ontology statements (Noy and Musen 2002; Klein et al. 2002; Jiménez-Ruiz et al. 2011) can be used to identify changes to the logical meaning of terms in every situation. However, such a correctness guarantee can be achieved by checking the equivalence of the ontologies resulting from forgetting the terms under consideration before and after the changes occurred.

*Partially supported by the DFG within the Cluster of Excellence 'Center for Advancing Electronics Dresden' (cfAED) and by the EPSRC under the grant EP/H043594/1.
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In this paper we develop an algorithm based on clausal resolution for computing uniform interpolants of TBoxes formulated in the description logic $\mathcal{ALC}$ which can preserve all the consequences which do not make use of some given *concept names* and which have a quantifier nesting level limited to some given depth. Subsequently, we present an experimental evaluation of our approach which demonstrates that in many practical cases uniform interpolants (without limitations on the quantifier nesting depth) exist and that they can be computed with our algorithm. We also apply our prototype tool to compute the logical difference between versions of ontologies from the biomedical domain.

The material presented here extends earlier work (Ludwig and Konev 2013). All the missing proofs can be found in the full version of this paper.[1]

Related work. Until recently research on uniform interpolation and forgetting in the setting of DL mainly has concentrated on theoretical foundations of forgetting. This could be partly explained by the high computational complexity of this task and by the fact that uniform interpolants do not always exist. The notion of forgetting has been introduced by Reiter and Lin (1994). Konev, Walther, and Wolter (2009) prove tractability of uniform interpolation for $\mathcal{EL}$ TBoxes of a specific syntactic form. Uniform interpolation for general $\mathcal{EL}$-TBoxes has been investigated by Nikitina and Rudolph (2012).

Wang et al. (2008; 2010; 2012) have developed algorithms for forgetting in expressive description logics. A tight 2-ExpTime-complete bound on the complexity for deciding the existence of a Σ-uniform interpolant in $\mathcal{ALC}$ and a worst-case triple-exponential procedure for computing a Σ-uniform interpolant if it exists, have been given by Lutz and Wolter (2011). Koopmann and Schmidt (2013) have introduced a two-stage resolution-based algorithm for computing uniform interpolants. As outcome of the first stage, a representation of the uniform interpolant in a description logic with fixpoint operators is computed (such a representation always exists) Then in the second stage an attempt is made to eliminate the newly-introduced fixpoints (which may not succeed). In contrast to this approach, our algorithm has one stage and it can be guaranteed that a uniform interpolant of bounded depth is returned.

The notion of the logical difference has been introduced by Konev, Walther, and Wolter (2008) as a way of capturing the difference in the meaning of terms that is independent of the representation of ontologies.

Preliminaries

We start with introducing the description logic $\mathcal{ALC}$. Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite and mutually disjoint sets of *concept names* and *role names*. *$\mathcal{ALC}$-concepts* are built according to the following syntax rule

$$C ::= \quad A \quad | \quad \top \quad | \quad \neg C \quad | \quad \exists r.C \quad | \quad C \sqcap D,$$

where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. As usual, other $\mathcal{ALC}$ concept constructors are introduced as abbreviations: $\bot$ stands for $\neg\top$, $C \sqcup D$ stands for $\neg(\neg C \sqcap \neg D)$ and $\forall r.C$ stands for $\neg\exists r.\neg C$. An $\mathcal{ALC}$-TBox $\mathcal{T}$ is a finite set of $\mathcal{ALC}$-inclusions of the form $C \sqsubseteq D$, where C and D are $\mathcal{ALC}$-concepts. A concept equation $C \equiv D$ is an abbreviation for the two inclusions $C \sqsubseteq D$ and $D \sqsubseteq C$. An $\mathcal{ALC}$-TBox $\mathcal{T}$ is *acyclic* if all its inclusions are of the form $A \sqsubseteq C$ and $A \equiv C$, where $A \in \mathsf{N_C}$ and C is an $\mathcal{ALC}$-concept, such that no concept name occurs more than once on the left-hand side and $\mathcal{T}$ contains no cycle in its definitions, i.e., it does not contain inclusions $A_1 \bowtie C_1, \ldots, A_k \bowtie C_k$, where $\bowtie \in \{\sqsubseteq, \equiv\}$, such that A_{i+1} occurs in C_i, for $i = 1, \ldots, k-1$ and $A_k = A_1$.

A signature Σ is a finite subset of $\mathsf{N_C} \cup \mathsf{N_R}$. The signature of a concept C, denoted by $\mathsf{sig}(C)$, is the set of concept and role names that occur in C. If $\mathsf{sig}(C) \subseteq \Sigma$, we call C a Σ-concept. We assume that the two previous definitions also apply to concept inclusions/equations $C \bowtie D$ with $\bowtie \in \{\sqsubseteq, \equiv\}$ and to TBoxes $\mathcal{T}$. The size of a concept C is the length of the string that represents it, where concept names and role names are considered to be of length one. The size of an inclusion/equation $C \bowtie D$ with $\bowtie \in \{\sqsubseteq, \equiv\}$ is the sum of the sizes of C and D plus one. The size of a TBox $\mathcal{T}$ is the sum of the sizes of its inclusions.

The semantics of $\mathcal{ALC}$ is given by *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the *domain* $\Delta^{\mathcal{I}}$ is a non-empty set, and $\cdot^{\mathcal{I}}$ is a function mapping each concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The *extension* $C^{\mathcal{I}}$ of a concept C is defined by induction as follows:

$$\begin{array}{lcl} \top^{\mathcal{I}} & := & \Delta^{\mathcal{I}} \\ (\neg C)^{\mathcal{I}} & := & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\ (\exists r.C)^{\mathcal{I}} & := & \{\, d \in \Delta^{\mathcal{I}} \mid \exists\, e \in C^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}} \,\} \\ (C \sqcap D)^{\mathcal{I}} & := & C^{\mathcal{I}} \cap D^{\mathcal{I}}. \end{array}$$

Then $\mathcal{I}$ *satisfies* a concept inclusion $C \sqsubseteq D$, in symbols $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

We say that an interpretation $\mathcal{I}$ is a *model of a TBox* $\mathcal{T}$ if $\mathcal{I} \models C \sqsubseteq D$ for all $C \sqsubseteq D \in \mathcal{T}$. An $\mathcal{ALC}$-inclusion $C \sqsubseteq D$ *follows from* (or is *entailed by*) a TBox $\mathcal{T}$ if every model of $\mathcal{T}$ is a model of $C \sqsubseteq D$, in symbols $\mathcal{T} \models C \sqsubseteq D$. We use $\models C \sqsubseteq D$ to denote that $C \sqsubseteq D$ follows from the empty TBox. Finally, a TBox $\mathcal{T}'$ *follows from* (or is *entailed by*) a TBox $\mathcal{T}$ if every model of $\mathcal{T}$ is a model of $\mathcal{T}'$, in symbols $\mathcal{T} \models \mathcal{T}'$.

We now introduce the main notion that we study in this paper.

Definition 1. *Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox and let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ be a signature. We say that an $\mathcal{ALC}$-TBox $\mathcal{T}_\Sigma$ is a* Σ-uniform interpolant *of the TBox $\mathcal{T}$ iff $\mathsf{sig}(\mathcal{T}_\Sigma) \subseteq \Sigma$, $\mathcal{T} \models \mathcal{T}_\Sigma$, and for every $\mathcal{ALC}$ Σ-concept inclusion $C \sqsubseteq D$ with $\mathcal{T} \models C \sqsubseteq D$ it holds that $\mathcal{T}_\Sigma \models C \sqsubseteq D$.*

Uniform interpolation can be seen as the dual notion of *forgetting*: a TBox $\mathcal{T}_\Upsilon$ is the result of forgetting about a signature Υ in a TBox $\mathcal{T}$ iff $\mathcal{T}_\Upsilon$ is a uniform interpolant of $\mathcal{T}$ w.r.t. $\Sigma = \mathsf{sig}(\mathcal{T}) \setminus \Upsilon$. As the following example shows, uniform interpolants of $\mathcal{ALC}$-TBoxes do not always exist.

Example 2. *Let $\mathcal{T} = \{A \sqsubseteq B,\ B \sqsubseteq C \sqcap \exists r.B\}$ and $\Sigma = \{A, C, r\}$. Then there does not exist a Σ-uniform inter-*

[1] Available from http://lat.inf.tu-dresden.de/~michel/publications/

polant of $\mathcal{T}$ as (in particular) the infinite number of consequences of the form $A \sqsubseteq \exists r.C$, $A \sqsubseteq \exists r.\exists r.C, \ldots$ cannot be captured by an $\mathcal{ALC}$-TBox $\mathcal{T}'$ with $\mathsf{sig}(\mathcal{T}') \subseteq \Sigma$. On the other hand, for $\mathcal{T}' = \{A \sqsubseteq B,\ B \sqsubseteq C \sqcap \exists r.B,\ D \equiv B\}$ and $\Sigma' = \{A, C, D, r\}$, a Σ'-uniform interpolant of $\mathcal{T}'$ is $\{A \sqsubseteq D,\ D \sqsubseteq C \sqcap \exists r.D\}$.

Uniform interpolation is also related to the notion of logical difference between ontologies.

Definition 3. *The Σ-logical difference between $\mathcal{ALC}$-TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ is the set $\mathsf{Diff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ of all $\mathcal{ALC}$-concept inclusions $C \sqsubseteq D$ such that $\mathsf{sig}(C \sqsubseteq D) \subseteq \Sigma$, $\mathcal{T}_1 \models C \sqsubseteq D$, and $\mathcal{T}_2 \not\models C \sqsubseteq D$.*

It is easy to see that $\mathsf{Diff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2) = \emptyset$ if, and only if, $\mathcal{T}_2 \models \mathcal{T}_1^{(\Sigma)}$ where $\mathcal{T}_1^{(\Sigma)}$ is a Σ-uniform interpolant of $\mathcal{T}_1$. Moreover, if $\mathcal{T}_2 \not\models \mathcal{T}_1^{(\Sigma)}$, every inclusion $C \sqsubseteq D \in \mathcal{T}_1^{(\Sigma)}$ with $\mathcal{T}_2 \not\models C \sqsubseteq D$ can be regarded as a *witness* of $\mathsf{Diff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$.

With the exception of acyclic $\mathcal{EL}$-TBoxes, checking whether the logical difference between two ontologies is nonempty is at least one exponential harder than reasoning (Konev et al. 2012). Additionally, if the set $\mathsf{Diff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ is nonempty, it is typically infinite. Therefore, in practice, the notion of logical difference is primarily used as a theoretical underpinning of its *approximations* that limit the choice of inclusions $C \sqsubseteq D$ in Definition 3 to Σ-inclusions constructed according to some syntactic rules, see e.g. (Jiménez-Ruiz et al. 2009), (Gonçalves, Parsia, and Sattler 2012).

Computing Uniform Interpolants by $\mathcal{ALC}$-Resolution

The aim of our work is to investigate a practical approach for computing uniform interpolants when they exist. Note that the worst-case optimal procedure given by Lutz and Wolter (2011) is inherently inefficient as it requires one to explicitly construct the internalisation $C_\mathcal{T}$ of a given TBox $\mathcal{T}$, i.e. a concept $C_\mathcal{T}$ of size double exponential in the size of $\mathcal{T}$ having the property that for any inclusion $C \sqsubseteq D$ it holds that $\mathcal{T} \models C \sqsubseteq D$ iff $\models C \sqcap C_\mathcal{T} \sqsubseteq D$.

Our approach is to introduce a resolution-like calculus for $\mathcal{ALC}$ that derives consequences of a TBox $\mathcal{T}$ such that a concept inclusion $C \sqsubseteq D$ is entailed by $\mathcal{T}$ iff a contradiction can be derived from $\mathcal{T}$ and $C \sqcap \neg D$. Similarly to Herzig and Mengin (2008), we then show that any derivation can be restructured in such a way that inferences on selected concept names always precede inferences on other concept names. Then, if the signature Σ is such that $\mathsf{sig}(\mathcal{T}) \setminus \Sigma$ only contains concept names, we generate a set of Σ-consequences $\mathcal{T}'$ of $\mathcal{T}$ by applying the inference rules in a forward chaining manner such that for an arbitrary Σ-inclusion $C \sqsubseteq D$ a contradiction can be derived from $\mathcal{T}$ and $C \sqcap \neg D$ iff a contradiction can be derived from $\mathcal{T}'$ and $C \sqcap \neg D$. Thus, if the forward-chaining process terminates, $\mathcal{T}'$ is a Σ-uniform-interpolant for $\mathcal{T}$.

$\mathcal{ALC}$-Resolution. $\mathcal{ALC}$-resolution operates on $\mathcal{ALC}$ formulae in a conjunctive normal form that is defined according to the following grammar (which is similar to the normal form introduced by Herzig and Mengin (2008)):

$$\begin{array}{rcl} \text{Literal} & ::= & A \mid \neg A \mid \forall r.\text{Clause} \mid \exists r.\text{CNF} \\ \text{Clause} & ::= & \text{Literal} \mid \text{Clause} \sqcup \text{Clause} \mid \bot \\ \text{CNF} & ::= & \top \mid \text{Clause} \mid \text{Clause} \sqcap \text{CNF} \end{array}$$

To simplify the presentation, we assume that clauses are sets of literals and that CNF expressions are sets of clauses. Then $\bot$ corresponds to the empty clause and $\top$ to the empty set of clauses. In the following, the calligraphic letters $\mathcal{C}, \mathcal{D}, \mathcal{E}$ symbolise clauses and $\mathcal{F}, \mathcal{G}$ represent sets of clauses. Similarly to first-order formulae, every $\mathcal{ALC}$ concept can be transformed into an equivalent set of $\mathcal{ALC}$ clauses. The depth of a clause $\mathcal{C}$, $\mathrm{Depth}(\mathcal{C})$, is defined to be the maximal nesting depth of the quantifiers contained in $\mathcal{C}$.

We additionally assume that every clause is assigned a *type*. Clauses obtained from the clausification of TBox inclusions are of the type *universal*, and clauses resulting from the clausification of inclusions to be tested for entailment by the TBox are of the type *initial*. The type of a derived clause is determined by the types of the clauses from which it is derived and by the derivation rule that is used.

Example 4. *The clausification of $\mathcal{T}$ from Example 2 produces three universal clauses: $\neg A \sqcup B$, $\neg B \sqcup C$, $\neg B \sqcup \exists r.B$.*

We now introduce the two resolution calculi $\mathfrak{T}$ and $\mathfrak{T}^u$. The former calculus assumes the TBox to be empty, whereas the latter takes TBox inclusions into account. Thus, $\mathfrak{T}$ derives the empty clause from the set of initial clauses stemming from the clausification of an inclusion $\top \sqsubseteq C \sqcap \neg D$ iff $\models C \sqsubseteq D$; and $\mathfrak{T}^u$ derives the empty clause from the universal clauses stemming from the clausification of a TBox $\mathcal{T}$ and the initial clauses stemming from the clausification of an inclusion $\top \sqsubseteq C \sqcap \neg D$ iff $\mathcal{T} \models C \sqsubseteq D$.

The calculus $\mathfrak{T}$ is defined with the help of the relation $\Rightarrow_\alpha$ given in Fig. 1. For every $\alpha \in \mathsf{N_C} \cup \{\bot\}$, the relation $\Rightarrow_\alpha$ associates with a set of clauses $\mathcal{N}$ a new clause $\mathcal{C}$ which can be 'derived' from the set $\mathcal{N}$ by 'resolving' on α. $\mathfrak{T}$ now consists of the following two inference rules.

$$\frac{\mathcal{C}}{\mathcal{E}}\ (\text{if } \mathcal{C} \Rightarrow_\alpha \mathcal{E}) \qquad\qquad \frac{\mathcal{C} \quad \mathcal{D}}{\mathcal{E}}\ (\text{if } \mathcal{C}, \mathcal{D} \Rightarrow_\alpha \mathcal{E}),$$

where $\mathcal{C}, \mathcal{D}$, and $\mathcal{E}$ are initial clauses.

The calculus $\mathfrak{T}^u$ operates initial and universal clauses and also consists of two rules:

$$\frac{\mathcal{C}}{\mathcal{E}}\ (\text{if } \mathcal{C} \Rightarrow_\alpha \mathcal{E}) \qquad\qquad \frac{\mathcal{C}' \quad \mathcal{D}}{\mathcal{E}'}\ (\text{if } \mathcal{C}', \mathcal{D} \Rightarrow^u_\alpha \mathcal{E}'),$$

where $\mathcal{C}, \mathcal{C}', \mathcal{D}$ are initial or universal clauses, and $\mathcal{C}', \mathcal{D} \Rightarrow^u_\alpha \mathcal{E}'$ holds iff either $\mathcal{C}', \mathcal{D} \Rightarrow_\alpha \mathcal{E}'$, or $\mathcal{D}$ is a universal clause and there exist role names $r_1, \ldots, r_n \in \mathsf{N_R}$ ($n \geq 1$) such that $\mathcal{C}', \forall r_1. \ldots \forall r_n.\mathcal{D} \Rightarrow_\alpha \mathcal{E}'$. (Intuitively, the calculus $\mathfrak{T}^u$ allows for inferences with universal clauses at arbitrary nesting levels of quantifiers, which the calculus $\mathfrak{T}$ does not.) Then $\mathcal{E}$ is a universal clause if $\mathcal{C}$ is a universal clause, and an initial clause otherwise. Similarly, $\mathcal{E}'$ is a universal clause if both $\mathcal{C}'$ and $\mathcal{D}$ are universal clauses, and an initial clause otherwise.

We assume that every clause $\mathcal{E}$ that results from a $\mathfrak{T}$- or $\mathfrak{T}^u$-inference is implicitly simplified by exhaustively removing all occurrences of literals of the form $\exists r.(\mathcal{F}, \bot)$.

(rule $\bot$)	$\mathcal{C}'_1 \sqcup \forall r.\bot,\ \ \mathcal{C}'_2 \sqcup \exists r.\mathcal{F} \Longrightarrow_\bot \mathcal{C}'_1 \sqcup \mathcal{C}'_2$	
(rule A)	$\mathcal{C}'_1 \sqcup A,\ \ \mathcal{C}'_2 \sqcup \neg A \Longrightarrow_A \mathcal{C}'_1 \sqcup \mathcal{C}'_2$	
(rule $\forall\exists$)	$\mathcal{C}'_1 \sqcup \forall r.\mathcal{C}_1,\ \ \mathcal{C}'_2 \sqcup \exists r.(\mathcal{C}_2,\mathcal{F}) \Longrightarrow_\alpha \mathcal{C}'_1 \sqcup \mathcal{C}'_2 \sqcup \exists r.(\mathcal{C}_2,\mathcal{F},\mathcal{C}_3),$	if $\mathcal{C}_1,\mathcal{C}_2 \Longrightarrow_\alpha \mathcal{C}_3$
(rule $\forall\forall$)	$\mathcal{C}'_1 \sqcup \forall r.\mathcal{C}_1,\ \ \mathcal{C}'_2 \sqcup \forall r.\mathcal{C}_2 \Longrightarrow_\alpha \mathcal{C}'_1 \sqcup \mathcal{C}'_2 \sqcup \forall r.\mathcal{C}_3,$	if $\mathcal{C}_1,\mathcal{C}_2 \Longrightarrow_\alpha \mathcal{C}_3$
(rule $\exists_1$)	$\mathcal{C}' \sqcup \exists r.(\mathcal{C}_1,\mathcal{F}) \Longrightarrow_\alpha \mathcal{C}' \sqcup \exists r.(\mathcal{C}_1,\mathcal{F},\mathcal{C}_2),$	if $\mathcal{C}_1 \Longrightarrow_\alpha \mathcal{C}_2$
(rule $\exists_2$)	$\mathcal{C}' \sqcup \exists r.(\mathcal{C}_1,\mathcal{C}_2,\mathcal{F}) \Longrightarrow_\alpha \mathcal{C}' \sqcup \exists r.(\mathcal{C}_1,\mathcal{C}_2,\mathcal{F},\mathcal{C}_3),$	if $\mathcal{C}_1,\mathcal{C}_2 \Longrightarrow_\alpha \mathcal{C}_3$
(rule $\forall$)	$\mathcal{C}' \sqcup \forall r.\mathcal{C}_1 \Longrightarrow_\alpha \mathcal{C}' \sqcup \forall r.\mathcal{C}_2,$	if $\mathcal{C}_1 \Longrightarrow_\alpha \mathcal{C}_2$

Figure 1: Rules of $\Longrightarrow_\alpha$.

Example 5. *For the universal clauses from Example 4, we have for instance,*

$$\neg A \sqcup B, \neg B \sqcup \exists r.B \ \Rightarrow_B \ \neg A \sqcup \exists r.B \quad \textit{by (rule A).}$$

So, the universal clause $\neg A \sqcup \exists r.B$ is derivable by $\mathfrak{T}^u$ from $\neg A \sqcup B$ and $\neg B \sqcup \exists r.B$. As $\neg B \sqcup C$ is a universal clause and

$$\neg B \sqcup \exists r.B, \forall r.\neg B \sqcup C \Rightarrow_B \neg B \sqcup \exists r.(B,C) \quad \textit{by (rule } \forall\exists\textit{),}$$

the universal clause $\neg B \sqcup \exists r.(B,C)$ is derivable by $\mathfrak{T}^u$ from $\neg B \sqcup \exists r.B$ and $\neg B \sqcup C$. By applying the inference rules to old and newly generated clauses, one can conclude that the universal clauses $\neg A \sqcup \exists r.(B,C)$ and $\neg A \sqcup \exists r.(B, \exists r.B)$ are also derivable by $\mathfrak{T}^u$ from $\mathcal{N} = \{\neg A \sqcup B,\ \neg B \sqcup C,\ \neg B \sqcup \exists r.B\}$.

For $x \in \{\mathfrak{T}, \mathfrak{T}^u\}$, a x-derivation (tree) Δ built from a set of clauses $\mathcal{N}$ is a finite binary tree in which each leaf is labelled with a clause from $\mathcal{N}$ and each non-leaf node n is labelled with a clause $\mathcal{C}$ such that $\mathcal{C}$ results from an x-inference on the parent(s) of n in Δ. We say that Δ is a derivation of a clause $\mathcal{C}$ if the root of Δ is labelled with $\mathcal{C}$. A derivation of the empty clause is called a *refutation*. Every path $n_1, \ldots, n_m$ of nodes in Δ where n_1 is a leaf node and n_m is the root node induces an *inference path* $\alpha_2, \ldots, \alpha_m$, where $\alpha_i \in \mathsf{N_C} \cup \{\bot\}$ $(2 \leq i \leq m)$ denotes the concept name, or $\bot$, which has been resolved upon to obtain the clause that is the label of the node n_i. For a signature $\Upsilon \subseteq \mathsf{N_C}$ and a strict total order $\succ \subseteq \Upsilon \times \Upsilon$, a derivation Δ is a $(x, \Upsilon, \succ)$-derivation if for every inference path $\alpha_1, \ldots, \alpha_n$ of Δ (with $\alpha_i \in \mathsf{N_C} \cup \{\bot\}$ for every $1 \leq i \leq n$) there exists $0 \leq k \leq n$ such that $\{\alpha_1, \ldots, \alpha_k\} \subseteq \Upsilon$, $\alpha_j \succ \alpha_{j+1}$ or $\alpha_j = \alpha_{j+1}$ for every $1 \leq j < k$, and $\alpha_j \notin \Upsilon$ for every $k < j \leq n$.

We prove that for every unsatisfiable set of initial clauses there always exists a $(\mathfrak{T}, \Upsilon, \succ)$-refutation by extending the results and proof methods of Herzig and Mengin (2008).

Theorem 6 ($\mathfrak{T}$-Completeness)**.** *Let $\Upsilon \subseteq \mathsf{N_C}$, let $\succ \subseteq \Upsilon \times \Upsilon$ be a strict total order on Υ and let C and D be $\mathcal{ALC}$ concepts. Then it holds that $\models C \sqsubseteq D$ iff there exists a $(\mathfrak{T}, \Upsilon, \succ)$-derivation of the empty clause from the initial clauses* $\mathrm{Cls}(C \sqcap \neg D)$.

A weaker version of this result, stating that any derivation in $\mathfrak{T}$ can be reordered so that inferences on concept names from Υ always precede inferences on other concept names, or $\bot$, has been previously announced by Herzig and Mengin (2008); however, as we show in the full version of the paper, the proof appears to have some gaps.

To prove completeness for $\mathfrak{T}^u$, we observe the following link between derivations in $\mathfrak{T}$ and $\mathfrak{T}^u$. Let $\mathcal{N}$ be a set of clauses and let

$$\begin{aligned}
\mathrm{Univ}_0(\mathcal{N}) &= \mathcal{N};\\
\mathrm{Univ}_{i+1}(\mathcal{N}) &= \mathrm{Univ}_i(\mathcal{N}) \cup \\
&\qquad \textstyle\bigcup_{r \in \mathsf{N_R} \cap \mathsf{sig}(\mathcal{N})} \{\, \forall r.\mathcal{C} \mid \mathcal{C} \in \mathrm{Univ}_i(\mathcal{N}) \,\}\\
\text{and } \mathrm{Univ}(\mathcal{N}) &= \textstyle\bigcup_{i \geq 0} \mathrm{Univ}_i(\mathcal{N}).
\end{aligned}$$

Theorem 7. *Let $\mathcal{M}$ be a set of initial clauses and let $\mathcal{N}$ be a set of universal clauses. Additionally, let Δ be a $(\mathfrak{T}, \Upsilon, \succ)$-refutation from $\mathcal{M} \cup \mathrm{Univ}(\mathcal{N})$ such that there exists $n \in \mathbb{N}$ with $\mathrm{Depth}(\mathcal{C}) \leq n$ for every $\mathcal{C} \in$ Clauses(Δ). Then there exists a $(\mathfrak{T}^u, \Upsilon, \succ)$-derivation Δ^u of the empty clause from $\mathcal{M} \cup \mathcal{N}$ such that $\mathrm{Depth}(\mathcal{C}) \leq n$ for every $\mathcal{C} \in$ Clauses(Δ^u).*

We then use Theorems 6 and 7 and the fact that every $\mathcal{ALC}$-TBox can be internalised. Notice that the actual TBox internalisation $C_\mathcal{T}$ does not have to be computed as it is only used for the proof of completeness.

Corollary 8 ($\mathfrak{T}^u$-Completeness)**.** *Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox, let $\Upsilon \subseteq \mathsf{N_C}$, let $\succ \subseteq \Upsilon \times \Upsilon$ be a strict total order on Υ and let C and D be $\mathcal{ALC}$ concepts. Then it holds that $\mathcal{T} \models C \sqsubseteq D$ iff there exists a $(\mathfrak{T}^u, \Upsilon, \succ)$-derivation of the empty clause from the universal clauses* $\mathrm{Cls}(\mathcal{T})$ *and the initial clauses* $\mathrm{Cls}(C \sqcap \neg D)$.

Computing Uniform Interpolants. The procedure UNIFORMINTERPOLANT depicted in Algorithm 1 takes as input an $\mathcal{ALC}$-TBox $\mathcal{T}$, a signature $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ such that $\Sigma \cap \mathsf{N_R} = \mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R}$ and a strict total order $\succ \subseteq \Upsilon \times \Upsilon$ over $\Upsilon = \mathsf{sig}(\mathcal{T}) \setminus \Sigma$. Following the outline given by Herzig and Mengin (2008), after the clausification of $\mathcal{T}$, the procedure iterates over the concept names contained in Υ in descending order according to the relation $\succ$. In each iteration the clause set $\mathcal{N}$ is expanded with all possible $\mathfrak{T}^u$-inferences on the current concept name $A \in \Upsilon$. Finally, after iterating over all the concept names from $\Upsilon = \mathsf{sig}(\mathcal{T}) \setminus \Sigma$, the operator 'Supp' is applied on the resulting clauses, which replaces all occurrences of Υ concept names in clauses with $\top$ and then simplifies the resulting CNF.

Example 9. *For the clauses obtained in Example 5,* $\mathrm{Supp}(\{B\}, \neg A \sqcup C) = \neg A \sqcup C$, $\mathrm{Supp}(\{B\}, \neg A \sqcup \exists r.B) = \neg A \sqcup \exists r.\top$, $\mathrm{Supp}(\{B\}, \neg A \sqcup \exists r.(B,C)) = \neg A \sqcup \exists r.C$.

Algorithm 1

procedure UNIFORMINTERPOLANT($\mathcal{T}$, Σ, $\succ$)
  $\Upsilon := \mathsf{sig}(\mathcal{T}) \setminus \Sigma$
  $\mathcal{N} := \mathrm{Cls}(\mathcal{T})$
  while $\Upsilon \neq \emptyset$ **do**
    $A := \max_{\succ}(\Upsilon)$
    $\mathcal{N} := \mathrm{Res}^{\infty}_{\mathfrak{T}^u,\{A\}}(\mathcal{N})$
    $\Upsilon := \Upsilon \setminus \{A\}$
  end while
  return $\mathcal{F}_{\Sigma}(\mathcal{T}) = \mathrm{Supp}(\mathsf{sig}(\mathcal{T}) \setminus \Sigma, \mathcal{N})$
end procedure

One can show that if Algorithm 1 terminates, then for all $\mathcal{ALC}$ Σ-concepts C, D such that there exists a $(\mathfrak{T}^u, \Upsilon, \succ)$-refutation Δ^u from the universal clauses $\mathrm{Cls}(\mathcal{T})$ and the initial clauses $\mathrm{Cls}(C \sqcap \neg D)$ it holds that $\mathcal{F}_{\Sigma}(\mathcal{T}) \models C \sqsubseteq D$. Thus, it follows from Corollary 8 that if Algorithm 1 terminates, it computes a Σ-uniform interpolant of $\mathcal{T}$. However, Algorithm 1 does not terminate if a uniform interpolant does not exist. For example, when applied to $\mathcal{T}$ from Example 2, Algorithm 1 can generate, among others, the infinite sequence of universal clauses $\neg A \sqcup \exists r.C$, $\neg A \sqcup \exists r.(C, \exists r.C), \ldots$ and so on. Moreover, as the TBox $\mathcal{T}$ from Example 2 is a subset of $\mathcal{T}'$, and so $\mathrm{Cls}(\mathcal{T}) \subseteq \mathrm{Cls}(\mathcal{T}')$, Algorithm 1 will derive, among others, the same clauses when it is applied on $\mathcal{T}'$. Thus, in some cases Algorithm 1 does not terminate even though a uniform interpolant exists.

To guarantee termination on all inputs, we focus on the notion of depth-bounded uniform interpolation (related to the notion of 'bounded forgetting' (Zhou and Zhang 2011)). Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox and let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ be a signature. We say that an $\mathcal{ALC}$-TBox $\mathcal{T}_{\Sigma}$ is a *depth n-bounded uniform interpolant* of the TBox $\mathcal{T}$ w.r.t. Σ iff $\mathsf{sig}(\mathcal{T}_{\Sigma}) \subseteq \Sigma$, $\mathcal{T} \models \mathcal{T}_{\Sigma}$, and for every $\mathcal{ALC}$ Σ-concept inclusion $C \sqsubseteq D$ with $\mathcal{T} \models C \sqsubseteq D$ and $\max\{\mathrm{Depth}(C), \mathrm{Depth}(D)\} \leq n$ it holds that $\mathcal{T}_{\Sigma} \models C \sqsubseteq D$. Let $\mathcal{F}_{\Sigma,m}(\mathcal{T})$ be the outcome of Algorithm 1 where in Step 6 only clauses up to depth m are generated. The following example shows that it might be necessary to consider intermediate clauses of a depth $m > n$ in order to preserve all the Σ-consequences of depth n entailed by $\mathcal{T}$.

Example 10. *Let $\mathcal{T} = \{A \sqsubseteq \exists r.C,\ C \sqsubseteq \exists s.\top,\ \neg B \sqsubseteq \forall s.\bot\}$, $\Sigma = \{A, B, r, s\}$, $\Upsilon = \{C\}$ and $\succ = \emptyset$. Then every $(\mathfrak{T}^u, \Upsilon, \succ)$-refutation from the universal clauses $\mathrm{Cls}(\mathcal{T})$ and the initial clauses $\{A, \forall r.\neg B\}$ derives the clause $\neg A \sqcup \exists r.(C, \exists s.\top)$.*

We establish, however, that by choosing the maximal depth of derived clauses appropriately, the procedure depicted in Algorithm 1 computes uniform interpolants that preserve consequences up to a specified depth n.

Theorem 11. *Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox, $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ a signature such that $\Sigma \cap \mathsf{N_R} = \mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R}$, and let $n \geq 0$. Set $m = n + 2^{|\mathsf{sub}(\mathrm{Cls}(\mathcal{T}))|+1} + \max\{\mathrm{Depth}(\mathcal{C}) \mid \mathcal{C} \in \mathrm{Cls}(\mathcal{T})\}$, where $\mathsf{sub}(\mathrm{Cls}(\mathcal{T}))$ is the set of subconcepts of $\mathrm{Cls}(\mathcal{T})$. Then it holds that $\mathcal{F}_{\Sigma,m}(\mathcal{T})$ is a depth n-bounded uniform interpolant of the TBox $\mathcal{T}$ w.r.t. Σ.*

We can combine this result with the results of (Lutz and Wolter 2011): for any $\mathcal{ALC}$-TBox $\mathcal{T}$ and signature Σ, if a Σ-uniform interpolant of $\mathcal{T}$ exists, then there exists a uniform interpolant of depth bounded by $2^{2^{|\mathcal{T}|+1}} + 1$. Thus, if $\Sigma \cap \mathsf{N_R} = \mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R}$, there exists m, which can be computed based on the bound in Theorem 11 and the results of (Lutz and Wolter 2011), such that $\mathcal{F}_{\Sigma,m}(\mathcal{T})$ is a Σ-uniform interpolant of $\mathcal{T}$.

The bound in Theorem 11 can be significantly improved if the TBox is acyclic. For an acyclic $\mathcal{ALC}$-TBox $\mathcal{T}$ we define $\mathrm{ExpansionDepth}(\mathcal{T}) = \max\{\mathrm{Depth}(A[\mathcal{T}]) \mid A \in \mathsf{sig}(\mathcal{T})\}$, where $A[\mathcal{T}]$ denotes the concept obtained by exhaustively replacing every concept B with C_B if $B \sqsubseteq C_B \in \mathcal{T}$ or $B \equiv C_B \in \mathcal{T}$.

Theorem 12. *Let $\mathcal{T}$ be an acyclic $\mathcal{ALC}$ TBox, $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ a signature such that $\Sigma \cap \mathsf{N_R} = \mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R}$, and let $n \geq 0$. Set $m = \mathrm{ExpansionDepth}(\mathcal{T}) + n$. Then it holds that $\mathcal{F}_{\Sigma,m}(\mathcal{T})$ is a uniform interpolant limited to consequence depth n of the TBox $\mathcal{T}$ w.r.t. Σ.*

Note that in the description logic $\mathcal{EL}$ (i.e. the fragment of $\mathcal{ALC}$ that does not allow $\bot$, negation, disjunction, or universal quantification) the acyclicity of a TBox guarantees the existence of uniform interpolants (Konev, Walther, and Wolter 2009) for any signature Σ. Interestingly, this is not true in the case of $\mathcal{ALC}$. Moreover, as the following example shows, there exists an acyclic $\mathcal{EL}$-TBox $\mathcal{T}$ and a signature Σ for which no $\mathcal{ALC}$ Σ-uniform interpolant exists.

Example 13. *Consider $\Sigma = \{A, A_0, A_1, A_2, E, r\}$ and $\mathcal{T} = \{A \sqsubseteq \exists r.B,\ A_0 \sqsubseteq \exists r.(A_1 \sqcap B),\ E \equiv A_1 \sqcap B \sqcap \exists r.(A_2 \sqcap B)\}$. Then for every $n \geq 0$, $\mathcal{T}$ entails the inclusion*

$$A_0 \sqcap \prod_{i=1}^{n} \underbrace{\forall r. \ldots \forall r.}_{i}(A \sqcap \neg E \sqcap (A_1 \sqcup A_2)) \sqsubseteq \underbrace{\exists r. \ldots \exists r.}_{n} A_1.$$

This infinite sequence of $\mathcal{ALC}$ consequences of $\mathcal{T}$ cannot be captured by any $\mathcal{ALC}$ Σ-TBox $\mathcal{T}'$, which can be proved formally using Theorem 9 of Lutz and Wolter (2011).

Case Study

We have implemented a prototype of an inference computation architecture using the calculus $\mathfrak{T}^u$ and the inference relation $\Rightarrow_{\alpha}$ in Java. However, it turned out that our initial implementation of Algorithm 1 did not perform well in practice. This was in particular due to the fact that clauses can contain sets $\mathcal{F}$ of other clauses in existential literals $\exists r.\mathcal{F}$, which renders all the possible inferences on clauses from $\mathcal{F}$ 'explicit'. For example, if we resolve the universal clause which just consists of the existential literal $\exists r.(A)$ with the universal clauses $\neg A \sqcup B_1, \ldots, \neg A \sqcup B_n$ on the concept name A, then not only the clauses $\exists r.(A, B_1)$, $\exists r.(A, B_2)$, $\exists r.(A, B_3)$,... could be derived but all clauses of the form $\exists r.(A, \mathcal{G})$, where $\mathcal{G}$ is a subset of $\{B_1, \ldots, B_n\}$.

A common technique to reduce the number of inferences that have to be made is to use forward- and backward deletion of subsumed clauses (Bachmair and Ganzinger 2001). However, it is known (Auffray, Enjalbert, and Hébrard 1990) that the subsumption lemma (stating that if a clause $\mathcal{E}$ results from an inference involving two clauses $\mathcal{C}$ and $\mathcal{D}$, and

	Uniform Interpolation				Forgetting					
	$\lvert\Sigma \cap \mathsf{N_C}\rvert = 5$		$\lvert\Sigma \cap \mathsf{N_C}\rvert = 10$		$\lvert\Upsilon\rvert = 10$		$\lvert\Upsilon\rvert = 15$		$\lvert\Upsilon\rvert = 25$	
	Success Rate (%)	Avrg # Axioms	Success Rate (%)	Avrg # Axioms	Success Rate (%)	Avrg # Axioms	Success Rate (%)	Avrg # Axioms	Success Rate (%)	Avrg # Axioms
AMINO-ACID v1.2	100	61.40	92	143.35	100	645.67	87	665.24	64	396.98
BHO v0.4	71	30.01	16	52.43	99	2374.73	99	2363.42	91	2383.96
CAO v1.4	100	279.02	100	283.33	100	369.54	100	369.22	10	366.07
CDAO	100	288.21	100	288.42	100	293.48	100	293.41	10	293.02
CHEMBIO v1.1	92	7 1.89	60	94.40	100	295.85	100	293.09	10	293.64
CPRO v0.85	100	585.08	100	533.82	100	307.76	100	309.46	10	316.31
DDI v0.9	100	249.80	100	259.41	100	276.27	100	278.55	10	276.61
DIKB v1.4	2	1591.50	0	-	97	622.67	83	689.44	56	816.39
GRO v0.5	0	-	0	-	94	959.85	91	940.03	79	997.59
IDO	0	-	0	-	94	1202.71	90	1203.78	80	1215.36
LIPRO v1.1	73	7.93	58	13.22	91	2287.24	58	2381.43	45	2297.37
NCI v08.10e	23	887.34	1	1397.00	97	100693.26	98	100611.60	99	100889.50
NEOMARK v4.1	31	19.45	14	27.28	100	338.52	100	333.26	10	324.86
OMRSE	100	485.00	100	485.00	100	485.00	100	485.00	10	485.00
OBIWS v1.1	100	112.56	100	118.70	100	189.66	100	187.71	10	184.13
ONTODM v1.1.1	0	-	0	-	98	1711.40	98	1704.67	93	1693.61
OPL	100	829.41	100	832.93	100	848.60	100	848.99	10	848.73
PROPREO v1.1	41	2.07	19	31.84	100	561.43	100	560.85	99	578.08
RNAO r113	100	355.86	100	362.83	100	439.64	100	439.10	10	439.71
SAO v1.2.4	0	-	0	-	99	2702.23	100	2700.85	98	2715.30
SITBAC v1.3	0	-	0	-	93	508.40	93	537.48	79	595.51
TOK v0.2.1	0	-	0	-	97	496.12	93	529.06	72	567.11
VSO	0	-	0	-	83	348.87	79	397.65	50	371.38

Table 1: Uniform Interpolation and Forgetting for BioPortal Ontologies on Small Signatures.

if there exist clauses $\mathcal{C}'$, $\mathcal{D}'$ such that $\mathcal{C}'$ subsumes $\mathcal{C}$ and $\mathcal{D}'$ subsumes $\mathcal{D}$, then either $\mathcal{E}$ is subsumed by one of $\mathcal{C}'$, $\mathcal{D}'$, or a clause $\mathcal{E}'$ can be derived from $\mathcal{C}'$ and $\mathcal{D}'$ such that $\mathcal{E}'$ subsumes $\mathcal{E}$) does not hold even in the modal logic K for the standard *minimal subsumption relation* $\leq_s$ (Auffray, Enjalbert, and Hébrard 1990) and $\Rightarrow_\alpha$. To be able to prove that one can safely discard subsumed clauses, we have modified the inference relation $\Rightarrow_\alpha$ by introducing the following additional rule (rule $\exists_f$)

$$\mathcal{C}_1 \sqcup \forall r.\mathcal{D},\ \ \mathcal{C}_2 \sqcup \exists r.\mathcal{F} \Longrightarrow_{\exists_f} \mathcal{C}_1 \sqcup \mathcal{C}_2 \sqcup \exists r.(\mathcal{F}, \mathcal{D}).$$

We will denote the resulting inference relation by $\Rightarrow_\alpha^f$ with $\alpha \in \mathsf{N_C} \cup \{\bot, \exists_f\}$. One can then prove that a variant of the subsumption lemma holds for the relations $\leq_s$ and $\Rightarrow_\alpha^f$, which allows us to employ forward- and backward deletion of subsumed clauses in our implementation.

In order to further speed up computations, we first extract the locality-based $\top\bot^*$ Σ-module (Cuenca Grau et al. 2008; Sattler, Schneider, and Zakharyaschev 2009) for a given TBox $\mathcal{T}$. The locality-based module entails the same Σ-inclusions as the TBox $\mathcal{T}$ but it is often considerably smaller in size. We also rely on ontologies to have structure: if a concept name occurs in several inclusions, it is likely that it occurs in the same syntactic pattern. We therefore transform clause sets as follows.

1. If the clause set contains some clauses $\mathcal{C}_1 \sqcup \mathcal{D}_\Upsilon, \ldots, \mathcal{C}_m \sqcup \mathcal{D}_\Upsilon$ such that for every $1 \leq i \leq m$ we have $\mathsf{sig}(\mathcal{C}_i) \cap \Upsilon = \emptyset$, we rewrite them into $X \sqcup \mathcal{D}_\Upsilon$, where $X \equiv \mathcal{C}_1 \sqcap \ldots \sqcap \mathcal{C}_m$, perform forgetting on Υ symbols and then replace X with its definition.
2. If the clause set contains a clause $\mathcal{C} \sqcup \exists r.(\mathcal{F}_\Upsilon, \mathcal{G}_1) \sqcup \ldots \sqcup \exists r.(\mathcal{F}_\Upsilon, \mathcal{G}_m)$ where $\mathsf{sig}(\mathcal{G}_i) \cap \Upsilon = \emptyset$ for every $1 \leq i \leq m$, we rewrite it into $\mathcal{C} \sqcup \exists r.(\mathcal{F}_\Upsilon, Y)$, where $Y \equiv \mathcal{G}_1 \sqcup \ldots \sqcup \mathcal{G}_m$, perform forgetting on Υ and then replace Y with its definition.

Experimental setting. All experiments were conducted on PCs equipped with an Intel Core i5-2500K CPU running at 3.30GHz. 15 GiB of RAM were allocated to the Java VM and an execution timeout of 60 CPU minutes was imposed on each problem. Whenever necessary we pre-processed the ontologies we used for our experiments as follows. For a given ontology $\mathcal{T}$, we first rewrote concept disjointness statements and role domain/range restrictions into $\mathcal{ALC}$ inclusions and then removed any remaining axiom which contained non-$\mathcal{ALC}$ concept (or role) constructors to obtain the $\mathcal{ALC}$-fragment of $\mathcal{T}$. We used Algorithm 1 to forget concept names one by one i.e. for $\Upsilon = \{A_1, \ldots, A_n\}$, Algorithm 1 was applied iteratively on $A_1, \ldots, A_n$, and we did *not* impose a bound on the depth of clauses; so the computed clause

	Computing $\mathsf{Diff}_\Sigma(\mathcal{T}_{i+1}, \mathcal{T}_i)$			Computing $\mathsf{Diff}_\Sigma(\mathcal{T}_i, \mathcal{T}_{i+1})$		
	Successful/ Total Runs	Success Rate (%)	Average # of Witnesses	Successful/ Total Runs	Success Rate (%)	Average # of Witnesses
BDO	3/5	60	12.33	5/5	100	211.40
CHEMINF	25/26	96	7.00	26/26	100	2.26
COGAT	4/4	100	272.00	3/4	75	4.00
JERM	8/13	61	7.00	9/13	69	9.33
NCI	101/108	93	787.10	105/108	97	906.20
NEMO	14/15	93	13.35	15/15	100	33.46
NPO	12/18	66	27.08	12/18	66	5.58
OMRSE	11/11	100	0.54	11/11	100	0.00
OPL	4/4	100	18.75	4/4	100	2.25
SIO	18/35	51	0.00	19/35	54	0.00

Table 3: Computing the Logical Difference between Ontology Versions on their Common Signature.

	$\lvert\Sigma \cap \mathsf{N_C}\rvert$	Success Rate (%)	Avrg # Axioms
DIKB	5	85	7.482
	10	60	14.033
	15	44	25.114
NCI	5	82	1.62
	10	64	2.65
	50	65	21.369
	100	56	41.089
	150	41	63.146

Table 2: Computing Uniform Interpolants of DIKB v1.4 and of NCI v08.10e Limited to Expansion Depth 3.

sets contain depth n-bounded uniform interpolants for every $n > 0$. Thus, in all the experiments reported on we computed *true* Σ*-uniform interpolants* (i.e. not a depth-bounded variant). The correctness of our extensions to Algorithm 1 can be shown by model-theoretic arguments.

Experiments with small signatures. We applied our uniform interpolation tool to compute uniform interpolants w.r.t. small concept signatures $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ with $\mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R} = \Sigma \cap \mathsf{N_R}$ for 21 small to medium size ontologies taken from the BioPortal repository[2]. The number of axioms in the selected ontologies ranges from 192 (for the *Ontology of Medically Related Social Entries*) to 2702 (for the *Subcellular Anatomy Ontology*). To make the experiments more interesting, we also included version 08.10e of the *National Cancer Institute Thesaurus* (NCI). For each considered sample size x and terminology $\mathcal{T}$ we generated 100 signatures Σ by randomly choosing x concept names from $\mathsf{sig}(\mathcal{T})$ and by adding all the role names from $\mathsf{sig}(\mathcal{T})$ to Σ. The results that we obtained are shown in Table 1.

In the left half of Table 1 one can see that the number of successful computations decreased while the size of $\Sigma \cap \mathsf{N_C}$ was increasing, which seems to be due to the fact that the $\top\bot^*$ Σ-modules then contain more symbols that lead to a large number of inferences. Most uniform interpolants that we obtained are relatively small and contain a lot of expressions of the form $\exists r_1 \ldots \exists r_n.\top$. In some cases the process of forgetting certain intermediate concept names generated a few hundred clauses that were simplified or deleted in the remaining computation steps. The success rate, however, varied significantly from one ontology to another. To further investigate this phenomenon, we computed uniform interpolants for a *fragment* of version 08.10e of NCI and for a fragment of version 1.4 of the *Drug Interaction Knowledge Base* (DIKB) that are of expansion depth 3 (that is, we removed all the axioms from both ontologies that led to an expansion depth greater than 3). The resulting DIKB fragment is a small acyclic terminology that contains 120 concept names, 27 roles names, and 127 axioms. The NCI fragment is also an acyclic terminology with 53571 concept names, 78 role names and 62494 axioms (of which 2362 are of the form $A \equiv C$). The results obtained are shown in Table 2. Limiting the expansion depth drastically improved the performance of our prototype implementation with the success rate for signatures containing 5 randomly selected concept names rising from 2% to 85% in the case of DIKB and from 23% to 82% in the case of NCI. For NCI our tool was capable of handling signatures containing up to 150 randomly selected concept names.

As proof of concept for ontology obfuscation, we applied our uniform interpolation tool on (a fragment of) the *Lipid Ontology* (LIPRO) to forget 45 concept names which are intermediate concept names in the ontology's induced concept hierarchy, i.e. those concept names group certain subconcepts together to give structure to the ontology. LIPRO is an acyclic terminology with 593 axioms, 574 concept names and one role name. The maximal size of an axiom is 50. It then took 192 CPU seconds to compute the uniform interpolant, which contains 3415 axioms that have a maximal size of 283. The uniform interpolant that we computed thus approximately contains 6 times more axioms than the origi-

[2] All ontologies used for the experiments reported on in this section can be accessed from the BioPortal repository: http://bioportal.bioontology.org/ontologies

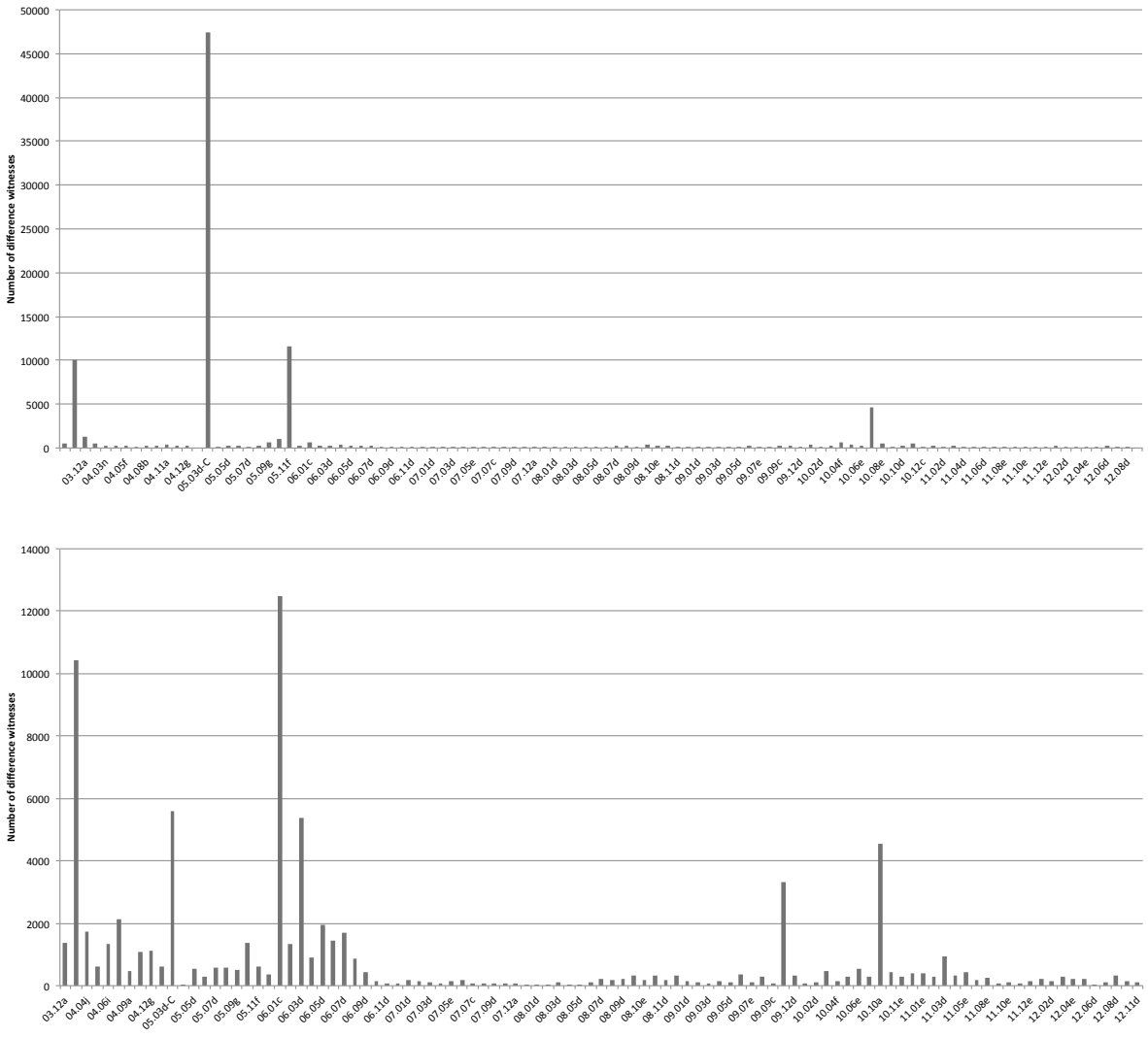


Figure 2: Logical Difference between NCI Versions i and $i+1$ (Top) and Versions $i+1$ and i (Bottom).

nal ontology and the maximal axiom size has increased by a factor of 6 as well. Notice that most of the original structure of the ontology has been destroyed while preserving all the consequences entailed by the retained concept names.

Finally, in the right half of Table 1 we report on our success rate for forgetting a small number of concept names. Notice that our prototype implementation performs significantly better in this scenario. This observation suggests that our tool is suitable for checking whether a change made to an ontology interferes with the meaning of the terms outside the (typically small) fragment under consideration in the context of computing the logical difference between two versions of an ontology.

Computing the Logical Difference. We selected 10 ontologies that have at least 5 submissions and whose expressivity is at least $\mathcal{ALC}$, including 109 versions of the NCI Thesaurus, from the BioPortal repository.

For every pair of consecutive versions $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$, where version $i+1$ represents the more recent version, and every considered signature Σ, we computed both $\mathsf{Diff}_\Sigma(\mathcal{T}_i, \mathcal{T}_{i+1})$ and $\mathsf{Diff}_\Sigma(\mathcal{T}_{i+1}, \mathcal{T}_i)$. We used the reasoner FaCT++ v1.6.2 (Tsarkov and Horrocks 2006) to determine whether any axiom $C \sqsubseteq D \in \mathcal{T}_i^{(\Sigma)}$ is a witness of $\mathsf{Diff}_\Sigma(\mathcal{T}_i, \mathcal{T}_{i+1})$, where $\mathcal{T}_i^{(\Sigma)}$ is the Σ-uniform interpolant of $\mathcal{T}_i$ computed with our tool (similarly for $\mathsf{Diff}_\Sigma(\mathcal{T}_{i+1}, \mathcal{T}_i)$). Note that the signatures Σ we used for the experiments contained (at least) all the roles present in $\mathcal{T}_i$ due to the limitations of our uniform interpolation procedure. However, such a restriction does not limit the applicability of logical difference in the sense that differences involving role names can still be detected. Note further that the results we obtained here are not directly comparable with the logical difference computed for description logics of the $\mathcal{EL}$ family (Konev, Walther, and Wolter 2008; Konev et al. 2012) as illustrated

$\lvert(\mathsf{sig}(\mathcal{T}) \setminus \Sigma) \cap \mathsf{N_C}\rvert$	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v08.09d}}, \mathrm{NCI}_{\text{v08.10e}})$ Success Rate (%)	Avrg # Witnesses	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v05.03d}}, \mathrm{NCI}_{\text{v05.05d}})$ Success Rate (%)	Avrg # Witnesses	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v05.12f}}, \mathrm{NCI}_{\text{v06.01c}})$ Success Rate (%)	Avrg # Witnesses
5	100	446.01	100	47 458.14	100	11 564.71
10	99	446.05	100	47 456.66	97	11 595.85
20	100	445.95	100	47 453.26	94	11 671.79
50	88	445.73	100	47 436.72	84	11 849.16
100	88	445.67	100	47 403.76	70	12 468.64

$\lvert(\mathsf{sig}(\mathcal{T}) \setminus \Sigma) \cap \mathsf{N_C}\rvert$	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v08.10e}}, \mathrm{NCI}_{\text{v08.09d}})$ Success Rate (%)	Avrg # Witnesses	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v05.05d}}, \mathrm{NCI}_{\text{v05.03d}})$ Success Rate (%)	Avrg # Witnesses	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v06.01c}}, \mathrm{NCI}_{\text{v05.12f}})$ Success Rate (%)	Avrg # Witnesses
5	98	2338.89	96	1347.92	99	13 704.29
10	98	2338.45	98	1348.47	100	13 788.15
20	97	2347.08	95	1348.66	95	13 841.52
50	92	2340.72	86	1351.56	87	14 062.52
100	86	2385.88	74	1354.04	80	14 504.40

Table 4: Computing the Logical Difference between Versions of NCI.

by Example 13.

In our first experiment we used $\Sigma = (\mathsf{sig}(\mathcal{T}_i) \cap \mathsf{sig}(\mathcal{T}_{i+1})) \cup \mathsf{N_R}$. This test captures any change to the meaning of the terms common to both versions. The results of computing the logical difference are given in Table 3. Notice that the success rate of computing $\mathsf{Diff}_\Sigma(\mathcal{T}_i, \mathcal{T}_{i+1})$ was slightly higher than the one of the converse direction. This observation can probably be attributed to the fact that these cases correspond to knowledge contained in an older version being removed from a newer one, which does not seem to happen often.

Interestingly, we could observe one of the highest success rates among all our experiments whilst computing logical differences for distinct versions of NCI. This can possibly be explained by the fact that versions of NCI are released frequently and changes to the ontology are hence introduced gradually. Figure 2 depicts the number of witnesses that correspond to the logical difference between consecutive versions of NCI on their common signature. Gonçalves, Parsia, and Sattler (2012) provide a comprehensive analysis of the changes between 14 consecutive versions of NCI using various techniques, ranging from a manual inspection of the log files to approximations of the logical difference. Versions 05.12f, 06.01c, and 06.08d were identified as having the highest number of differences. In our experiments, the highest number of logical difference witnesses were also present in NCI version 06.01c; the computations for versions 05.12f and 06.08d did not finish in time.

Furthermore, to make the experiments more challenging for the reasoner, we focused on comparing version i with version $i + 1$, and vice versa, on the 2 pairs of NCI versions for which the highest number of difference witnesses was identified in the first experiment. We also included version 08.10e as this is the last acyclic $\mathcal{ALC}$ TBox in the corpus. We performed tests on randomly generated large signatures Σ with $\Sigma \cap \mathsf{N_R} = \mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R}$. In that way the computed uniform interpolants remained rather large as well.

For each sample size $x \in \{5, 10, 20, 50, 100\}$ we generated 100 signatures by randomly choosing $|\mathsf{sig}(\mathcal{T}) \cap \mathsf{N_C}| - x$ concept names from $\mathsf{sig}(\mathcal{T})$ and by including all the role names from $\mathsf{sig}(\mathcal{T})$. The results that we obtained are now shown in Table 4.

One can observe that as the size of $\mathsf{sig}(\mathcal{T}) \setminus \Sigma$ increased, i.e. more symbols had to be forgotten from the $\top\bot^*$ Σ-modules, the success rate dropped slightly. Overall, the average number of witnesses and the average maximal size of the witnesses remained comparable throughout the different sample sizes. Also, the axioms generated by the computation of the uniform interpolant did not pose a problem for FaCT++ as computing the logical difference for a given signature never took more than 20 seconds in our experiments.

Conclusion

In this paper we presented an approach based on clausal resolution for computing uniform interpolants of $\mathcal{ALC}$-TBoxes $\mathcal{T}$ w.r.t. signatures $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ that contain all the role names present in $\mathcal{T}$. We proved that whenever the saturation process under $\mathcal{ALC}$-resolution terminates, the algorithm computes a uniform interpolant. To guarantee termination on all inputs, we introduced a depth-bounded version of our algorithm. We showed that by choosing an appropriate bound on the depth of clauses, one can axiomatise all Σ-inclusions implied by the given TBox up to a specified depth. Combined with a known bound on the size of uniform interpolants, our depth-bounded procedure always computes a uniform interpolant if it exists.

In the second part of this paper we investigated how often our unrestricted resolution-based algorithm terminates with a uniform interpolant by applying our prototype implementation on a number of case studies. Our findings suggest that despite a high computational complexity uniform interpolants can be computed in many practical cases. The com-

putation procedure could further benefit from better redundancy elimination techniques, which, together with extending our approach to forgetting role names, constitutes future work. It would also be interesting to explore proof strategies for our resolution calculi that guarantee termination when uniform interpolants exist.

References

Auffray, Y.; Enjalbert, P.; and Hébrard, J.-J. 1990. Strategies for modal resolution: Results and problems. *Journal of Automated Reasoning* 6(1):1–38.

Bachmair, L., and Ganzinger, H. 2001. Resolution theorem proving. In *Handbook of automated reasoning*, volume 1. Elsevier. chapter 2, 19–99.

Collberg, C. S.; Thomborson, C. D.; and Low, D. 1998. Manufacturing cheap, resilient, and stealthy opaque constructs. In *Proceedings of POPL '98*, 184–196. ACM.

Conradi, R., and Westfechtel, B. 1998. Version models for software configuration management. *ACM Computing Surveys (CSUR)* 30(2):232–282.

Cuenca Grau, B.; Horrocks, I.; Kazakov, Y.; and Sattler, U. 2008. Modular reuse of ontologies: theory and practice. *Journal of Artificial Intelligence Research (JAIR)* 31:273–318.

Eiter, T.; Ianni, G.; Schindlauer, R.; Tompits, H.; and Wang, K. 2006. Forgetting in managing rules and ontologies. In *Proceedings of the 2006 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2006)*, 411–419. IEEE Computer Society.

Gonçalves, R. S.; Parsia, B.; and Sattler, U. 2012. Concept-based semantic difference in expressive description logics. In *Proceedings of ISWC 2012*, volume 7649 of *LNCS*, 99–115. Springer.

Herzig, A., and Mengin, J. 2008. Uniform interpolation by resolution in modal logic. In *Proceedings of JELIA 2008*, volume 5293 of *LNCS*, 219–231. Springer.

Jiménez-Ruiz, E.; Grau, B. C.; Horrocks, I.; and Llavori, R. B. 2009. ContentCVS: A CVS-based collaborative ontology engineering tool. In *Proceedings of the Workshop on Semantic Web Applications and Tools for Life Sciences*, volume 559 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Jiménez-Ruiz, E.; Cuenca Grau, B.; Horrocks, I.; and Llavori, R. B. 2011. Supporting concurrent ontology development: Framework, algorithms and tool. *Data & Knowledge Engineering* 70(1):146–164.

Klein, M. C. A.; Fensel, D.; Kiryakov, A.; and Ognyanov, D. 2002. Ontology versioning and change detection on the web. In *Proceedings of EKAW 2002*, volume 2473 of *LNCS*. Springer. 247–259.

Konev, B.; Ludwig, M.; Walther, D.; and Wolter, F. 2012. The logical difference for the lightweight description logic $\mathcal{EL}$. *Journal of Artificial Intelligence Research (JAIR)* 44:633–708.

Konev, B.; Walther, D.; and Wolter, F. 2008. The logical difference problem for description logic terminologies. In *Proceedings of IJCAR 2008*, volume 5195 of *LNCS*, 259–274. Springer.

Konev, B.; Walther, D.; and Wolter, F. 2009. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proceedings of IJCAI 2009*, 830–835.

Koopmann, P., and Schmidt, R. A. 2013. Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In *Proceedings of FroCoS 2013*, volume 8152 of *LNCS*. Springer.

Ludwig, M., and Konev, B. 2013. Towards practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes. In *Proceedings of DL 2013*, volume 1014 of *CEUR Workshop Proceedings*, 377–389. CEUR-WS.org.

Lutz, C., and Wolter, F. 2011. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proceedings of IJCAI 2011*, 989–995.

Lutz, C.; Seylan, I.; and Wolter, F. 2012. An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In *Proceedings of KR 2012*. AAAI Press.

Nikitina, N., and Rudolph, S. 2012. ExpExpExplosion: Uniform interpolation in general $\mathcal{EL}$ terminologies. In *Proceedings of ECAI 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, 618–623. IOS Press.

Noy, N. F., and Musen, M. A. 2002. PromptDiff: A fixed-point algorithm for comparing ontology versions. In *Proceedings of AAAI 2002*, 744–750. AAAI Press.

Reiter, R., and Lin, F. 1994. Forget it! In *Proceedings of AAAI Fall Symposium on Relevance*, 154–159. AAAI Press.

Sattler, U.; Schneider, T.; and Zakharyaschev, M. 2009. Which kind of module should I extract? In *Proceedings of DL 2009*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org.

ten Cate, B.; Conradie, W.; Marx, M.; and Venema, Y. 2006. Definitorially complete description logics. In *Proceedings of KR 2006*, 79–89. AAAI Press.

Tsarkov, D., and Horrocks, I. 2006. FaCT++ description logic reasoner: System description. In *Proceedings of IJCAR 2006*, volume 4130 of *LNCS*, 292–297. Springer.

Wang, Z.; Wang, K.; Topor, R.; and Pan, J. Z. 2008. Forgetting concepts in DL-Lite. In *Proceedings of ESWC2008*, volume 5021 of *LNCS*, 245–257. Springer.

Wang, Z.; Wang, K.; Topor, R. W.; and Zhang, X. 2010. Tableau-based forgetting in $\mathcal{ALC}$ ontologies. In *Proceedings of ECAI 2010*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, 47–52. IOS Press.

Wang, K.; Wang, Z.; Topor, R.; Pan, J. Z.; and Antoniou, G. 2012. Eliminating concepts and roles from ontologies in expressive descriptive logics. *Computational Intelligence*. DOI: 10.1111/j.1467-8640.2012.00442.x.

Zhou, Y., and Zhang, Y. 2011. Bounded forgetting. In *Proceedings of AAAI 2011*. AAAI Press.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Certain Answers as Objects and Knowledge

Leonid Libkin
School of Informatics, University of Edinburgh

Abstract

The standard way of answering queries over incomplete databases is to compute *certain answers*, defined as the intersection of query answers on all complete databases that the incomplete database represents. But is this universally accepted definition correct? We argue that this "one-size-fits-all" definition can often lead to counterintuitive or just plain wrong results, and propose an alternative framework for defining certain answers.

The idea of the framework is to move away from the standard, in the database literature, assumption that query results be given in the form of a database object, and to allow instead two alternative representations of answers: as objects defining all other answers, or as knowledge we can deduce with certainty about all such answers. We show that the latter is often easier to achieve than the former, that in general certain answers need not be defined as intersection, and may well contain missing information in them. We also show that with a proper choice of semantics, we can often reduce computing certain answers – as either objects or knowledge – to standard query evaluation. We describe the framework in the most general way, applicable to a variety of data models, and test it on three concrete relational semantics of incompleteness: open, closed, and weak closed world.

1 Introduction

Handling incomplete information is one of the oldest topics in database research. It has been tackled both from the database perspective, resulting in classical notions of the semantics and complexity of query evaluation (Abiteboul, Kanellakis, and Grahne 1991; Imielinski and Lipski 1984), and from the AI perspective, providing an alternative view of the problem, see, e.g., (Reiter 1982; Lenzerini 1991). With the shifting focus in database applications, owing to the ever increasing amounts of data as well as data heterogeneity, the problem of incomplete information is becoming much more pronounced. It appears in many important application areas such as data integration (Lenzerini 2002), data exchange (Arenas et al. 2014), inconsistent databases (Bertossi 2011), probabilistic data (Suciu et al. 2011), and data quality (Fan and Geerts 2012).

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The central problem in the database field is of course query answering, and in the presence of incompleteness, one looks for *certain answers*: those that do not depend on the interpretation of unknown data. The concept was first mentioned in (Grant 1977) and formally defined 35 years ago in (Lipski 1979) as follows. Assume that the semantics $[\![D]\!]$ of an incomplete database D is given as the set of all complete databases D' which D can represent. Then the certain answer to Q on D is defined as $\mathsf{cert}_\cap(Q, D) = \bigcap\{Q(D') \mid D' \in [\![D]\!]\}$. Since database queries produce collections (sets, multisets, etc.), it makes sense to talk about their intersection. This definition has been universally applied to all the semantics of incompleteness, and in all the scenarios such as those listed above. The intuition is that this gives us the set of tuples independent of the interpretation of the missing information in D. Such a universal adoption of this basic definition has another consequence: certain answers themselves contain no missing information. In fact many algorithms for computing certain answers have, as the last step, elimination of any objects (say, rows in relational databases) with missing data.

The question that we address here is the following: *Is this standard "one-size-fits-all" definition really the right one to use for all the semantics, and all the applications?* The answer, as we shall argue, is negative: the standard intersection semantics, as well as the assumption that no missing information is present in the answers, leads to many problems, and crucially to producing meaningless query answers.

To argue that this is the case, and to explain some basic ideas behind the alternative approach we present here, note that in the database field, one tends to operate with *objects* (i.e., relations, XML documents, graph databases, etc.); in particular, queries take objects and return objects. Thus, the idea behind certain answers is to find an *object* A representing the *set of objects* $Q([\![D]\!]) = \{Q(D') \mid D' \in [\![D]\!]\}$. Such an object A must contain information common to all the objects in $Q([\![D]\!])$: that is, it must be no more informative than any of the objects in $Q([\![D]\!])$.

Now take a simple example: we have a relation $R = \{(1, 2), (3, \bot)\}$ in a database, where $\bot$ represents a null, or a missing value. The query Q returns R itself. Then $\mathsf{cert}_\cap(Q, D) = \{(1, 2)\}$ under every reasonable semantics of incompleteness. But is it less informative than all of $Q(D')$ for $D' \in [\![D]\!]$? The answer depends on the semantics. Under

the common *open-world* semantics, the answer is yes: in fact $(1, 2)$ is precisely the greatest lower bound of $Q([\![D]\!])$ under the ordering whose meaning is "being less informative". But under the equally common *closed-world* semantics, the answer is no. Even worse, $(1, 2)$ is not less informative than any of the answers $Q(D')$ for $D' \in [\![D]\!]$ which are of the form $\{(1, 2), (3, n)\}$ for different values n. Indeed, under the closed world semantics, the answer $\{(1, 2)\}$ contains *additional* information that no tuple except $(1, 2)$ is present. Thus, returning just $(1, 2)$ in this case makes no sense at all.

The problem with returning the single tuple $(1, 2)$ as the certain answer becomes even more pronounced if we follow the approach, pioneered by (Reiter 1982), that views databases as logical theories and query answering as logical implication. The fact $R(1, 2)$ is certainly implied by the database. But is it the only fact that is implied? Of course not: under the open-world semantics, we can deduce $\exists x\, R(1, 2) \wedge R(3, x)$ with certainty, adding the fact that there is a tuple whose first component is 3. And under the closed world semantics, we know for certain even more, for instance $\exists x \forall y\ (y = 1 \vee y = 2 \vee y = 3 \vee y = x)$, since we cannot expand the database.

Even from this simple example, we learn a few lessons:

- Certain answers can be presented as both objects and logical formulae;
- they depend on both logical languages and semantics used; and
- taking intersection and removing missing values from the answers is not always the right way to compute them.

Note that viewing answers as formulae brings us closer to knowledge bases. The difference though is that while in databases we start with objects and produce objects, and in querying knowledge bases we start with logical theories and produce logical theories, here we may also combine the two, starting with an object, and returning a formula, or a theory, as the result.

The goal of this paper is to develop an alternative framework for handling certain answers to queries. For that, we combine the approaches to viewing databases as objects (Imielinski and Lipski 1984) and as logical theories (Reiter 1982), rather than treat them separately. Specifically, the key elements of our framework are as follows.

- Certain answers can be viewed as either objects or theories, depending on the semantics, and the logical formalism used. The former is in line with the standard database approach, while the latter defines *certain knowledge* about query answers over incomplete databases.
- Both ways are based on extracting certain information from a set of objects. Each way defines certainty as a greatest lower bound: either of a set of objects, or the *theory* of that set of objects, with the ordering meaning "being more informative".
- Proper query answering is based on the notion of *representation systems*: these are a natural relaxation of a rather restrictive concept of what database people call strong representation systems. Representation systems let one define important sets of objects by logical formulae, in the spirit of (Reiter 1982).
- Under the choice of the right semantics for both query inputs and query answers, certain answers – as both objects and knowledge – can be found by straightforward database query evaluation. Thus, with the correct choice of semantics and representation system, we can use *existing* query evaluation techniques for obtaining correct answers in the presence of incomplete information.
- In general, it is easier to find certain answers as knowledge as opposed to certain answers as objects: sometimes the former exists and the latter does not. Thus, representing certain knowledge about query answers is not a mere convenience, it may well be a necessity.

Most of the results in the paper are shown in an abstract setting, for two reasons. First, it makes them applicable to other data models, beyond relational databases (e.g., XML and graph data). Second, it helps us see the essential conditions that need to be imposed on queries and the semantics of incompleteness, without being too "clouded" by details of a particular data model. At the same time we use three common relational semantics – open world, closed world, and weak closed world – to translate general results into concrete examples, to test the approach.

Organization. After presenting basic facts about relational incompleteness in Section 2, we introduce the abstract setting and notions of certainty as objects and knowledge in Section 3. Section 4 defines representation systems and connects knowledge certainty with greatest lower bounds. Section 5 defines certain answers for queries, and Section 6 shows how to compute them. Summary and future work are in Section 7.

2 Background: relational incompleteness

Incomplete databases These have two types of values: *constants* (e.g., $1, 2, \ldots$) and *nulls*. We thus assume countably infinite sets of constants, denoted by Const, and of nulls, denoted by Null. Nulls themselves are denoted by $\bot$, sometimes with sub- or superscripts.

A relational *vocabulary* (often called *schema* in database literature) is a set of relation names with associated arities. An incomplete relational instance D assigns to each k-ary relation symbol R from the vocabulary a k-ary relation R^D over $\mathsf{Const} \cup \mathsf{Null}$, i.e., a finite subset of $(\mathsf{Const} \cup \mathsf{Null})^k$. If the instance D is clear from the context, we may write R instead of R^D. Sets of constants and nulls that occur in D are denoted by $\mathsf{Const}(D)$ and $\mathsf{Null}(D)$. The *active domain* of D is $\mathsf{adom}(D) = \mathsf{Const}(D) \cup \mathsf{Null}(D)$. A *complete* database D has no nulls, i.e., $\mathsf{adom}(D) \subseteq \mathsf{Const}$.

Semantics and valuations A *valuation* of nulls on an incomplete database D is a map $v : \mathsf{Null}(D) \to \mathsf{Const}$ assigning a constant value to each null. It naturally extends to databases, so we can write $v(D)$ as well. The standard semantics of incompleteness in relational databases are defined in terms of valuations, see (Abiteboul, Hull, and Vianu 1995; Imielinski and Lipski 1984). These are the *closed world assumption*, or CWA semantics:

$$[\![D]\!]_{\text{CWA}} = \{v(D) \mid v \text{ is a valuation}\},$$

and the *open-world assumption*, or OWA semantics:

$$[\![D]\!]_{\text{OWA}} = \left\{ D' \;\middle|\; \begin{array}{l} D' \text{ is complete and} \\ v(D) \subseteq D' \text{ for some valuation } v \end{array} \right\}.$$

We shall also consider the *weak* CWA, or WCWA semantics, inspired by (Reiter 1980), given by

$$[\![D]\!]_{\text{WCWA}} = \left\{ v(D) \cup D' \;\middle|\; \begin{array}{l} v \text{ is a valuation and} \\ \text{adom}(D') \subseteq \text{adom}(v(D)) \end{array} \right\}.$$

That is, under CWA, we simply instantiate nulls by constants; under OWA, we can also add arbitrary tuples, and under WCWA, we can only add tuples formed by elements already present. For instance, if $D_0 = \{(\bot, \bot')\}$, then $[\![D]\!]_{\text{CWA}}$ only has instances $\{(c, c')\}$ for $c, c' \in \mathsf{Const}$, while $[\![D]\!]_{\text{WCWA}}$ can have in addition instances that add to (c, c') tuples $(c, c), (c', c')$, and (c', c), and $[\![D]\!]_{\text{OWA}}$ has all instances containing at least one tuple.

Information orderings With an arbitrary semantics $[\![\,]\!]$ of incompleteness, we can associate an *information ordering*

$$D \preceq D' \;\Leftrightarrow\; [\![D']\!] \subseteq [\![D]\!].$$

The meaning is that D' is more informative than D. Indeed, the more possible objects we have in the semantics of an incomplete database, the less we know about it. Note that $\preceq$ is a preorder, i.e., it is reflexive and transitive, but both $D \preceq D'$ and $D' \preceq D$ may be true if $[\![D]\!] = [\![D']\!]$.

We write $\preceq_*$ for the ordering given by $[\![\,]\!]_*$, where $*$ is one of CWA, WCWA, or OWA. These can be described in terms of homomorphisms. Given two relational databases D and D', a homomorphism $h : D \to D'$ is a map from $\text{adom}(D)$ to $\text{adom}(D')$ such that $h(c) = c$ for each constant c, and such that for every relation symbol R and a tuple $\bar{t} \in R^D$, the tuple $h(\bar{t})$ is in $R^{D'}$. The image of the homomorphism is denoted by $h(D)$. A homomorphism is *onto* if $\text{adom}(h(D)) = \text{adom}(D')$, and *strong onto* if $D' = h(D)$.

It is known (Libkin 2011; Gheerbrant, Libkin, and Sirangelo 2013) that $D \preceq_* D'$ iff there exists a

- homomorphism $h : D \to D'$, for $* =$OWA;
- onto homomorphism $h : D \to D'$, for $* =$WCWA;
- strong onto homomorphism $h : D \to D'$, for $* =$CWA.

Classical definition of certain answers Given an incomplete database D, a semantics of incompleteness $[\![\,]\!]$, and a query Q, the standard notion of certain answers under $[\![\,]\!]$ is

$$\mathsf{cert}_\cap(Q, D) = \bigcap \{ Q(R) \mid R \in [\![D]\!] \}.$$

Note that $\mathsf{cert}_\cap(Q, D)$ cannot contain any tuples with nulls.

In some cases, these answers are obtained by almost straightforward query evaluation, namely by evaluating $Q(D)$ and then throwing away the tuples with nulls. We shall denote this by $Q^{\mathsf{C}}(D)$. For instance, if the query Q just returns a relation R, and $R^D = \{(1, 2), (1, \bot)\}$, then both $Q^{\mathsf{C}}(D)$ and $\mathsf{cert}_\cap(Q, D)$ are $\{(1, 2)\}$. In general of course the two need not coincide, as finding certain answers may be computationally very expensive: for instance, undecidable under OWA, or coNP-hard under CWA for first-order queries (Abiteboul, Kanellakis, and Grahne 1991).

Logics Most of database query languages are based on *first-order predicate logic*, or FO, whose formulae are built from relational atoms $R(\bar{x})$, where R is a vocabulary symbol, equational atoms $x = y$, and are closed under Boolean connectives $\wedge, \vee, \neg$ and quantifiers $\exists$ and $\forall$.

The fragment that disallows $\neg$ and $\forall$ (i.e., has $\wedge, \vee, \exists$) is referred to as *existential positive formulae*, denoted by $\exists\mathsf{Pos}$. In terms of their expressiveness, they correspond precisely to *unions of conjunctive queries* (although $\exists\mathsf{Pos}$ formulae can be more compact).

The fragment without negation (i.e., the $\wedge, \vee, \exists, \forall$ fragment) is referred to as *positive formulae*, denoted by Pos.

It is known that $\mathsf{cert}_\cap(Q, D) = Q^{\mathsf{C}}(D)$ for $\exists\mathsf{Pos}$ queries under OWA (Imielinski and Lipski 1984), and for Pos queries under WCWA (Gheerbrant, Libkin, and Sirangelo 2013). There is a fragment for which the equality holds under CWA as well, and it will be given later.

3 Objects and knowledge

The key idea, as explained in the introduction, is to decouple objects and their descriptions in terms of some logical formalisms, i.e., to decouple objects and knowledge about them. We want to do this at the highest level of abstraction, so that the framework would not be limited to just relational databases, but instead would be applicable across multiple data models. For this, we use a very minimalist setting inspired by abstract model theory (Barwise and Feferman 1985) or information systems (Gunter 1992), with some specific features tailored to handle incompleteness, as also used in (Libkin 2011; Gheerbrant, Libkin, and Sirangelo 2013).

We have two basic entities: (database) *objects*, and *formulae* they satisfy. Objects themselves are either incomplete or complete, and each object has its semantics defined as the set of more informative complete objects.

To formalize this, we define a *database pre-domain* as a triple $\mathbb{D}^\circ = \langle \mathcal{D}, \mathcal{C}, [\![\,]\!] \rangle$, where

- $\mathcal{D}$ is a set of database objects (e.g., relational databases over the same schema),
- $\mathcal{C}$ is the set of complete objects (e.g., databases without nulls);
- $[\![\,]\!]$ is a function from $\mathcal{D}$ to subsets of $\mathcal{C}$; the set $[\![x]\!] \subseteq \mathcal{C}$ is the semantics of an incomplete database x.
- We require that for $c \in [\![x]\!]$, two conditions hold: $c \in [\![c]\!]$ and $[\![c]\!] \subseteq [\![x]\!]$.

The last two conditions hold in the standard semantics of incompleteness: they say that a complete object should denote at least itself, and that we have less uncertainty about an object c in the semantics of x than about x itself.

As for relational semantics seen earlier, we have the *information ordering*

$$x \preceq y \;\Leftrightarrow\; [\![y]\!] \subseteq [\![x]\!].$$

Note that $c \in [\![x]\!]$ implies $x \preceq c$, i.e., an incomplete object is less informative than the objects it represents.

A *pre-representation system* is a triple $\mathbb{RS}^\circ = \langle \mathbb{D}^\circ, \mathbb{F}, \models \rangle$, where

- $\mathbb{D}^\circ$ is a pre-domain;

- $\mathbb{F}$ is a set of *formulae*, and
- $\models$ is the satisfaction relation, i.e., a subset of $\mathcal{D} \times \mathbb{F}$ such that $x \preceq y$ and $x \models \varphi$ imply $y \models \varphi$.

The intuition is that formulae in $\mathbb{F}$ express knowledge we possess about objects in $\mathcal{D}$, and if we know something about an object, we also know it about a more informative object.

We shall write $\mathsf{Th}(x)$ for the *theory* of x, i.e., $\{\varphi \mid x \models \varphi\}$ and $\mathsf{Mod}(\varphi)$ for models of φ, i.e., $\{x \mid x \models \varphi\}$. These are extended to sets in the usual way: $\mathsf{Th}(X) = \bigcap_{x \in X} \mathsf{Th}(x)$ and $\mathsf{Mod}(\Phi) = \bigcap_{\varphi \in \Phi} \mathsf{Mod}(\varphi)$.

Note that Mod and Th define a Galois connection between $\mathcal{D}$ and $\mathbb{F}$; in particular $\mathsf{Mod}(\mathsf{Th}(\cdot))$ is a closure operator.

Certain information

Computing certain answers boils down to finding certain information contained in a set of objects; in the case of query answering, in $Q([\![D]\!]) = \{Q(D') \mid D' \in [\![D]\!]\}$. Thus, we need to know how to define certain information contained in a set of objects $X \subseteq \mathcal{D}$. The usual database approach is to represent this information as another object, but of course we argue that it can be viewed as both object and knowledge.

Certain information as object If we want to represent what we know about X with certainty by an object y, this object must be less informative than any object $x \in X$ (as it reflects knowledge contained in all other objects in X as well). If we have two such objects y and y', and $y' \preceq y$, then of course we prefer y as giving us more information.

Thus, the object that we seek must be less informative than all objects in X, and at the same time the most informative among such objects. This is precisely the *greatest lower bound* of X, with respect to $\preceq$ (or $\bigwedge X$, using the standard order-theoretic notation). If it is exists, we denote it by $\Box_{\mathcal{O}} X$.

Certain information as knowledge We want to describe X by a single formula summarizing what we know about it with certainty. If $X = \mathsf{Mod}(\varphi)$, then φ is such a formula, but generally, X need not be of the form $\mathsf{Mod}(\varphi)$.

So we go for the next best thing: we want a formula that is equivalent to the *theory* of X. Of course two sets of formulae are equivalent when they have the same models, so a formula equivalent to the theory of X is a formula φ such that $\mathsf{Mod}(\varphi) = \mathsf{Mod}(\mathsf{Th}(X))$. We denote such a formula by $\Box_{\mathcal{K}} X$.

Thus, certain information contained in X is described as:

- At the object level as $\Box_{\mathcal{O}} X = \bigwedge X$; and
- At the knowledge level as a formula $\Box_{\mathcal{K}} X$ so that $\mathsf{Mod}(\Box_{\mathcal{K}} X) = \mathsf{Mod}(\mathsf{Th}(X))$.

We now make a couple observations about these concepts. First, neither $\Box_{\mathcal{O}} X$ nor $\Box_{\mathcal{K}} X$ need exist in general (in fact it is easy to come up with the examples of preorders without greatest lower bounds).

Second, even if they exist, they are not unique. This is not an issue, however, as they are *equivalent*. Since $\preceq$ is a preorder, the greatest lower bound is, technically speaking, a set of objects, but every two such objects y, y' are equivalent: $y \preceq y'$ and $y' \preceq y$, and thus $[\![y]\!] = [\![y']\!]$. If we have multiple formulae φ for which $\mathsf{Mod}(\varphi) = \mathsf{Mod}(\mathsf{Th}(X))$, then every two such formulae φ, φ' are equivalent: $\mathsf{Mod}(\varphi) = \mathsf{Mod}(\varphi')$. So we shall write $y = \Box_{\mathcal{O}} X$ or $\varphi = \Box_{\mathcal{K}} X$, meaning y or φ is one of the equivalent objects or formulae.

Also note that a rudimentary case of $\Box_{\mathcal{K}} X$ was used in (David, Libkin, and Murlak 2010), which looked at certain answers for XML queries. There, the set $\mathbb{F}$ of formulae was $\mathcal{D}$ itself, and $x \models y$ simply meant $y \preceq x$. Just in that simple case, Theorem 1 below is a consequence of (Libkin 2011).

Example Consider the example from the introduction, of a database D containing $(1, 2)$ and $(3, \bot)$ in a relation R. if $X = [\![D]\!]_{\text{OWA}}$, then $\Box_{\mathcal{O}} X$ is just D itself, as expected. If $\mathbb{F} = \exists\mathsf{Pos}$, then $\Box_{\mathcal{K}} X = \exists z\, R(1, 2) \wedge R(3, z)$. If $\mathbb{F}$ is the set of ground facts and their conjunctions, then $\Box_{\mathcal{K}} X = R(1, 2)$.

If $X = [\![D]\!]_{\text{CWA}}$, then $\bigwedge X$ is still D, but we need a more complex class to describe $\Box_{\mathcal{K}} X$; in fact the two formulae above can easily be shown to violate the definition of $\Box_{\mathcal{K}} X$ in this case. We shall continue with this example in the next section and present a formula $\Box_{\mathcal{K}} X$ under CWA, that comes from a more expressive subclass of FO.

4 Representation systems

So far we imposed no conditions on pre-representation systems. We now define representation systems, which are pre-representation systems with two conditions imposed. These conditions, that hold in the standard semantics of incompleteness, say essentially that the sets of objects and formulae are not too "thin": there are enough complete objects, and there are formulae defining some fairly basic sets of objects.

There are enough objects To motivate this condition, consider a database D with a single tuple $(\bot, \bot)$. In its semantics, we are allowed to replace this tuple by an arbitrary tuple (c, c) with $c \in \mathsf{Const}$. Moreover, the resulting database is isomorphic to the original one: for instance, they agree on logical formulae not mentioning constants. Even if we have a logical formula mentioning constants from a finite set C, we can replace $\bot$ by constants outside this set C: this is what we mean by the existence of enough complete objects. Then D and the result of the replacement will still agree on formulae that only refer to constants in C.

To formalize this, define a *database domain* $\mathbb{D}$ as a tuple $\langle \mathcal{D}, \mathcal{C}, [\![\,]\!], \text{Iso}\rangle$ where $\langle \mathcal{D}, \mathcal{C}, [\![\,]\!]\rangle$ is a pre-domain, and Iso is a family $\{\approx_j\}_{j \in J}$ of equivalence relations on $\mathcal{D}$ so that:

- The set $[\![x]\!]_{\approx_j} = \{c \in [\![x]\!] \mid x \approx_j c\}$ is nonempty for each $x \in \mathcal{D}$ and $j \in J$;
- for every $j, j' \in J$, there is $k \in J$ so that $\approx_k \subseteq \approx_j \cap \approx_{j'}$.

Coming back to the relational intuition, $j \in J$ enumerate finite sets of constants $C_j \subset \mathsf{Const}$, and $D \approx_j D'$ means that there is an isomorphism between D and D' preserving constants in C_j. Then the first condition says that we can replace nulls by constants outside C_j (since $\mathsf{Const} - C_j$ is infinite), and the second one says that $C_k = C_j \cup C_{j'}$ preserves constants in both C_j and $C_{j'}$.

There are enough formulae First, we assume that formulae are closed under conjunction, i.e. for $\varphi, \psi \in \mathbb{F}$ we have $\varphi \wedge \psi \in \mathbb{F}$ with $\mathsf{Mod}(\varphi \wedge \psi) = \mathsf{Mod}(\varphi) \cap \mathsf{Mod}(\psi)$.

The second assumption is that the set $\uparrow x = \{y \mid x \preceq y\}$ of objects more informative than x can be defined by a formula. We shall see shortly that this is true in the classical incompleteness semantics, and that this is equivalent to finite axiomatizability of $\mathsf{Th}(x)$.

Formally the condition states that for each object x, there is a formula δ_x in $\mathbb{F}$, such that $\mathsf{Mod}(\delta_x) = \uparrow x$.

Representation systems A *representation system* is a triple $\mathbb{RS} = \langle \mathbb{D}, \mathbb{F}, \models \rangle$, where:

- $\mathbb{D}$ is a database domain and $\langle \mathbb{D}, \mathbb{F}, \models \rangle$ is a pre-representation system whose formulae are closed under conjunction;
- for each $x \in \mathcal{D}$, there is a formula δ_x with $\mathsf{Mod}(\delta_x) = \uparrow x$;
- for each $\varphi \in \mathbb{F}$, there is $j \in J$ so that $x \models \varphi \Leftrightarrow y \models \varphi$ whenever $x \approx_j y$.

The last condition is essentially the analog of the condition that a formula can only refer to finitely many constants, and thus cannot distinguish objects equivalent with respect to $\approx_j$ for some j.

Examples of domains and representation systems

We now provide examples of representation systems corresponding to relational OWA, WCWA, and CWA semantics. We use the notation $\mathcal{D}(\sigma)$ for the set of all relational databases of vocabulary σ over $\mathsf{Const} \cup \mathsf{Null}$, and $\mathcal{C}(\sigma)$ for the set of all such databases that do not use nulls in Null. The database domains will be of the form $\mathbb{D}_*(\sigma) = \langle \mathcal{D}(\sigma), \mathcal{C}(\sigma), [\![\,]\!]_*, \mathrm{Iso} \rangle$, where $*$ is one of OWA, WCWA, or CWA.

The equivalence relations Iso are as follows. Let J enumerate all the finite subsets of Const. Given such a finite set C, we have $D \approx_C D'$ if there is an isomorphism $f : D \to D'$ such that both f and f^{-1} are identity on C.

In what follows, we describe formulae $\mathbb{F}$ and formulae δ_D for each D. Let $\mathrm{PosDiag}(D)$ be the positive diagram of D in the vocabulary including constants for each $a \in \mathsf{Const}$, where with each null $\bot_i$ in D we associate a variable x_i. For instance, if D contains relation R with tuples $(1,2), (2,\bot_1), (\bot_1,\bot_2)$, then $\mathrm{PosDiag}(D) = R(1,2) \wedge R(2,x_1) \wedge R(x_1,x_2)$.

OWA The OWA representation system is $\mathbb{RS}_{\mathrm{OWA}}(\sigma) = \langle \mathbb{D}_{\mathrm{OWA}}(\sigma), \exists\mathsf{Pos}, \models \rangle$. For each D with $\mathsf{Null}(D) = \{\bot_1, \ldots, \bot_n\}$, we have $\delta_D = \exists x_1, \ldots, x_n\ \mathrm{PosDiag}(D)$.

WCWA The WCWA representation system is $\mathbb{RS}_{\mathrm{WCWA}}(\sigma) = \langle \mathbb{D}_{\mathrm{WCWA}}(\sigma), \mathsf{Pos}, \models \rangle$. For each D with $\mathsf{Const}(D) = \{a_1, \ldots, a_m\}$ and $\mathsf{Null}(D) = \{\bot_1, \ldots, \bot_n\}$, the formula δ_D is

$$\exists x_1 \ldots x_n \big(\mathrm{PosDiag}(D) \wedge \forall y\, (\bigvee_{i=1}^{m} y = a_i \ \vee \ \bigvee_{i=1}^{n} y = x_i)\big)$$

CWA We need to define an extension of the class of positive formulae, introduced by (Compton 1983) and used recently in (Gheerbrant, Libkin, and Sirangelo 2013). The class, denoted by $\mathsf{Pos}^{\forall \mathsf{G}}$, extends Pos with a special type of guarded formulae. It is defined as the closure of positive atoms of the form $R(\bar{x})$ and $x = y$ under $\wedge, \vee, \forall, \exists$ and the following rule: if $\varphi(\bar{x}, \bar{y})$ is a $\mathsf{Pos}^{\forall \mathsf{G}}$ formula in which all variables in $\bar{x}$ are distinct, and R is a relation symbol of the arity $|\bar{x}|$, then $\forall \bar{x}\ (R(\bar{x}) \to \varphi(\bar{x}, \bar{y}))$ is a $\mathsf{Pos}^{\forall \mathsf{G}}$ formula.

With this, the CWA representation system is defined as $\mathbb{RS}_{\mathrm{OWA}}(\sigma) = \langle \mathbb{D}_{\mathrm{CWA}}(\sigma), \mathsf{Pos}^{\forall \mathsf{G}}, \models \rangle$. For each D with $\mathsf{Null}(D) = \{\bot_1, \ldots, \bot_n\}$, the formula δ_D is

$$\exists x_1, \ldots, x_n \Big(\mathrm{PosDiag}(D) \wedge \bigwedge_{R \in \sigma} \forall \bar{y}\, \big(R(\bar{y}) \to \bigvee_{\bar{t} \in R^D} \bar{y} = \bar{t}\big)\Big),$$

where the length of $\bar{y}$ and $\bar{t}$ is the arity of R, and $\bar{y} = \bar{t}$ means $\bigwedge_{i \le \mathrm{arity}(R)} (y_i = t_i)$.

Proposition 1 *Each of $\mathbb{RS}_{\mathrm{OWA}}(\sigma)$, $\mathbb{RS}_{\mathrm{WCWA}}(\sigma)$, and $\mathbb{RS}_{\mathrm{CWA}}(\sigma)$ is a representation system.*

Proof sketch. Conditions on Iso are easily checked: sets $[\![D]\!]_C$ are nonempty, since one can replace nulls with distinct constants from $\mathsf{Const} - C$, and $\approx_{C \cup C'} \subseteq \approx_C \cap \approx_{C'}$. Monotonicity follows from the description of orderings $\preceq_*$ by the existence of homomorphisms (see Section 2) and the fact that $\exists\mathsf{Pos}$ (resp., Pos and $\mathsf{Pos}^{\forall \mathsf{G}}$) formulae are preserved under homomorphisms (resp., onto and strong onto), see (Rossman 2008; Chang and Keisler 1990; Compton 1983). And the properties of δ_D again follow from the properties of the ordering, as one can easily check that $\mathsf{Mod}(\delta_D)$ are the structures into which there is a homomorphism (resp., onto or strong onto homomorphism) from D. □

Properties of representation systems

We now collect some useful properties of representation systems. First, we look at certain information contained in sets $[\![x]\!]$ and prove that, as expected, it is represented at the object level by x, and at the knowledge level by δ_x.

Proposition 2 *In a pre-domain, $\Box_{\mathcal{O}}[\![x]\!] = x$ for every x.*

Proof. Assume that $x \neq \bigwedge [\![x]\!]$. Then there is $y \not\preceq x$ such that $y \preceq c$ for each $c \in [\![x]\!]$. Since $y \not\preceq x$ we have $[\![x]\!] \not\subseteq [\![y]\!]$, i.e., there is $c \in [\![x]\!]$ such that $c \notin [\![y]\!]$. But since $y \preceq c$, we have $[\![c]\!] \subseteq [\![y]\!]$, and since $c \in \mathcal{C}$, this implies $c \in [\![y]\!]$ as $c \in [\![c]\!]$, which gives us the desired contradiction. □

Let $\approx = \bigcup_{j \in J} \approx_j$, and let $[\![x]\!]_{\approx} = \{c \in [\![x]\!] \mid c \approx x\}$. In case of relational databases, $D \approx D'$ if D, D' are isomorphic objects; for instance, $D = \{(\bot, \bot)\}$ and $D' = \{(1,1)\}$ are isomorphic. Note that FO formulae not using constants are preserved by $\approx$. In general though, objects related by $\approx_j$ may not agree on all the formulae of $\mathbb{F}$ (e.g., D and D' do not agree on $\exists x\ (x = 1)$) and hence there are potentially formulae in $\mathsf{Th}([\![x]\!]_{\approx_j})$ which are not satisfied by x. However,

Proposition 3 *In a representation system, $\mathsf{Th}([\![x]\!]) = \mathsf{Th}([\![x]\!]_{\approx}) = \mathsf{Th}(x)$ for every x.*

Proof. It suffices to prove $\mathsf{Th}([\![x]\!]_{\approx}) = \mathsf{Th}(x)$ since $\mathsf{Th}(x) \subseteq \mathsf{Th}([\![x]\!]) \subseteq \mathsf{Th}([\![x]\!]_{\approx})$. Let $c \in [\![x]\!]_{\approx}$; then $x \preceq c$ and thus $\mathsf{Th}(x) \subseteq \mathsf{Th}(c)$, and hence $\mathsf{Th}(x) \subseteq \mathsf{Th}([\![x]\!]_{\approx})$. Conversely, take $\varphi \in \mathsf{Th}([\![x]\!]_{\approx})$. We know that there is $j \in J$ so that whenever $y \approx_j y'$, then y, y' agree on φ. Note that $\varphi \in \mathsf{Th}([\![x]\!]_{\approx_j})$, since $[\![x]\!]_{\approx_j} \subseteq [\![x]\!]_{\approx}$. We know that $[\![x]\!]_{\approx_j} \neq \emptyset$, so pick $c \in [\![x]\!]_{\approx_j}$. Since $\varphi \in \mathsf{Th}([\![x]\!]_{\approx_j})$, we have $c \models \varphi$, and since $c \approx_j x$, we have $x \models \varphi$. Hence $\varphi \in \mathsf{Th}(x)$, as required. □

Now we can show:

Proposition 4 *In a representation system, $\delta_x = \Box_{\mathcal{K}}[\![x]\!]$ for every x.*

Proof. We first show that $\mathsf{Mod}(\mathsf{Th}(x)) = \uparrow x$. Let $y \succeq x$. By monotonicity we have $y \in \mathsf{Mod}(\mathsf{Th}(x))$. Conversely let $y \not\succeq x$; then $\delta_x \in \mathsf{Th}(x) - \mathsf{Th}(y)$ and hence $y \notin \mathsf{Mod}(\mathsf{Th}(x))$. Thus indeed $\mathsf{Mod}(\mathsf{Th}(x)) = \uparrow x$. Since $\uparrow x = \mathsf{Mod}(\delta_x)$ and $\mathsf{Mod}(\mathsf{Th}(x)) = \mathsf{Mod}(\mathsf{Th}([\![x]\!]))$ by Proposition 3, we have $\mathsf{Mod}(\delta_x) = \mathsf{Mod}(\mathsf{Th}([\![x]\!]))$, i.e., $\delta_x = \Box_{\mathcal{K}}[\![x]\!]$. □

Corollary 1 *In a representation system:*

- $\mathsf{Mod}(\delta_x) = \mathsf{Mod}(\mathsf{Th}(x))$, *and*
- $\Box_{\mathcal{O}}[\![x]\!] \models \Box_{\mathcal{K}}[\![x]\!]$

for every x.

We finally give a condition equivalent to the existence of formulae δ_x. A set Φ of formulae is *finitely axiomatizable* if there is a finite set Φ_0 such that $\mathsf{Mod}(\Phi) = \mathsf{Mod}(\Phi_0)$.

Proposition 5 *In a pre-representation system $\mathbb{RS}^{\circ}$ whose formulae are closed under conjunction, the following are equivalent:*

1. *Each $\mathsf{Th}(x)$ is finitely axiomatizable, and $\mathsf{Th}(x) \subseteq \mathsf{Th}(y)$ implies $x \preceq y$ for all x, y;*
2. *for each x, there is a formula δ_x so that $\mathsf{Mod}(\delta_x) = \uparrow x$.*

Proof sketch. If Φ is a finite axiomatization of $\mathsf{Th}(x)$, it is easy to see that δ_x can be taken to be $\bigwedge\{\varphi \mid \varphi \in \Phi\}$. Conversely, one shows that δ_x axiomatizes $\mathsf{Th}(x)$. □

$\Box_{\mathcal{K}}$ as a greatest lower bound

We now show that $\Box_{\mathcal{K}}X$ can be viewed as a greatest lower bound as well. Note that we have a well-known preorder on sets of formulae, namely implication: $\Phi \vdash \Psi$ iff $\mathsf{Mod}(\Phi) \subseteq \mathsf{Mod}(\Psi)$. Thus, for any set of formulae Φ, we can look at its greatest lower bound in this preorder, i.e., a formula φ so that $\varphi \vdash \Phi$, and whenever $\psi \vdash \Phi$, we have $\psi \vdash \varphi$. If such a formula exists, it is denoted by $\bigwedge \Phi$. Note that as with objects, $\vdash$ is a preorder, so technically $\bigwedge \Phi$ is a set of formulae, all of which, however, are equivalent, so we shall write $\varphi = \bigwedge \Phi$ when φ is one of such formulae.

Theorem 1 *In a representation system, $\Box_{\mathcal{K}}X = \bigwedge \mathsf{Th}(X)$ for every set X of objects.*

Proof. Assume that $\alpha = \bigwedge \mathsf{Th}(X)$ exists. We have $\mathsf{Mod}(\alpha) \subseteq \mathsf{Mod}(\varphi)$ for each $\varphi \in \mathsf{Th}(X)$ and thus $\mathsf{Mod}(\alpha) \subseteq \mathsf{Mod}(\mathsf{Th}(X))$. Suppose, for the sake of contradiction, that $\mathsf{Mod}(\alpha) \subsetneq \mathsf{Mod}(\mathsf{Th}(X))$, and take $y \in \mathsf{Mod}(\mathsf{Th}(X)) - \mathsf{Mod}(\alpha)$. Since $y \models \varphi$ for each $\varphi \in \mathsf{Th}(X)$, we have that the same is true for every $z \succeq y$, and hence $\mathsf{Mod}(\delta_y) \subseteq \mathsf{Mod}(\varphi)$ for each $\varphi \in \mathsf{Th}(X)$. Thus $\delta_y \vdash \mathsf{Th}(X)$, and by the definition of α as the greatest lower bound, we have $\delta_y \vdash \alpha$, and thus $\mathsf{Mod}(\delta_y) \subseteq \mathsf{Mod}(\alpha)$.

Now assume $\mathsf{Mod}(\alpha) \subseteq \mathsf{Mod}(\delta_y)$. Then $\mathsf{Mod}(\alpha) = \mathsf{Mod}(\delta_y)$ and $y \in \mathsf{Mod}(\alpha)$, a contradiction. Hence, $\mathsf{Mod}(\alpha) \not\subseteq \mathsf{Mod}(\delta_y)$. But then $\mathsf{Mod}(\delta_y) \not\subseteq \mathsf{Mod}(\alpha)$ since $y \not\models \alpha$, and at the same time $\mathsf{Mod}(\delta_y) \subseteq \mathsf{Mod}(\mathsf{Th}(X))$ since $y \in \mathsf{Mod}(\mathsf{Th}(X))$. Thus sets $\mathsf{Mod}(\delta_y)$ and $\mathsf{Mod}(\alpha)$ are incomparable subsets of $\mathsf{Mod}(\mathsf{Th}(X))$: in particular, $\alpha \vdash \mathsf{Th}(X)$, $\delta_y \vdash \mathsf{Th}(X)$, and yet neither $\alpha \vdash \delta_y$ nor $\delta_y \vdash \alpha$ holds, contradicting the assumption that $\alpha = \bigwedge \mathsf{Th}(X)$. This shows that $\mathsf{Mod}(\mathsf{Th}(X)) - \mathsf{Mod}(\alpha) = \emptyset$ and thus $\mathsf{Mod}(\alpha) = \mathsf{Mod}(\mathsf{Th}(X))$.

Conversely, suppose we have α so that $\mathsf{Mod}(\alpha) = \mathsf{Mod}(\mathsf{Th}(X)) = \bigcap_{\varphi \in \mathsf{Th}(X)} \mathsf{Mod}(\varphi)$. Then $\mathsf{Mod}(\alpha) \subseteq \mathsf{Mod}(\varphi)$ for each $\varphi \in \mathsf{Th}(X)$ and thus $\alpha \vdash \varphi$ for each such φ. If there is any other formula ψ such that $\psi \vdash \varphi$ for each $\varphi \in \mathsf{Th}(X)$, then $\mathsf{Mod}(\psi) \subseteq \bigcap_{\varphi \in \mathsf{Th}(X)} \mathsf{Mod}(\varphi) = \mathsf{Mod}(\alpha)$ and thus $\psi \vdash \alpha$, proving that $\alpha = \bigwedge \mathsf{Th}(X)$. □

5 Defining certain answers to queries

Now we move to answering queries. A query is a mapping Q that takes an object and returns another object. For instance, relational queries take relational databases and return relational databases (most commonly, single relations: queries in FO, or in commercial languages such as SQL, are such).

Thus, for two database domains $\mathbb{D} = \langle \mathcal{D}, \mathcal{C}, [\![\,]\!], \mathrm{Iso} \rangle$ and $\mathbb{D}' = \langle \mathcal{D}', \mathcal{C}', [\![\,]\!]', \mathrm{Iso}' \rangle$, a *query* $Q : \mathbb{D} \to \mathbb{D}'$ is a mapping associating with an object $x \in \mathcal{D}$ its answer, $Q(x) \in \mathcal{D}'$.

We require that if $x \preceq y$ then $Q(x) \preceq' Q(y)$, where $\preceq'$ is the ordering associated with the semantics $[\![\,]\!]'$. Indeed, the semantics of answers must be such that if we know more about the input, then we know more about the output. In particular, using blindly the same semantics for both databases and query results (as is often actually done) does not necessarily make sense.

Certain answers to Q on an object x represent certain information in the set $Q([\![x]\!]) = \{Q(c) \mid c \in [\![x]\!]\}$. We have seen that there are two ways to define it: as object, and as knowledge. For the letter, we need to have a representation system $\mathbb{RS} = \langle \mathbb{D}', \mathbb{F}, \models \rangle$ over the target domain $\mathbb{D}'$. If we have it, we can either extract the most general object representing $Q([\![x]\!])$, or the most general knowledge representing $\mathsf{Th}(Q([\![x]\!]))$. That is, we have two certain answers notions, as objects and as knowledge:

- As objects: $\mathsf{cert}_{\mathcal{O}}(Q, x) = \Box_{\mathcal{O}} Q([\![x]\!])$;
- As knowledge: $\mathsf{cert}_{\mathcal{K}}(Q, x) = \Box_{\mathcal{K}} Q([\![x]\!])$.

Comparing with relational theory Let us now review the standard approach to query answering in relational databases. Ideally, one tries to find a query answer A so that $[\![A]\!]' = Q([\![D]\!])$. This is often impossible, in fact even for very simple queries (Imielinski and Lipski 1984). So the next attempt is to find a formula $\varphi_{Q,D}$ in some logical formalism so that

$$\mathsf{Mod}(\varphi_{Q,D}) = Q([\![D]\!]) \tag{1}$$

When this happens, one refers to such a logical formalism as a *strong representation system* (see (Abiteboul, Hull, and Vianu 1995; Imielinski and Lipski 1984)), which explains why we used the name 'representation system'.

The problem is that the structure of $Q(\llbracket D \rrbracket)$ may be too "irregular" to be described by a nice formalism. For instance, it is known that under CWA, for relational calculus queries formulae $\varphi_{Q,D}$ can be of the following form: $\exists \bar{u}\ (\alpha(\bar{u}) \wedge \bigwedge_{R \in \sigma} \forall \bar{x}\ (R(\bar{x}) \leftrightarrow \bigvee_i (\bar{x} = \bar{v}_i \wedge \beta_i(\bar{x}, \bar{u}))))$, where α and β_i are boolean combinations of equalities, $\bar{v}_i$s combine variables from $\bar{u}$ and constants, and $\bar{u}$ ranges over the underlying domain of constants rather than the active domain, see (Imielinski and Lipski 1984). Syntactically, this is quite heavy, and it works only under the CWA.

If the set $Q(\llbracket D \rrbracket)$ does not happen to be of the form $\mathsf{Mod}(\varphi)$ for some nice formula φ, the approach adopted in the database literature is to consider the object $\bigcap Q(\llbracket D \rrbracket)$ as the answer. This is completely ad hoc, however: as we have seen in the Introduction (and as we shall see in the next section), in general it does not have much in common with certain information contained in $Q(\llbracket D \rrbracket)$.

It seems much better to ask then, in place of (1), for an answer $\varphi_{Q,D}$ that is *equivalent to the theory* of $Q(\llbracket D \rrbracket)$, rather than defining $Q(\llbracket D \rrbracket)$ precisely. That is, we replace (1) with

$$\mathsf{Mod}(\varphi_{Q,D}) \ = \ \mathsf{Mod}(\mathsf{Th}(Q(\llbracket D \rrbracket))) \tag{2}$$

which is, of course, our definition of certain answers expressed as knowledge.

Note that (1) implies (2): this is an immediate consequence of the fact that $\mathsf{Mod}(\cdot)$ and $\mathsf{Th}(\cdot)$ define a Galois connection. Thus, the notion of certain answers as knowledge in a representation system is a weakening of the notion of the strong representation system, but much less ad hoc that replacing $Q(\llbracket D \rrbracket)$ with $\bigcap Q(\llbracket D \rrbracket)$.

Example: when representation system makes a difference We can easily construct examples of relational queries Q and representation systems so that (1) fails while (2) is easily achieved. Suppose we have a schema with two relations R, S (for simplicity, just sets), and the query $R - S$ (in FO, $R(x) \wedge \neg S(x)$), and assume closed-world semantics. Consider D in which $R = \{1, 2\}$ and $S = \{\bot\}$. Then $Q(\llbracket D \rrbracket_{\text{CWA}}) = \{\{1\}, \{2\}, \{1, 2\}\}$. Suppose the representation system is $\langle \mathbb{D}(\sigma), \exists\mathsf{Pos}, \models \rangle$. Then there is no α with $\mathsf{Mod}(\alpha) = Q(\llbracket D \rrbracket_{\text{CWA}})$ but there is one such that $\mathsf{Mod}(\alpha) = \mathsf{Mod}(\mathsf{Th}(Q(\llbracket D \rrbracket_{\text{CWA}})))$; in fact, the obvious answer $\alpha = A(1) \vee A(2)$ does the job.

6 Computing certain answers

We now look at computing certain answers. We show that with the proper semantics of query answering, where more informative inputs lead to more informative answers, finding certain answers is reduced to query evaluation:

(a) for objects, $\mathsf{cert}_{\mathcal{O}}(Q, x) = Q(x)$;

(b) for knowledge, $\mathsf{cert}_{\mathcal{K}}(Q, x) = \delta_{Q(x)}$.

Crucially, (a) is a corollary of (b): without a representation system for query answers, (a) may not be true. We explain how these general results apply to relational OWA, WCWA, and CWA semantics. We also revisit the intersection-based definition of certain answers and show that it only makes sense with restricted representation systems under OWA, and is just plain wrong under CWA. Finally, we show that the knowledge approach to certain answers gives us extra flexibility compared to the object approach.

Recall that queries are required to be monotone: $x \preceq y$ implies $Q(x) \preceq' Q(y)$. We need an additional condition of *genericity*, standard in the database context (see, e.g., (Abiteboul, Hull, and Vianu 1995)). In our abstract framework it is expressed as follows: for every j, there is k so that $x \approx_k y$ implies $Q(x) \approx'_j Q(y)$. Essentially, this condition says that queries applied to isomorphic objects return isomorphic objects. For instance, for FO queries that do not refer to constants, it is usually formulated as $D \approx D' \Rightarrow Q(D) \approx Q(D')$. We use a slightly more refined version that, in the case of logically expressed queries, accounts for constants by using multiple equivalence relations. All queries expressed in FO and other logics over the vocabulary of relation symbols and constants are generic in the standard database domains for relational databases.

Theorem 2 *Let $Q : \mathbb{D} \to \mathbb{D}'$ be a monotone generic query, and let $\mathbb{RS} = \langle \mathbb{D}', \mathbb{F}, \models \rangle$ be a representation system. Then, for every object x,*

- $\mathsf{cert}_{\mathcal{O}}(Q, x) = Q(x)$, *and*
- $\mathsf{cert}_{\mathcal{K}}(Q, x) = \delta_{Q(x)}$.

Proof. We start by showing $\mathsf{cert}_{\mathcal{K}}(Q, x) = \delta_{Q(x)}$. By the definition of $\mathsf{cert}_{\mathcal{K}}(Q, x)$ as $\Box_{\mathcal{K}} Q(\llbracket x \rrbracket)$, it suffices to show that

$$\mathsf{Mod}(\delta_{Q(x)}) \ = \ \mathsf{Mod}(\mathsf{Th}(Q(\llbracket x \rrbracket))) \tag{3}$$

for every x. Since we know that $\mathsf{Mod}(\delta_z) = \mathsf{Mod}(\mathsf{Th}(z))$, for every z, by Corollary 1, we just need to show

$$\mathsf{Th}(Q(\llbracket x \rrbracket)) \ = \ \mathsf{Th}(Q(x)) \tag{4}$$

to conclude (3). Suppose $\varphi \in \mathsf{Th}(Q(x))$. Since $c \succeq x$ for every $c \in \llbracket x \rrbracket$, then $Q(x) \preceq' Q(c)$ and $Q(c)$ satisfies φ as well by the properties of representation systems, proving that $\varphi \in \mathsf{Th}(Q(\llbracket x \rrbracket))$.

Conversely, if $\varphi \in \mathsf{Th}(Q(\llbracket x \rrbracket))$, consider j such that $z \approx'_j z'$ implies that z, z' agree on φ for every z, z' (which exists by the definition of representation systems). By genericity, we have k so that $y \approx_k y'$ implies $Q(y) \approx'_j Q(y')$ for every y, y'. We know that $\llbracket x \rrbracket_{\approx_k}$ is nonempty; this pick an element c in this set. We have $c \in \llbracket x \rrbracket$ and $c \approx_k x$. Hence (a) $Q(c) \in Q(\llbracket x \rrbracket)$ and (b) $Q(c) \approx'_j Q(x)$. Then (a) implies that $Q(c) \models \varphi$, and (b) then implies that $Q(x) \models \varphi$, thus showing that $\varphi \in \mathsf{Th}(Q(x))$ and proving (4) and (3).

Now we prove the result about certain answers at the object level. Since Q is monotone, we have $Q(x) \preceq' z$ for each $z \in Q(\llbracket x \rrbracket)$ (we denote this by $Q(x) \preceq' Q(\llbracket x \rrbracket)$. Suppose we have $y \preceq' Q(\llbracket x \rrbracket)$. Then $\mathsf{Th}(y) \subseteq \mathsf{Th}(Q(\llbracket x \rrbracket))$ and by (4), $\mathsf{Th}(y) \subseteq \mathsf{Th}(Q(x))$. Since $\delta_y \in \mathsf{Th}(y)$, we have $\delta_y \in \mathsf{Th}(Q(x))$, and thus $Q(x) \models \delta_y$ and $y \preceq' Q(x)$. Thus, $Q(x)$ is the greatest lower bound of $Q(\llbracket x \rrbracket)$, i.e., $Q(x) = \mathsf{cert}_{\mathcal{O}}(Q, x)$. □

Remark It is easy to construct examples of monotone queries for which, in the absence of a representation system over query answers, $\mathsf{cert}_{\mathcal{O}}(Q,x) \neq Q(x)$. Thus, it is essential to go via certain answer as knowledge to get its object representation as well.

The following easy observation shows that classes of queries for which certain answers can be computed by directly evaluating the query are closed under composition, giving us a way of producing more complex queries that behave well with respect to certain answer computation.

Proposition 6 *If $Q : \mathbb{D} \to \mathbb{D}'$ and $Q' : \mathbb{D}' \to \mathbb{D}''$ are monotone and generic, then so is $Q' \circ Q : \mathbb{D} \to \mathbb{D}''$.*

Genericity applies to most of the logical formalisms used for querying databases, but not all: formalisms capable of referring to infinitely many constants would not fall into that category. Examples of such formalisms occur, for instance, in data exchange, where one deals with logics that use the predicate $const(\cdot)$ to distinguish constants from nulls, see (Arenas, Barceló, and Reutter 2009; Fagin 2007).

There is another condition we can use in place of genericity, namely a substitution property with respect to two representation systems $\mathbb{RS} = \langle \mathbb{D}, \mathbb{F}, \models \rangle$ and $\mathbb{RS}' = \langle \mathbb{D}', \mathbb{F}', \models' \rangle$. We say that a query Q has the *substitution property* with respect to $\mathbb{RS}$ and $\mathbb{RS}'$ if for each $\varphi \in \mathbb{F}'$, there exists $\varphi_Q \in \mathbb{F}$ so that $x \models \varphi_Q$ iff $Q(x) \models \varphi$. Intuitively, we can think of both φ and Q given as logical formulae, and allow Q to be substituted for predicate symbols used in φ.

Theorem 3 *Let $Q : \mathbb{D} \to \mathbb{D}'$ be a monotone query, and assume we have representation systems $\mathbb{RS}$ and $\mathbb{RS}'$ so that Q has the substitution property with respect to them. Then, for every object x,*

- $\mathsf{cert}_{\mathcal{O}}(Q,x) = Q(x)$*, and*
- $\mathsf{cert}_{\mathcal{K}}(Q,x) = \delta_{Q(x)}$.

Proof. We just need to prove (4) and then the proof of Theorem 2 will apply. The $\mathsf{Th}(Q(x)) \subseteq \mathsf{Th}(Q([\![x]\!]))$ inclusion is proved as before. For the converse, let φ be in $\mathsf{Th}(Q([\![x]\!]))$; take φ_Q that exists by the substitution property. Since $Q(c) \models \varphi$ for each $c \in [\![x]\!]$, we have $c \models \varphi_Q$, and thus $\varphi_Q \in \mathsf{Th}([\![x]\!])$. But we know by Proposition 3 that $\mathsf{Th}([\![x]\!]) = \mathsf{Th}(x)$ and hence $\varphi_Q \in \mathsf{Th}(x)$. But now the substitution property and $x \models \varphi_Q$ imply $Q(x) \models \varphi$, showing $\varphi \in \mathsf{Th}(Q(x))$ and proving the reverse inclusion. □

Certain answers for relational queries

We now look at examples of concrete domains and representation systems for relational databases. The domains are always $\mathbb{D}_*(\sigma)$, containing databases of vocabulary σ, with $*$ being the semantics. If a query Q returns sets of m-ary tuples, the domain $\mathbb{D}'$ will have as objects databases of vocabulary σ_m that contains a single m-ary relation $A(\cdot)$. But what will the semantics be? A natural thing to do is to assume the same semantics for query answers and input data. But will this guarantee conditions that let us apply results of the previous section?

It turns out that for classes of queries given by the same first-order formulae we use in representation systems $\mathbb{RS}_*$, we have all the required conditions.

Proposition 7 *Consider an m-ary relational query $Q : \mathbb{D}_*(\sigma) \to \mathbb{D}_*(\sigma_m)$. Then Q is monotone*

- *for $* = $*OWA*: when Q is an $\exists\mathsf{Pos}$ query;*
- *for $* = $*WCWA*: when Q is an Pos query;*
- *for $* = $*CWA*: when Q is an $\mathsf{Pos}^{\forall\mathsf{G}}$ query.*

Moreover, for $*$ being OWA *or* WCWA*, each such Q has the substitution property with respect to $\mathbb{RS}_*(\sigma)$ and $\mathbb{RS}_*(\sigma_m)$.*

Proof. It is very similar to the proof of Proposition 1, as it follows from the preservation properties of the classes of formulae we use. The second property is by a simple inspection of the syntactic structure of the formulae. □

Corollary 2 *Let Q be an m-ary relational query. Under the* OWA*,* WCWA*, and* CWA *semantics, if Q comes from $\exists\mathsf{Pos}$, Pos, or $\mathsf{Pos}^{\forall\mathsf{G}}$ respectively, then $\mathsf{cert}_{\mathcal{O}}(Q,D) = Q(D)$ and $\mathsf{cert}_{\mathcal{K}}(Q,D) = \delta_{Q(D)}$ for every database D (with δ coming from $\mathbb{RS}_*(\sigma_m)$, when $*$ is* OWA*,* WCWA*, or* CWA*).*

Returning to the first example from the introduction, assume that our query just returns a relation R itself, and $R = \{(1,2),(3,\bot)\}$. Then, at the object level, $\mathsf{cert}_{\mathcal{O}}(Q,R) = R$ for all three semantics. Certain answers as knowledge, or $\mathsf{cert}_{\mathcal{K}}(Q,R)$ are given:

- under OWA, by the $\exists\mathsf{Pos}$ formula $\exists x \big(R(1,2) \wedge R(3,x)\big)$;
- under WCWA, by the Pos formula

$$\exists x \left(R(1,2) \wedge R(3,x) \ \wedge \ \forall y \ \left(\begin{array}{l} y = 1 \vee y = 2 \\ \vee \ \ y = 3 \vee y = x \end{array} \right) \right);$$

- and under CWA, by the $\mathsf{Pos}^{\forall\mathsf{G}}$ formula

$$\exists x \left(\begin{array}{l} R(1,2) \wedge R(3,x) \\ \wedge \ \ \forall y,z \ \left(R(y,z) \rightarrow \ \begin{array}{l} (y = 1 \wedge z = 2) \\ \vee \ \ (y = 3 \wedge z = x) \end{array} \right) \end{array} \right)$$

No intersection for certain answers

Now we return to the "classical" of defining certain answers which involves throwing away tuples with nulls and computing intersection, and give more evidence to the fact that this practice does not really work, outside a fairly restrictive representation system under the OWA. As already mentioned in the preliminaries section, for the classes of queries listed in Proposition 7, the well-known equivalences say that $\mathsf{cert}_{\cap}(Q,D)$ is the same $Q^{\mathsf{C}}(D)$, which is $Q(D)$ from which tuples of nulls have been eliminated. We can also view this as a composition of two mappings: Q^{C} is $\pi^{\mathsf{C}} \circ Q$, where π^{C} simply keeps all tuples that only use constants.

The condition that we need on π^{C}, and on Q^{C}, is monotonicity. However, these are not guaranteed in general.

Proposition 8 *The mapping π^{C} is monotone with respect to the ordering $\preceq_{\text{OWA}}$ but it is* not *monotone with respect to $\preceq_{\text{CWA}}$ and $\preceq_{\text{WCWA}}$.*

In fact there are $\exists\mathsf{Pos}$ queries Q such that Q^{C} is not monotone with respect to $\preceq_{\text{CWA}}$ and $\preceq_{\text{WCWA}}$.

Proof. Since there is the identity homomorphism from any substructure to the structure, we have monotonicity under $\preceq_{\text{OWA}}$. For closed world semantics, simply take $R = \{(1,2),(2,\bot)\}$. Then $\pi^{\mathsf{C}}(R) = R^{\mathsf{C}} = \{(1,2)\}$ and $R \not\preceq_{\text{CWA}} R^{\mathsf{C}}$ nor $R \not\preceq_{\text{WCWA}} R^{\mathsf{C}}$. Thus the $\exists\mathsf{Pos}$ query can be taken to be the query returning relation R. □

The consequences of this under WCWA and CWA are twofold. We may easily have situations such that:

- $Q^{\mathsf{C}}(D) \not\preceq \mathsf{cert}_{\mathcal{O}}(Q, D)$, and
- $\mathsf{cert}_{\mathcal{K}}(Q, D)$ does not logically imply $\delta_{Q^{\mathsf{C}}(D)}$.

In other words, such constant 'certain answer' need not represent certain information in $Q([\![D]\!])$, and thus may contain some false positives. This is also true for certain answers defined as intersection because in the example used in Proposition 8, we have $Q^{\mathsf{C}}(D) = \bigcap\{Q(D') \mid D' \in [\![D]\!]\}$ under both WCWA and CWA semantics.

Thus, the use of the intersection operator is indeed very problematic under WCWA and CWA. In fact this should not come as a surprise at all, even if we deal with query answering at the object level. The $\bigcap$ operator is a greatest lower bound for the $\subseteq$ ordering, and $\subseteq$ results in a more informative instance only under OWA, but not under closed-world assumptions! Thus, choosing $\bigcap$ to define certain answers for all semantics was a completely ad hoc choice.

OWA: when intersection still makes sense We now look at OWA, the only case where the mapping π^{C} is monotone, and where intersection corresponds to the greatest lower bound in an ordering compatible with the semantics.

Given a database domain $\mathbb{D} = \langle \mathcal{D}, \mathcal{C}, [\![\,]\!], \text{Iso}\rangle$, let $\mathbb{D}^{\mathsf{C}}$ be its restriction to complete objects, i.e., $\mathbb{D}^{\mathsf{C}} = \langle \mathcal{C}, \mathcal{C}, [\![\,]\!], \text{Iso}\rangle$. It is easy to check that this is still a database domain.

Let $\mathsf{PBCatom}$ stand for the set of Positive Boolean Combinations of atoms $R(\bar{c})$, with $\bar{c}$ consisting only of elements of Const. Recall that for OWA, we used the representation system $\mathbb{RS}_{\text{OWA}}(\sigma) = \langle \mathbb{D}(\sigma), \exists\mathsf{Pos}, \models\rangle$. Define a new representation system $\mathbb{RS}^{\mathsf{C}}_{\text{OWA}}(\sigma) = \langle \mathbb{D}^{\mathsf{C}}(\sigma), \mathsf{PBCatom}, \models\rangle$.

Proposition 9 *$\mathbb{RS}^{\mathsf{C}}_{\text{OWA}}(\sigma)$ is a representation system, and if Q is an $\exists\mathsf{Pos}$ query, then Q^{C} is monotone and generic with respect to it.*

Proof. Immediate from the fact that $\text{PosDiag}(D)$ for a complete D is a conjunction of atoms of the form $R(\bar{c})$, and that $\exists\mathsf{Pos}$ queries are preserved under homomorphisms. □

Thus, for every database D, and every $\exists\mathsf{Pos}$ query Q, we have $\mathsf{cert}_{\mathcal{O}}(Q^{\mathsf{C}}, D) = Q^{\mathsf{C}}(D)$ and $\mathsf{cert}_{\mathcal{K}}(Q^{\mathsf{C}}, D) = \delta_{Q^{\mathsf{C}}(D)}$ with respect to $\mathbb{RS}^{\mathsf{C}}_{\text{OWA}}(\sigma)$. Hence, under OWA, the traditional way of finding certain answers makes sense if we use a representation system that only 'cares' about complete databases.

Certain knowledge is more flexible

Note that for sets X of the form $Q([\![x]\!])$ for monotone generic queries Q, we have $\Box_{\mathcal{O}}X \models \Box_{\mathcal{K}}X$, as is implied by Theorem 2. In general, however, this need not be the case. The reason is that taking the greatest lower bound of X may lose more information than taking the greatest lower bound of $\mathsf{Th}(X)$, which indicates that working with certain answers as knowledge may be preferable, as they convey more information.

Proposition 10 *There is a representation system and a set of objects X such that $\Box_{\mathcal{O}}X \not\models \Box_{\mathcal{K}}X$.*

Proof. Consider a domain $\mathcal{D}$ with the ordering $x_1 \succeq x_2 \succeq \ldots \succeq x_n \succeq \ldots \succeq x_*$. We assume that all $\approx_j$s are the same, and for each x_i, there is $c_i \succeq x_i$ with $x_i \approx c_i$. The formulas are φ_i for $i \geq 0$. Let $\mathsf{Th}(x_i) = \{\varphi_0\} \cup \{\varphi_j \mid j \geq i\}$ (with $\mathsf{Th}(c_i) = \mathsf{Th}(x_i)$) and $\mathsf{Th}(x_*) = \emptyset$. Let $X = \{x_i \mid i \neq 0\}$. Then $\bigwedge X = x_*$ and $\mathsf{Th}(X) = \{\varphi_0\}$; hence $\Box_{\mathcal{O}}X = x_*$ and $\Box_{\mathcal{K}}X = \varphi_0$ and thus $\Box_{\mathcal{O}}X \not\models \Box_{\mathcal{K}}X$. □

We finish with examples showing that certain answer may not exist as an object while it still exists as knowledge over an appropriate representation system. Take a query $Q = R - S$, and a database D with $R^D = \{1,2\}$ and $S^D = \{3,\bot\}$.

Under CWA, we have $Q([\![D]\!]_{\text{CWA}}) = \{\ \{1,2\}, \{1\}, \{2\}\ \}$. If we take the greatest lower bound of these, then for $\preceq_{\text{CWA}}$ (and $\preceq_{\text{WCWA}}$) it simply does not exist. For the subset ordering, or the $\preceq_{\text{OWA}}$ ordering, the greatest lower bound is $\emptyset$, which is, incidentally, what a commercial DBMS would return if one runs this query in SQL (e.g., as *select r.a from r where r.a not in (select * from s)*). It seems bizarre, to say the least, to use $\preceq_{\text{OWA}}$ while dealing with the CWA semantics. However, the alternative is to return no answers at all!

This is of course yet another illustration that returning certain answers as objects is not always possible, and one needs to return certain answers as knowledge instead. What sort of knowledge will depend, of course, on the representation system. If we use the set of $\exists\mathsf{Pos}$ formulae, then the best way of describing certain answers will be $A(1) \vee A(2)$.

Next, consider the same query under OWA. Then $Q([\![D]\!]_{\text{OWA}}) = \{A \mid 3 \notin A\}$. Again, this is not representable by a single object: taking the greatest lower of $Q([\![D]\!]_{\text{OWA}})$ results again in $\emptyset$, missing some valuable information. If we use representation systems that allow negations of atomic formulae, then $\neg A(3)$ will represent the certain answer properly.

7 Conclusions

We have argued that the standard definition of certain answers in the database literature has a number of deficiencies, and proposed a new approach to handling queries over incomplete databases. Its key features are as follows.

- Certain answers can be defined at two different levels: as (database) objects, or as knowledge we possess about query answers with certainty.
- The proposed framework, that applies to multiple data models, defines both types of certain answers as greatest lower bounds in orderings that capture the level of informativeness. It also leads to a proper definition of representation systems for query answers.
- If the semantics of query answering is chosen properly, then the process of finding certain answers is reduced to

query evaluation, at both object and knowledge level. Furthermore, the knowledge level is crucial for obtaining results at the object level.

- This tells us that with the right choice of semantics, no new tools are needed for computing query answers and one can rely on the standard database query evaluation engine. It also tells us that using the traditional way of finding certain answers only makes sense under OWA, and with restricted representation systems.
- In general, certain answers as knowledge give us more information than certain answers as objects.

Future work. There are several directions to consider. Commercial languages such as SQL use multi-valued logic for reasoning. Evaluation algorithms of similar nature have been explored in the knowledgebase literature (Levesque 1998), sometimes even using database evaluation techniques (Liu and Levesque 2003). While not directly applicable to relational databases, the connection is worth studying, especially for representing certain answers as knowledge.

Another direction has to do with the detailed study of the efficiency of computing certain answers, and, if intractability is encountered, finding either tractable restrictions, or approximation schemes, perhaps in the spirit of (Reiter 1986).

Yet another direction is to apply the framework to non-relational models, particularly semi-structured and XML, for which incompleteness has been studied extensively (Abiteboul, Segoufin, and Vianu 2006; Barceló et al. 2010; Calvanese, De Giacomo, and Lenzerini 1998; David, Libkin, and Murlak 2010), and beyond, to graph data, where only preliminary results have been established so far (Barceló, Libkin, and Reutter 2014).

Acknowledgments I am grateful to Pablo Barceló and anonymous referees for their comments. Work partly supported by EPSRC grant J015377.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Abiteboul, S.; Kanellakis, P.; and Grahne, G. 1991. On the representation and querying of sets of possible worlds. *Theoretical Computer Science* 78(1):158–187.

Abiteboul, S.; Segoufin, L.; and Vianu, V. 2006. Representing and querying XML with incomplete information. *ACM Transactions on Database Systems* 31(1):208–254.

Arenas, M.; Barceló, P.; and Reutter, J. 2009. Query languages for data exchange: beyond unions of conjunctive queries. In *International Conference on Database Theory (ICDT)*, 73–83.

Arenas, M.; Barceló, P.; Libkin, L.; and Murlak, F. 2014. *Foundations of Data Exchange*. Cambridge University Press.

Barceló, P.; Libkin, L.; Poggi, A.; and Sirangelo, C. 2010. XML with incomplete information. *Journal of the ACM* 58(1).

Barceló, P.; Libkin, L.; and Reutter, J. 2014. Querying regular graph patterns. *Journal of the ACM* 61(1).

Barwise, J., and Feferman, S., eds. 1985. *Model-Theoretic Logics*. Springer Verlag.

Bertossi, L. 2011. *Database Repairing and Consistent Query Answering*. Morgan&Claypool Publishers.

Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 1998. Semi-structured data with constraints and incomplete information. In *Description Logics*.

Chang, C., and Keisler, H. 1990. *Model Theory*. North Holland.

Compton, K. 1983. Some useful preservation theorems. *Journal of Symbolic Logic* 48(2):427–440.

David, C.; Libkin, L.; and Murlak, F. 2010. Certain answers for XML queries. In *ACM Symposium on Principles of Database Systems (PODS)*, 191–202.

Fagin, R. 2007. Inverting schema mappings. *ACM Transactions on Database Systems* 32(4).

Fan, W., and Geerts, F. 2012. *Foundations of Data Quality Management*. Morgan&Claypool Publishers.

Gheerbrant, A.; Libkin, L.; and Sirangelo, C. 2013. When is naïve evaluation possible? In *ACM Symposium on Principles of Database Systems (PODS)*, 201–212.

Grant, J. 1977. Null values in a relational data base. *Inf. Process. Lett.* 6(5):156–157.

Gunter, C. 1992. *Semantics of Programming Languages: Structures and Techniques*. MIT Press.

Imielinski, T., and Lipski, W. 1984. Incomplete information in relational databases. *Journal of the ACM* 31(4):761–791.

Lenzerini, M. 1991. Type data bases with incomplete information. *Inf. Sci.* 53(1-2):61–87.

Lenzerini, M. 2002. Data integration: a theoretical perspective. In *ACM Symposium on Principles of Database Systems (PODS)*, 233–246.

Levesque, H. J. 1998. A completeness result for reasoning with incomplete first-order knowledge bases. In *KR*, 14–23.

Libkin, L. 2011. Incomplete information and certain answers in general data models. In *ACM Symposium on Principles of Database Systems (PODS)*, 59–70.

Lipski, W. 1979. On semantic issues connected with incomplete information databases. *ACM Transactions on Database Systems* 4(3):262–296.

Liu, Y., and Levesque, H. J. 2003. A tractability result for reasoning with incomplete first-order knowledge bases. In *IJCAI*, 83–88.

Reiter, R. 1980. Equality and domain closure in first-order databases. *Journal of the ACM* 27(2):235–249.

Reiter, R. 1982. Towards a logical reconstruction of relational database theory. In *On Conceptual Modelling*, 191–233.

Reiter, R. 1986. A sound and sometimes complete query evaluation algorithm for relational databases with null values. *Journal of the ACM* 33(2):349–370.

Rossman, B. 2008. Homomorphism preservation theorems. *Journal of the ACM* 55(3).

Suciu, D.; Olteanu, D.; Re, C.; and Koch, C. 2011. *Probabilistic Databases*. Morgan&Claypool Publishers.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

A Formalization of Programs in First-Order Logic with a Discrete Linear Order*

Fangzhen Lin (flin@cs.ust.hk)
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

We consider the problem of representing and reasoning about computer programs, and propose a translator from a core procedural iterative programming language to first-order logic with quantification over the domain of natural numbers that includes the usual successor function and the "less than" linear order, essentially a first-order logic with a discrete linear order. Unlike Hoare's logic, our approach does not rely on loop invariants. Unlike typical temporal logic specification of a program, our translation does not require a transition system model of the program, and is compositional on the structures of the program. Some non-trivial examples are given to show the effectiveness of our translation for proving properties of programs.

Introduction

In computer science, how to represent and reason about computer programs effectively has been a major concern since the beginning. For imperative, non-concurrent programs that we are considering here, notable approaches include Dijkstra's calculus of weakest preconditions (Dijkstra 1976; Dijkstra and Scholten 1990), Hoare's logic (Hoare 1969), dynamic logic (Harel 1979), and separation logic (Reynolds 2002). For the most part, these logics provide rules for proving assertions about programs. In particular, for proving assertions about iterative loops, these logics rely on what have been known as Hoare's loop invariants. In this paper, we propose a way to translate a program to a first-order theory with quantification over natural numbers. The properties that we need about natural numbers are that they have a smallest element (zero), are linearly ordered, and each of them has a successor (plus one). Thus we are essentially using first-order logic with a predefined discrete linear order. This logic is closely related to linear temporal logic, which is a main formalism for specifying concurrent programs (Pnueli 1981).

Given a program, we translate it to a first-order theory that captures the relationship between the input and output values of the program variables, independent of what one may want to prove about the program. For instance, trivially, the following assignment

```
X = X+Y
```

can be captured by the following two axioms:

$$X' = X + Y,$$
$$Y' = Y,$$

where X and Y denote the initial values of the corresponding program variables and X' and Y' their values after the statement is performed. Obviously, the question is how the same can be done for loops. This is where quantification over natural numbers come in. Consider the following while loop

```
while X < M do { X = X+1 }
```

It can be captured by the following set of axioms:

$$M' = M,$$
$$X \geq M \rightarrow X' = X,$$
$$X < M \rightarrow X' = X(N),$$
$$X(0) = X,$$
$$\forall n. X(n+1) = X(n) + 1,$$
$$X(N) \geq M,$$
$$\forall n. n < N \rightarrow X(n) < M,$$

where N is a natural number constant denoting the total number of iterations that the loop runs to termination, and $X(n)$ the value of X after the nth iteration. Thus the third axiom says that if the program enters the loop, then the output value of the program variable X, denoted by X', is $X(N)$, the value of X when the loop exits.

The purpose of this paper is to describe how this set of axioms can be systematically generated, and show by some examples how reasoning can be done with this set of axioms. Without going into details, one can already see that unlike Hoare's logic, our axiomatization does not make use of loop invariants. One can also see that unlike typical temporal logic specification of a program, we do not need a transition system model of the program, and do not need to keep track of program execution traces. We will discuss related work in more detail later.

*This work was supported in part by HK RGC under GRF 616013.
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Preliminaries

We use a typed first-order language. We assume a type for natural numbers (non-negative integers). Depending on the programs, other types such as integers may be used. For natural numbers, we use constant 0, linear ordering relation $<$ (and $\leq$), successor function $n+1$, and predecessor function $n-1$. We follow the convention in logic to use lower case letters, possibly with subscripts, for logical variables. In particular, we use m and n for natural number variables, and x, y, and z for generic variables. The variables in a program will be treated as functions in logic, and written as either upper case letters or strings of letters.

We use the following shorthands. The conditional expression:

$$e_1 = \text{ if } \varphi \text{ then } e_2 \text{ else } e_3$$

is a shorthand for the conjunction of the following two sentences:

$$\forall \vec{x}.\varphi \rightarrow e_1 = e_2,$$
$$\forall \vec{x}.\neg\varphi \rightarrow e_1 = e_3,$$

where $\vec{x}$ are all the free variables in φ and e_i, $i = 1, 2, 3$. Typically, all free variables in φ occur in e_1.

Our most important shorthand is the following expression which says that e is the smallest natural number that satisfies $\varphi(n)$:

$$smallest(e, n, \varphi)$$

is a shorthand for the following formula:

$$\varphi(n/e) \wedge \forall m.m < e \rightarrow \neg\varphi(n/m),$$

where n is a natural number variable in φ, m a new natural number variable not in e or φ, $\varphi(n/e)$ the result of replacing n in φ by e, similarly for $\varphi(n/m)$. For example, $smallest(M, k, k < N \wedge found(k))$ says that M is the smallest natural number such that $M < N \wedge found(M)$:

$$M < N \wedge found(M) \wedge \forall n.n < M \rightarrow \neg(n < N \wedge found(n)).$$

Finally, we use the convention that free variables in a displayed sentence are implicitly universally quantified from outside. For instance,

$$n < M \rightarrow \neg(n < N \wedge found(n))$$

stands for $\forall n.n < M \rightarrow \neg(n < N \wedge found(n))$. Notice however, that in the macro $smallest(M, k, k < N \wedge found(k))$, k is not a free variable.

The following two useful properties about the smallest macro are easy to prove.

Proposition 1 *If $\exists n\varphi(n)$, then $\exists m.smallest(m, n, \varphi(n))$.*

Proposition 2 *If $smallest(N, n, \varphi(n)) \wedge N > 0$, then $\varphi(N) \wedge \neg\varphi(N-1)$. Furthermore, if $\varphi(n)$ has the following property*

$$\exists n[(\forall m.m > n \rightarrow \varphi(m)) \wedge (\forall m.m \leq n \rightarrow \neg\varphi(m))] \quad (1)$$

then

$$smallest(N, n, \varphi(n)) \equiv \varphi(N) \wedge \neg\varphi(N-1).$$

Proof: If $smallest(N, n, \varphi(n))$, then $\varphi(N)$ and $\forall m.m < N \rightarrow \neg\varphi(m)$. Since $N > 0$, thus $\neg\varphi(N-1)$. Now suppose that $\varphi(N) \wedge \varphi(N-1)$. By (1), for some natural number M,

$$[(\forall m.m > M \rightarrow \varphi(m)) \wedge (\forall m.m \leq M \rightarrow \neg\varphi(m))].$$

This means that $M = N-1$, and $smallest(N, n, \varphi(n))$. ■

A simple class of programs

Consider the following simple class of programs:

```
E ::= array(E,...,E) |
      operator(E,...,E)
B ::= E = E |
      boolean-op(B,...,B)
P ::= array(E,...,E) = E |
      if B then P else P |
      P; P |
      while B do P
```

Here arrays are program variables, and operators are functions. The difference between the two is that the former can occur in the left hand side of an assignment and their values be changed while the latter cannot. Notice that instead of, for example in the case of two dimensional array, the notation of "array[i][j]" commonly used in programming languages to refer to an array element, we use the notation "array(i,j)" more commonly used in mathematics and logic.

As one can see, programs here are constructed using assignments, sequences, if-then-else, and while loops. Other constructs such as if-then and for-loop can be defined using these constructs. For instance, "if B then P" can be defined as "if B then P else X=X".

Given a program P and a set $\vec{X}$ of program variables including all variables used in P, we define inductively the set of axioms for P and $\vec{X}$, written $A(P, \vec{X})$, as follows:

- If P is

  `V(E1,...,Ek) = E`

  then $A(P, \vec{X})$ consists of following axioms that say that only the value of `V(E1,...,Ek)` is possibly changed:

  $$V'(\vec{x}) = \text{ if } (x_1 = E_1 \wedge \cdots \wedge x_k = E_k) \text{ then } E \text{ else } V(\vec{x}),$$
  $$X'(\vec{y}) = X(\vec{y}), \ X \in \vec{X} \text{ and } X \text{ different from } V$$

  where $\vec{x} = (x_1, ..., x_k)$, and k is the arity of the program variable (array) V. Recall that by our convention, these variables are universally quantified.

- If P is

  `if B then P1 else P2`

  then $A(P, \vec{X})$ is constructed from $A(P_1, \vec{X})$ and $A(P_2, \vec{X})$ as follows:

  $$B \rightarrow \varphi, \text{ for each } \varphi \in A(P_1, \vec{X}),$$
  $$\neg B \rightarrow \varphi, \text{ for each } \varphi \in A(P_2, \vec{X}).$$

- If P is

  `P1; P2`

  Then $A(P, \vec{X})$ is constructed from $A(P_1, \vec{X})$ and $A(P_2, \vec{X})$ by connecting the outputs of P_1 with the inputs of P_2 as follows:

  $$\varphi(\vec{X}'/\vec{Y}), \text{ for each } \varphi \in A(P_1, \vec{X}),$$
  $$\varphi(\vec{X}/\vec{Y}), \text{ for each } \varphi \in A(P_2, \vec{X}),$$

where $\vec{Y} = (Y_1, ..., Y_k)$ is a tuple of new program variables (array names) such that each Y_i is of the same arity as X_i in $\vec{X}$, $\varphi(\vec{X}'/\vec{Y})$ is the result of replacing in φ each occurrence of X_i' by Y_i, and similarly for $\varphi(\vec{X}/\vec{Y})$. By renaming if necessary, we assume here that $A(P_1, \vec{X})$ and $A(P_2, \vec{X})$ have no common program variables except those in $\vec{X}$.

- If P is

  `while B do P1`

  Then $A(P, \vec{X})$ is constructed by adding an index parameter n to all functions and predicates in $A(P_1, \vec{X})$ to record their values after the body P_1 has been executed n times. Formally, it consists of the following axioms:

$$\varphi[n], \text{ for each } \varphi \in A(P_1, \vec{X}),$$
$$X_i(\vec{x}) = X_i(\vec{x}, 0), \text{ for each } X_i \in \vec{X}$$
$$smallest(N, n, \neg B[n]),$$
$$X_i'(\vec{x}) = X_i(\vec{x}, N), \text{ for each } X_i \in \vec{X}$$

  where n is a new natural number variable not already in φ, and N a new constant not already used in $A(P_1, \vec{X})$, and for each formula or term α, $\alpha[n]$ denotes the value of α after the body P_1 has been executed n times, and is obtained from α by performing the following recursive substitutions:

  - for each X_i in $\vec{X}$, replace all occurrences of $X_i'(e_1, ..., e_k)$ by $X_i(e_1[n], ..., e_k[n], n+1)$, and
  - for each program variable X in α, replace all occurrences of $X(e_1, ..., e_k)$ by $X(e_1[n], ..., e_k[n], n)$. Notice that this replacement is for every program variable X, including those temporary program variables not in $\vec{X}$, but introduced during the construction of $A(P_1, \vec{X})$.

While we have used $A(P, \vec{X})$ to denote "the" set of axioms for P and $\vec{X}$, the construction above does not yield a unique set of axioms as the new program variables (functions in our first-order language) introduced when constructing axioms for program sequences are not unique. However, $A(P, \vec{X})$ is unique upto the renaming of these new program variables. In particular, any two different sets of these axioms are logically equivalent when considering only program variables from $\vec{X}$, i.e. when the new program variables are "forgotten". More precisely, given two theories Σ_1 and Σ_2, we say that they are equivalent when considering a subset Ω of their vocabularies if any model M_1 of Σ_1 can be modified into a model M_2 of Σ_2 such that M_1 and M_2 agree on Ω, and conversely any model of Σ_2 can be similarly modified into a model of Σ_1.

It is easy to see the following "local" property of our construction, similar to the "frame rule" in separation logic.

Proposition 3 *Let $\vec{Y}$ be a tuple of program variables that are not in P and not used in $A(P, \vec{X})$. Then considering only $\vec{X} \cup \vec{Y}$, $A(P, \vec{X} \cup \vec{Y})$ is equivalent to the union of $A(P, \vec{X})$ and the set of following "frame axioms":*

$$Y'(\vec{y}) = Y(\vec{y}), \textit{ for each } Y \in \vec{Y}$$

The construction rule for a sequence $P; Q$ can also be modified so that new program variables only need to be introduced for those that occur in both P and Q.

Proposition 4 *Let $\vec{X}$ be a tuple of program variables including those used in either P or Q, and $\vec{V} = (V_1, ..., V_k)$ the tuple of program variables used in both P and Q (thus a subset of $\vec{X}$). When considering only $\vec{X}$, $A(P; Q, \vec{X})$ is equivalent to the set of following axioms:*

$$\varphi(\vec{V}'/\vec{Y}), \textit{ for each } \varphi \in A(P, \vec{X}),$$
$$\varphi(\vec{V}/\vec{Y}), \textit{ for each } \varphi \in A(Q, \vec{X}),$$

where $\vec{Y} = (Y_1, ..., Y_k)$ is a tuple of new program variables such that each Y_i is of the same arity as V_i in $\vec{V}$. Again we assume that, by renaming if necessary, $A(P, \vec{X})$ and $A(Q, \vec{X})$ have no common program variables other than those in $\vec{X}$.

The following important property about our axiomatization says that we do not need to wait until we have the full set of axioms to do simplification. During the construction of the axioms for a program, we can simplify first the axioms for its subprograms. This greatly simplifies the above recursive procedure for constructing axioms of a program.

Proposition 5 *Let $\vec{X}$ be a tuple of program variables, including all those that occur in program P. For any subprogram P', if T is equivalent to $A(P', \vec{X})$ when considering only $\vec{X}$, then if we use T instead of $A(P', \vec{X})$ in computing $A(P, \vec{X})$, the resulting theory is equivalent to $A(P, \vec{X})$ when considering only $\vec{X}$ as well.*

Notice that in the above proposition, when we use T instead of $A(P', \vec{X})$ in computing $A(P, \vec{X})$, we assume that we will also rename program variables when necessary to avoid name conflicts. For example, if P is $P_1; P_2$, and a theory equivalent to $A(P_1, X)$ is

$$X' = Y \wedge Y = X + 1. \tag{2}$$

If $A(P_2, X)$ also mentions the temporary variable Y, then we need to rename either the Y in (2) or the Y in $A(P_2, X)$ when constructing $A(P, X)$.

Before we consider more interesting examples, we illustrate our construction of $A(P, \vec{X})$ using two simple programs.

A simple sequence

Consider the following program P and two program variables X_1 and X_2 (notice that X_1 is used in P, but X_2 is not):

```
X1 = 1; X1 = X1+1
```

$A(X1 = 1, (X_1, X_2))$ is the set of the following two sentences

$$X_1' = 1,$$
$$X_2' = X_2$$

and $A(X1 = X1 + 1, (X_1, X_2))$ the set of following two sentences:

$$X_1' = X_1 + 1,$$
$$X_2' = X_2$$

Thus $A(P, (X_1, X_2))$ is

$$Y_1 = 1,$$
$$Y_2 = X_2,$$
$$X_1' = Y_1 + 1,$$
$$X_2' = Y_2$$

Eliminating the helper names (temporary program variables) Y_1 and Y_2, we get $X_1' = 2$ and $X_2' = X_2$.

A simple loop

Consider the following program P with a simple loop.

```
while I < N do
  if X < A(I) then X = A(I);
  I = I+1
```

Notice that the program variables are X, A, I, and N. Among them, A is unary (a list), and the rest are 0-ary (constants).

Let P_1 be the body of the loop. $A(P_1, (X, A, I, N))$ is the set of following sentences (Y_1, Y_2, Y_3, Y_4 are helper names):

$$Y_1 = \text{ if } X < A(I) \text{ then } A(I) \text{ else } X,$$
$$Y_2(x) = \text{ if } X < A(I) \text{ then } A(x) \text{ else } A(x),$$
$$Y_3 = \text{ if } X < A(I) \text{ then } I \text{ else } I,$$
$$Y_4 = \text{ if } X < A(I) \text{ then } N \text{ else } N,$$
$$X' = Y_1,$$
$$A'(x) = Y_2(x),$$
$$I' = Y_3 + 1,$$
$$N' = Y_4.$$

Instead of using this set to compute $A(P, (X, A, I, N))$, by Proposition 5, we can simplify it first by eliminating the helper names Y_1, Y_2, Y_3, Y_4, and get the following equivalent set of axioms:

$$X' = \text{ if } X < A(I) \text{ then } A(I) \text{ else } X,$$
$$A'(x) = A(x),$$
$$I' = I + 1,$$
$$N' = N.$$

Thus $A(P, (X, A, I, N))$ is

$$X(0) = X,$$
$$A(x, 0) = A(x),$$
$$I(0) = I,$$
$$N(0) = N,$$
$$X(n+1) = \text{ if } X(n) < A(I(n), n) \text{ then } A(I(n), n) \text{ else } X(n),$$
$$A(x, n+1) = A(x, n),$$
$$I(n+1) = I(n) + 1,$$
$$N(n+1) = N(n),$$
$$smallest(M, n, \neg I(n) < N(n)),$$
$$X' = X(M),$$
$$A'(x) = A(x, M),$$
$$I' = I(M),$$
$$N' = N(M).$$

Clearly $A(x)$ and N do not change: $A(x, n) = A(x)$ and $N(n) = N$. So we get the following sentences by expanding the $smallest$ macro:

$$X(0) = X,$$
$$I(0) = I,$$
$$X(n+1) = \text{ if } X(n) < A(I(n)) \text{ then } A(I(n)) \text{ else } X(n),$$
$$I(n+1) = I(n) + 1,$$
$$I(M) \geq N,$$
$$n < M \rightarrow I(n) < N,$$
$$X' = X(M),$$
$$A'(x) = A(x),$$
$$I' = I(M),$$
$$N' = N.$$

Now suppose that initially $I = 0$. Solving the recurrence:

$$I(0) = 0,$$
$$I(n+1) = I(n) + 1$$

we have $I(n) = n$. It is also easy to see that $M = N$, so we can eliminate $I(n)$ and M and get the following axioms:

$$X(0) = X,$$
$$X(n+1) = \text{ if } X(n) < A(n) \text{ then } A(n) \text{ else } X(n),$$
$$X' = X(N),$$
$$A'(x) = A(x),$$
$$I' = N,$$
$$N' = N.$$

An example assertion to prove about the program is the following

$$0 \leq n < N \rightarrow X' \geq A(n), \tag{3}$$

which is equivalent to

$$0 \leq n < N \rightarrow X(N) \geq A(n),$$

which is easily proved by induction on N using the recurrence about $X(n+1)$.

Partial and total correctness

A program is partially correct w.r.t. a specification if the program satisfies the specification when it terminates. It is totally correct if it is partially correct and terminates.

In our framework, a program P with variables $\vec{X}$ is represented by a set of sentences, $A(P, \vec{X})$. Whatever properties that one wants to show about P are proved using this set

of sentences. A partial correctness corresponds to proving a sentence about $\vec{X}$ and $\vec{X}'$ from $A(P, \vec{X})$. An example is the assertion (3) above for the simple loop. On the other hand, termination of a program is proved by showing that the new natural number constants introduced by the loops and used in the $smallest$ macro expressions are well-defined. For instance, for the above simple loop, the smallest macro is $smallest(M, n, \neg I(n) < N(n))$. The fact that there is indeed a natural number M that satisfies this macro expression follows from Proposition 1 and the fact that $I(N) \geq N$ holds.

If a loop does not terminate, then its smallest macro will cause a contradiction. For instance, consider the following loop:

```
while I < M do
  if I>0 then I = I+1.
```

If initially $I = 0$ and $M > 0$, then it will loop forever. Our axioms for the loop are:

$$
\begin{aligned}
&I' = I(N) \wedge M' = M,\\
&I(0) = I,\\
&I(n+1) = \text{ if } I(n) > 0 \text{ then } I(n) + 1 \text{ else } I(n),\\
&n < N \rightarrow I(n) < M,\\
&I(N) \geq M.
\end{aligned}
$$

If we add $I = 0 \wedge 0 < M$ to these axioms, we will conclude $\forall n.I(n) < M$, which contradicts the last axiom $I(N) \geq M$.

Related work

Our formalization of the simple loop above also illustrates the difference between our approach and Hoare's logic, arguably the dominant approach for reasoning about non-parallel imperative computer programs. To begin with, an assertion like (3) would be represented by a triple like

$$\{I = 0\}P\{\forall m(0 \leq m < N \rightarrow X \geq A(m))\}$$

in Hoare's logic. To prove this assertion, one would need to find a suitable "loop invariant", a formula that if true initially will continue to be true after each iteration. In general, there are infinite number of such loop invariants. The key is to find one that, in conjunction with the negation of the loop condition, can entail the postcondition in the assertion. For this simple loop, the following is such a loop invariant:

$$\forall m(I_0 \leq m < I \rightarrow X \geq A(m)).$$

Finding suitable loop invariants is at the heart of Hoare's logic, and it is not surprising that there has been much work on discovering loop invariants (e.g. (Wegbreit 1974; Bjørner, Browne, and Manna 1997; Ernst et al. 2001; 2007; Nguyen et al. 2012)).

In comparison, our proof of (3) uses ordinary mathematical induction and recurrences on $I(n)$ and $X(n)$.

Another difference between our approach and Hoare's logic is that Hoare's logic is a set of general rules about program assertions, while we provide a translation from programs to first-order theories with quantification over natural numbers. Once the translation is done, assertions about it are proved with respect to the translated first-order theory, without reference to the original program. This is similar to Pnueli's temporal logic approach to program semantics (Pnueli 1981). According to a common classification used in formal method community (cf. (Kozen and Tiuryn 1990; Emerson 1990)): approaches like Hoare's logic and dynamic logic are *exogenous* in that they have programs explicitly in the language, while temporal logic approach to program semantics is typically *endogenous* in that a fixed program is often assumed and a program execution counter part of the specification language. Our approach is certainly not exogenous. It is a little endogenous as we use natural numbers to keep track of loop iterations, but not as endogenous as typical temporal logic specifications which requires program counters to be part of states. In particular, our mapping from programs to theories is compositional, build up from the structure of the program. Barringer *et al.* (1984) proposed a compositional approach using temporal logic, but only in the style of Hoare's logic, using Hoare triples. However, a caveat is that so far temporal logic approach to program semantics has been mainly for concurrent programs, while what we have proposed is for non-parallel programs. Given the close relationship between temporal logics and first-order logic with a linear order, if there are no nested loops, then our translation can be reformulated in a temporal logic. It is hard to see how this can be done when there are nested loops, as this will lead to nested time lines, modeled here by predicates with multiple natural number arguments. Of course, one can always construct a transition graph of a program, and model it in a temporal logic. But then the structure of the original program is lost.

We are certainly not the first to use first-order logic with a linear order to model dynamic systems. For instance, it has been used to model Turing machines in the proof of Trakhtenbrot's theorem in finite model theory (see, e.g. (Libkin 2004)).

A closely related work is Charguéraud's characteristic formulas for functional programs (Charguéraud 2010; 2011). However, these formulas are higher-order formulas that reformulate Hoare's rules by quantifying over preconditions and postconditions.

Our use of natural numbers as "indices" to model iterations is similar to Wallace's use of natural numbers to model rule applications in his semantics for logic programs (Wallace 1993).

While we use natural numbers to formalize loops, Levesque *et al.* (1997) used second-order logic to capture Golog programs with loops in the situation calculus. Recently, Lin (2014) showed that under the foundational axioms of the situation calculus, Golog programs can be defined in first-order logic as well. However, the crucial difference between the work here and the work in the situation calculus on Golog is that our axioms try to capture the changes of states in terms of values of program variables, while the semantics of Golog programs is more about defining legal sequences of executions. To illustrate the difference here, consider a program consists of assignments that make no change (nil actions). For this program, it would still be non-

trivial to define sequences of legal executions, although it does not matter which sequences are legal as none of them change the values of program variables. Another difference is that we consider only assignments and deterministic programs, while Golog programs allow any actions that can be axiomatized by successor state axioms, and can have nondeterministic choices.

However, perhaps the best way to show the correctness of our axiomatization is to connect it with Golog in the situation calculus. Given a program P, and a tuple $\vec{X}$ of program variables that include all those used in P, we can consider each X in $\vec{X}$ as a functional fluent in the situation calculus, and write successor state axioms for these fluents, assuming that the assignments used in P are the only actions. We can then show that M is a model of $A(P, \vec{X})$ iff there is a situation calculus model W such that for each $X \in \vec{X}$, $X^M = X^W(S_0)$, and $(X')^M = X^W(S_1)$, where S_1 is such that $W \models Do(P, S_0, S_1)$ (a program P here can be considered as a Golog complex action in a straightforward way). We omit the detail here as this should present no conceptual difficulties.

Cohen's integer division algorithm

For a more complex example, consider the following program P which implements the well-known Cohen's integer division algorithm (Cohen 1990) (our program below is adapted from (Nguyen et al. 2012)). It has two loops, one nested inside another.

```
// X and Y are two input integers
  Q=0; // quotient
  R=X; // remainder
  while (R >= Y) do {
    A=1;
    B=Y;
    while (R >= 2*B) do {
      A = 2*A;
      B = 2*B;
    }
    R = R-B;
    Q = Q+A
  }
//  return Q = X/Y;
```

The program variables are A, B, Q, R, X, Y, where X and Y are inputs, and Q is the output. Let $\vec{X} = (A, B, Q, R, X, Y)$. There are two loops. Let's name the inner loop $Inner$, and outer loop $Outer$. When computing $A(P, \vec{X})$, we again consider only equivalence under $\vec{X}$ and use Proposition 5 to simplify the process.

It is easy to see that $A(P, \vec{X})$ is equivalent to the union of $A(Outer, \vec{X})$ and $\{Q = 0 \wedge R = X\}$. To compute $A(Outer, \vec{X})$, we compute first $A(Inner, \vec{X})$, which is equivalent to the set of following sentences:

$$
\begin{aligned}
&A(n+1) = 2A(n),\\
&B(n+1) = 2B(n),\\
&Q(n+1) = Q(n),\\
&R(n+1) = R(n),\\
&X(n+1) = X(n),\\
&Y(n+1) = Y(n),\\
&A(0) = A,\\
&B(0) = B,\\
&Q(0) = Q,\\
&R(0) = R,\\
&X(0) = X,\\
&Y(0) = Y,\\
&smallest(N, n, R(n) < 2B(n)),\\
&A' = A(N),\\
&B' = B(N),\\
&Q' = Q(N),\\
&R' = R(N),\\
&X' = X(N),\\
&Y' = Y(N).
\end{aligned}
$$

Solving the recurrences, we have

$$
\begin{aligned}
&A(n) = 2^n A,\\
&B(n) = 2^n B,\\
&Q(n) = Q,\\
&R(n) = R,\\
&X(n) = X,\\
&Y(n) = Y\\
&smallest(N, n, R < 2^{n+1}B),\\
&A' = 2^N A,\\
&B' = 2^N B,\\
&Q' = Q,\\
&R' = R,\\
&X' = X,\\
&Y' = Y.
\end{aligned}
$$

We can now eliminate terms like $A(n)$ and $B(n)$, expand the smallest macro expression, and obtain $A(Inner, \vec{X})$ as the set of following sentences:

$$
\begin{aligned}
&R < 2^{N+1}B,\\
&m < N \rightarrow R \geq 2^{m+1}B,\\
&A' = 2^N A,\\
&B' = 2^N B,\\
&Q' = Q,\\
&R' = R,\\
&X' = X,\\
&Y' = Y.
\end{aligned}
$$

Thus the set of sentences for the body of the loop $Outer$ is equivalent to the set of the following sentences:

$$
\begin{aligned}
&R < 2^{N+1}Y,\\
&m < N \rightarrow R \geq 2^{m+1}Y,
\end{aligned}
$$

$$A' = 2^N,$$
$$B' = 2^N Y,$$
$$Q' = Q + A',$$
$$R' = R - B',$$
$$X' = X,$$
$$Y' = Y.$$

Thus $A(Outer, \vec{X}) \cup \{Q = 0, R = X\}$ is equivalent to

$$R(n) < 2^{N(n)+1} Y(n),$$
$$m < N(n) \rightarrow R(n) \geq 2^{m+1} Y(n),$$
$$A(n+1) = 2^{N(n)},$$
$$B(n+1) = 2^{N(n)} Y(n),$$
$$Q(n+1) = Q(n) + 2^{N(n)},$$
$$R(n+1) = R(n) - 2^{N(n)} Y(n),$$
$$X(n+1) = X(n),$$
$$Y(n+1) = Y(n),$$
$$A(0) = A,$$
$$B(0) = B,$$
$$Q(0) = 0,$$
$$R(0) = X,$$
$$X(0) = X,$$
$$Y(0) = Y,$$
$$smallest(M, n, R(n) < Y(n)),$$
$$A' = A(M),$$
$$B' = B(M),$$
$$Q' = Q(M),$$
$$R' = R(M),$$
$$X' = X(M),$$
$$Y' = Y(M).$$

Now get rid of $X(n)$ and $Y(n)$ as they do not change: $X(n) = X$ and $Y(n) = Y$, get rid of A and B as they are irrelevant now, and expand the smallest macro expression, we obtain $A(P, \vec{X})$ as the set of following sentences:

$$R(n) < 2^{N(n)+1} Y,$$
$$m < N(n) \rightarrow R(n) \geq 2^{m+1} Y,$$
$$Q(n+1) = Q(n) + 2^{N(n)},$$
$$R(n+1) = R(n) - 2^{N(n)} Y,$$
$$Q(0) = 0,$$
$$R(0) = X,$$
$$R(M) < Y,$$
$$m < M \rightarrow R(m) \geq Y,$$
$$Q' = Q(M),$$
$$R' = R(M).$$

From these axioms, we can show the correctness of Cohen's algorithm by proving the following two properties:

$$0 \leq R' < Y,$$
$$X = Q'Y + R'.$$

For the first property, $R' < Y$ trivially follows from the condition of the $Outer$ loop. For $R' \geq 0$, we have $R' = R(M) = R(M-1) - 2^{N(M-1)} Y$. By the axiom

$$m < N(n) \rightarrow R(n) \geq 2^{m+1} Y,$$

let $n = M-1$ and $m = N(M-1)-1$, we have $R(M-1) \geq 2^{(N(M-1)-1)+1} Y = 2^{N(M-1)} Y$. Thus $R' \geq 0$. For the second property, we have

$$\begin{aligned} & Q'Y + R' \\ &= Q(M)Y + R(M) \\ &= (Q(M-1) + 2^{N(M-1)})Y + R(M-1) - 2^{N(M-1)} Y \\ &= Q(M-1)Y + R(M-1) \\ &= \cdots = Q(0)Y + R(0) = X. \end{aligned}$$

Again this is partial correctness. To prove the termination, we need to show that the new terms introduced by the smallest macro expressions are all well-defined. For this program, it means that M (the outer loop counter) is bounded, and for every n, $N(n)$ (the inner loop counter for each outer loop iteration) is bounded. By Proposition 1, these can be proved by showing the following two properties:

$$\exists m.R(m) < Y,$$
$$\forall n \exists m.R(n) < 2^{m+1} Y.$$

Again we remark that we relied on mathematical induction in our proof and made no use of loop invariants. Notice also that our proof actually shows that for integer division, any program of the following form is correct:

```
// X and Y are two input integers
  Q=0; // quotient
  R=X; // remainder
  while (R >= Y) do {
    A=1;
    B=Y;
    while (R >= k*B) do {
      A = k*A;
      B = k*B;
    }
    R = R-B;
    Q = Q+A
  }
//  return Q = X/Y;
```

where $k > 1$ can be any constant.

Functions

One may ask how general our proposed approach is. Can it be done for programs with more complex structures like pointers, functions, classes, concurrency? We believe so. We have extended it to pointers and functions. Classes should present no problem as they are basically user defined types. While we have not done it for concurrency, we believe it can be done as well given that we were able to provide a first-order axiomatization of ConGolog (Lin 2014). In this section, we describe how the same approach can be used to axiomatize programs with user defined functions.

In practice, a program consists of a set of functions. To illustrate how we can handle functions, including recursive functions, consider the following class of programs:

```
E ::= array(E,...,E) |
      operator(E,...,E) |
      function(E,...,E) |
B ::= E = E |
      boolean-op(B,...,B)
Body ::= array(E,...,E) = E |
         if B then P else P |
         P; P |
         while B do P |
         return E
F ::= function(variable,...,variable)
      { Body }
P ::= F | P; P
```

Thus a program is a collection of functions. Presumably, one of them is the "main" function, the one that will be executed first when the program is run. In some programming languages, these functions can communicate by sharing some global variables. To simplify things a bit, we assume here that there are no global variables, and that all program variables in the body of a function must occur in the parameter list of the function.

If P is $F_1; \cdots; F_k$, then the set of axioms for P is the union of the sets of axioms for F_i, $1 \leq i \leq k$, with renaming of program variables in them if needed to avoid conflict of names.

Given a function definition $f(\vec{X})\{Body\}$, the set of sentences for it, written $A(f)$, is

$$\forall \vec{x} \varphi(\vec{X}/\vec{x})(Result'/f(\vec{x})), \varphi \in A(Body, \vec{X} \cup \{Result\}),$$

where

- $\varphi(\vec{X}/\vec{x})(Result'/f(\vec{x}))$ is the result of replacing in φ each X_i in $\vec{X}$ by x_i, X_i' by a new function name $g(\vec{x})$, and $Result'$ by $f(\vec{x})$. We assume that $Result$ is a reserved word used to denote the value of the function. Notice that once we replace each X_i by a variable x_i, X_i', the value of X_i when the function exits, is no longer relevant. Here we just replace it by a dummy new function g.
- $A(Body, \vec{X} \cup \{Result\})$ is defined as before, except that when $Body$ is `return E`, the axioms are

$$Result' = E,$$
$$X_i'(\vec{x}) = X_i(\vec{x}),\ X_i \text{ is a program variable.}$$

Notice that according to our axiomatization here, while the body of a function may execute the return statement multiple times, only the last time matters. For example, given

```
foo() { return 1; return 2 }
```

only the second return statement is meaningful, and the function is captured by the axiom $foo() = 2$. One could argue that it does not make sense for more than one instances of the return statement to be executed, and it is the programmer's responsibility to make sure that this does not happen. Alternatively, one can assume that as soon as a return statement is executed, the function exits. This can be modeled by introducing a special flag $Exit$, and replace each return statement by

```
if -Exit then {return E; Exit = true}
```

For a more meaningful example, consider the following two mutually defined functions $isEven$ and $isOdd$:

```
isEven(N) {
  if N=0 then return true
         else return -isOdd(N-1) }
isOdd(N) {
  if N=0 then return false
         else return -isEven(N-1) }
```

Suppose that we denote the body of $isEven(N)$ by $Body1$, and that of $isOdd(N)$ by $Body2$. Then $A(Body1, (N, Result))$ consists of the following axioms:

$$N' = N,$$
$$Result' = \text{ if } N = 0 \text{ then } true \text{ else -}isOdd(N-1)$$

and similarly for $A(Body2, (N, Result))$:

$$N' = N,$$
$$Result' = \text{ if } N = 0 \text{ then } false \text{ else -}isEven(N-1)$$

Thus $A(isOdd) \cup A(isEven)$ is

$$f(x) = x,$$
$$isEven(x) = \text{ if } x = 0 \text{ then } true \text{ else -}isOdd(x-1),$$
$$g(x) = x,$$
$$isOdd(x) = \text{ if } x = 0 \text{ then } false \text{ else -}isEven(x-1),$$

where f and g are two new functions used to denote the values of x when the functions $isEven(x)$ and $isOdd(x)$, respectively, return. They are irrelevant, so the two corresponding axioms can be deleted. By induction on n, it is easy to prove that the following hold for all $n \geq 0$:

$$isEven(2n) = true,$$
$$isOdd(2n) = false,$$
$$isEven(2n+1) = false,$$
$$isOdd(2n+1) = true.$$

Now consider the following program with a type definition:

```
List ::= [] | a::List

length(X:List) {
 if X=[] then return 0
   else return length(tail(X))+1}
tail(X:List) {
 if X=[] then return []
   else if X=a::X1 then return X1}
append(X:List, Y:List) {
 if X=[] then return Y
   else if X=a::X1
        then return a::append(X1,Y)}
```

To model the data type `List`, we introduce a corresponding $List$ sort in our first-order language, and write $(x : List)$

to mean that x is of sort $List$. In first-order terms, the definition of $List$ yields the following axioms:

$$(\forall x\!:\!List).x = [] \vee \exists a(\exists y\!:\!List)x = a :: y,$$
$$\forall a, b(\forall x, y\!:\!List).a :: x = b :: y \rightarrow (a = b \wedge x = y),$$
$$\forall a(\forall x\!:\!List)[] \neq a :: x,$$

and the three functions yield the following axioms:

$$\begin{aligned}
&(\forall x\!:\!List).length(x) = \text{ if } x = [] \text{ then } 0\\
&\qquad \text{else } length(tail(x)) + 1,\\
&(\forall x\!:\!List).tail(x) = \text{ if } x = [] \text{ then } []\\
&\qquad \text{else if } \exists a(\exists y\!:\!List)x = a :: y \text{ then } y,\\
&(\forall x, y\!:\!List).append(x, y) = \text{ if } x = [] \text{ then } y\\
&\qquad \text{else if } \exists a(\exists x_1\!:\!List)x = a :: x_1\\
&\qquad\qquad \text{then } a :: append(x1, y).
\end{aligned}$$

With these axioms, one can prove, for example $length(a :: b :: []) = 2$. However, they are not sufficient for proving general properties like the following simple one:

$$(\forall x, y\!:\!List)length(append(x, y)) = length(x)+length(y).$$

To prove properties like this, we need induction on lists. This can be done by using a second-order axiom on sort $List$, similar to the one on natural numbers. However, since we already have natural numbers, this is not necessary. We can introduce lists of n elements, and define a list to be a list of n elements, for some n. This way, we can use mathematical induction on natural numbers to prove inductive properties about lists. We show how this is done here. We introduce a binary predicate $List(x, n)$, meaning that x is a list with exactly n elements:

$$(\forall x\!:\!List)\exists n.List(x, n),$$
$$List(x, 0) \equiv x = [],$$
$$List(x, n + 1) \equiv (\exists a)(\exists y\!:\!List).x = a :: y \wedge List(y, n)$$

We first show that if x is a list, then there is a unique n such that $List(x, n)$ holds:

$$List(x, n) \wedge List(x, m) \rightarrow m = n. \qquad (4)$$

Suppose x is a list, and $List(x, m)$ and $List(x, n)$ are true. We do simultaneous induction on n and m. If $n = 0$, then $x = []$. If $m \neq 0$, then for some k, $m = k + 1$ and $x = [] = a :: y$ for some a and list y, a contradiction with one of our axioms about lists. Thus $m = 0$ as well. Similarly, if $m = 0$, then $n = 0$ as well. Suppose $n = k_1 + 1$ and $m = k_2 + 1$, and suppose inductively that for any $i, j < max\{m, n\}$, we have that

$$List(y, i) \wedge List(y, j) \rightarrow i = j$$

for any list y. We then have $x = y_1 :: z_1$ for some list z_1 such that $List(z_1, k_1)$ holds, and $x = y_2 :: z_2$ for some list z_2 such that $List(z_2, k_2)$ holds. From $y_1 :: z_1 = y_2 :: z_2$, we have $z_1 = z_2$, thus by the inductively assumption, $k_1 = k_2$. So $m = n$. This concludes the inductive step, thus the proof of (4).

Using (4), we can then prove the induction schema on lists: for any formula $\varphi(x)$,

$\varphi([]) \wedge \forall a(\forall x\!:\!List)(\varphi(x) \rightarrow \varphi(a :: x)) \rightarrow (\forall x\!:\!List)\varphi(x)$.

Suppose the premise is true and for some list x, $\neg\varphi(x)$. Suppose x is a shortest such list: if $List(x, n)$ then for any list y, if $List(y, m) \wedge m < n$, then $\varphi(y)$ holds. Notice that the existence of such an x follows from (4). Suppose $List(x, n)$. If $n = 0$, then $x = []$, which satisfies φ, a contradiction. Suppose $n = m + 1$, then there are some a and y such that $x = a :: y \wedge List(y, m)$. By our assumption about x, $\varphi(y)$ holds. By the premise, $\varphi(a :: y)$ holds as well, a contradiction.

The same idea can be used to axiomatize in first-order logic other inductively defined data structures such as trees.

For recursive functions, a challenge is to distinguish between cycles and undefined values. Consider the following example.

```
foo(X) { if X=0 then return foo(X)
         else if x=1 then return 1}
```

With our axiomatization, the set of axioms for $foo(x)$ is equivalent to a single fact $foo(1) = 1$. It leaves completely open the possible values for $foo(x)$ when $x \neq 1$. One could argue whether this is a right formalization. But operationally, there is a difference between function calls $foo(0)$ and $foo(2)$: calling $foo(0)$ will cause a cycle, but calling foo(2) will terminate without any value being returned. The former causes stack overflow and the latter abnormal exit.

In the following, we provide an axiomatization of functions that can differentiate these two cases address. The key idea is to keep a counter of the number of times a recursive function has been called.

Let $f_1, f_2, ..., f_k$ be functions that are mutually defined recursively: $f_i(X_1, ..., X_m)\{B_i\}$. Extend these functions with one more argument:

$$f_i(X_1, ..., X_m, M)\ \{\text{if } M = 0 \text{ then } B_{i0} \text{ else } B_{i1}\}$$

where

- B_{i0} is the result of replacing each function call $f_j(T_1, ..., T_m)$ in B_i by $Cycle$, and
- B_{i1} is the result of replacing each function call $f_j(T_1, ..., T_m)$ in B_i by $f_j(T_1, ..., T_m, M - 1)$.
- M is a natural number, and $Cycle$ is a new constant.

The set $A(f_i)$ of axioms for f_i is then

$$f_i(\vec{x}) = y \equiv \exists n \forall m \geq n.f_i(\vec{x}, m) = y.$$

Consider again function $foo()$ defined above. We have

```
foo(X,M) { if M=0 then
  {if X=0 then return Cycle else
    if X=1 then return 1} else
  {if X=0 then return foo(X,M-1) else
    if X=1 then return 1}
```

and the following axioms for $foo(X)$ and $foo(X, M)$:

$$\begin{aligned}
&foo(x) = y \equiv \exists m \forall n \geq m.foo(x, n) = y,\\
&foo(0, 0) = Cycle,\\
&foo(1, 0) = 1,\\
&foo(0, n + 1) = foo(0, n),\\
&foo(1, n + 1) = 1
\end{aligned}$$

Thus $\forall n.foo(0,n) = Cycle$ and $\forall n.foo(1,n) = 1$. So $foo(0) = Cycle$ and $foo(1) = 1$. The axioms again leave open the possible values for $foo(x)$ when x is not equal to 0 or 1.

Concluding remarks

My goal is to have a translator from a full programming language like C or Java to first-order logic. In this paper, I show how this is possible for a core procedural programming language with loops and functions. Instead of loop invariants used in Hoare's logic, the approach relies on mathematical induction and recurrences. I show that even for programs with nested while loops such as Cohen's integer division, typical properties about them can be proved effectively using their corresponding first-order theories.

The complexity of the translated first-order theory from a program depends on the domain that the program is about. If all program variables are propositional, then the resulting first-order theory is decidable for proving both partial and total correctness of the program with respect to any given propositional specification. If the program is about natural numbers and involves addition and multiplication, then we may need full arithmetic to reason about it. If the program is about predicting the trajectory of a planet, then a theory of physics is needed in order to prove anything interesting about it. How to integrate logical reasoning with a domain theory has long been a challenge in AI as well as in computer science. I hope that with this work, more KR researchers will take up this challenge and start to contribute to program verification. As a step in this direction, we plan to develop a reasoner for programs about mathematics by integrating a logic solver with Mathematica[1], which has a programmable set of powerful tools for solving problems in mathematics. For instance, it can easily solve the recurrences that we have seen in this paper.

References

Barringer, H.; Kuiper, R.; and Pnueli, A. 1984. Now you may compose temporal logic specifications. In *STOC*, 51–63.

Bjørner, N.; Browne, A.; and Manna, Z. 1997. Automatic generation of invariants and intermediate assertions. *Theor. Comput. Sci.* 173(1):49–87.

Charguéraud, A. 2010. Program verification through characteristic formulae. In *ACM Sigplan Notices*, volume 45 (9), 321–332. ACM.

Charguéraud, A. 2011. Characteristic formulae for the verification of imperative programs. In *ACM SIGPLAN Notices*, volume 46 (9), 418–430. ACM.

Cohen, E. 1990. *Programming in the 1990s: An Introduction to the Calculation of Programs*. Springer-Verlag.

Dijkstra, E. W., and Scholten, C. S. 1990. *Predicate Calculus and Program Semantics*. New York: Springer-Verlag.

Dijkstra, E. 1976. *A Discipline of Programming*. Englewood Cliffs, N.J.: Prentice Hall.

[1] http://www.wolfram.com/mathematica/

Emerson, E. A. 1990. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, 995–1072. Elsevier.

Ernst, M. D.; Cockrell, J.; Griswold, W. G.; and Notkin, D. 2001. Dynamically discovering likely program invariants to support program evolution. *Software Engineering, IEEE Transactions on* 27(2):99–123.

Ernst, M. D.; Perkins, J. H.; Guo, P. J.; McCamant, S.; Pacheco, C.; Tschantz, M. S.; and Xiao, C. 2007. The daikon system for dynamic detection of likely invariants. *Science of Computer Programming* 69(1):35–45.

Harel, D. 1979. *First-Order Dynamic Logic*. New York: Springer-Verlag: Lecture Notes in Computer Science 68.

Hoare, C. 1969. An axiomatic basis for computer programming. *Comm. ACM*.

Kozen, D., and Tiuryn, J. 1990. Logics of programs. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*. Elsevier. 789–840.

Levesque, H.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. 1997. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming, Special issue on Reasoning about Action and Change* 31:59–84.

Libkin, L. 2004. *Elements of Finite Model Theory*. Springer.

Lin, F. 2014. A first-order semantics for Golog and ConGolog under a second-order induction axiom for situations. In *Proceedings of KR 2014*.

Nguyen, T.; Kapur, D.; Weimer, W.; and Forrest, S. 2012. Using dynamic analysis to discover polynomial and array invariants. In *Proceedings of 34th International Conference on Software Engineering (ICSE 2012)*, 683–693. IEEE.

Pnueli, A. 1981. The temporal semantics of concurrent programs. *Theor. Comput. Sci.* 13:45–60.

Reynolds, J. C. 2002. Separation logic: A logic for shared mutable data structures. In *Proceedings of 17th Annual IEEE Symposium on Logic in Computer Science*, 55–74. IEEE.

Wallace, M. G. 1993. Tight, consistent, and computable completions for unrestricted logic programs. *Journal of Logic Programming* 15:243–273.

Wegbreit, B. 1974. The synthesis of loop predicates. *Communications of the ACM* 17(2):102–113.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Simultaneous Learning and Prediction

Loizos Michael
Open University of Cyprus
loizos@ouc.ac.cy

Abstract

Agents in real-world environments may have only *partial* access to available information, often in an arbitrary, or hard to model, manner. By reasoning with knowledge at their disposal, agents may hope to recover some missing information. By acquiring the knowledge through a process of learning, the agents may further hope to guarantee that the recovered information is indeed correct. Assuming only a black-box access to a learning process and a prediction process that are able to cope with missing information in some principled manner, we examine how the two processes should interact so that they improve their overall joint performance. We identify natural scenarios under which the interleaving of the processes is provably beneficial over their independent use.

Introduction

Consider a medical database, with one row for each patient, and one column for each attribute of interest to some medical doctor. Due to numerous reasons, not every patient-attribute pair is associated with a definite value. Can the missing values be reliably and efficiently predicted / recovered, given the rest of the available information, and without assuming any particular structure in the way information is missing?

It is typical for research in Knowledge Representation and Reasoning (KRR) to assume that an agent (e.g., the medical doctor in our example scenario) has access to some knowledge base of rules that collectively capture certain regularities in the environment of interest. By applying such rules on the available information, the agent draws inferences on those aspects of its available inputs that are not explicitly observed, effectively recovering some of the missing information. Research, then, seeks to identify how to most appropriately represent and reason with rules for this recovery task.

A knowledge base need not comprise only infallible rules, in that rules need not be fully qualified at an individual or local level. Instead, rules could be qualified via the reasoning process, so that they are blocked from drawing an inference they support if and when another piece of evidence supports a contradictory inference, and the latter takes precedence. First, rules could be *endogenously qualified*, if they are explicitly known to be overridden by certain other rules in the knowledge base (e.g., the rule that "high fever during winter suggests a flu infection" could be overridden by another rule that "high fever near a swamp suggests a malaria infection"). Second, rules could be *exogenously qualified*, if they yield in the presence of explicit, but external to the knowledge base, information that contradicts them (e.g., the rule that "pain or pressure in the chest or arms suggests a heart attack" could yield to an explicit observation of a negative test result.

Accounting for the qualifications, the underlying assumption remains that after applying a (sound) reasoning process on the knowledge base and the available information, the drawn inferences should be accepted as correctly recovering the missing information. And they should be, if the knowledge that is available for the agent to reason with is, indeed, appropriate for the environment on which it is being used.

It has often been argued that the appropriateness of knowledge can be guaranteed if (and in many settings, only if) it is acquired through a process of induction over the agent's experiences from its environment (Valiant 2013). Indeed, by a simple, in retrospect, statistical argument (Valiant 1984), rules that have been found to be correct on past experiences are also *Probably* (i.e., except with a small probability with which the agent did not have typical experiences from its environment) *Approximately Correct* (i.e., they will predict correctly on almost all of the agent's future experiences).

To learn knowledge with the aim to recover some missing information in its future partially observable experiences, an agent has access only to such equally partially observable experiences from which it can learn. It has been shown that the PAC learning semantics (Valiant 1984) can be extended to deal with arbitrarily missing information, and that certain PAC learning algorithms can be easily modified to cope with such experiences (Michael 2010), even under the seemingly arduous requirements that we later impose on the guarantees provided by these algorithms. In the interest of generality, we shall not deal in the sequel with the exact details of how such algorithms work. Instead, we shall, henceforth, assume a black-box access to any algorithm for learning from partially observable experiences, and call it the *base algorithm*.

Applying a base algorithm on the database, a medical doctor may seek to learn a predictor P_i for each of the database attributes x_i, and apply these predictors on each row of the database to complete some of the missing information. Due to partial information and imperfect training, the predictors

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

may sometimes abstain or predict incorrectly when applied, giving rise to a database that is generally more complete, but less sound. The question we seek to answer in this work is this: *Is there a natural and general technique to improve the completeness and soundness of the above naive approach?*

It is important to clarify upfront that the notions of soundness and completeness that are used in this work are akin, but not identical, to those used in KRR research. In the latter, one typically has a semantics that determines, for each input and knowledge base, the inferences that are valid / correct (e.g., by specifying the models of the given input and knowledge base). Then, one may conceivably consider a number of different reasoning processes, and evaluate each one in terms of soundness and completeness thus: a reasoning process is sound if the inferences it produces *agree* with the inferences specified by the semantics, and it is complete if it produces *all* the inferences that are specified by the semantics.

Since knowledge bases in KRR research are typically assumed to truly capture the structure of the environment of interest, a definition of soundness and completeness against a knowledge base is appropriate. When learning is involved, however, the knowledge base is only *approximately* correct. The goal of the combined learning and reasoning processes is, then, to produce inferences not as specified against the agent's own knowledge base, but rather as specified against some "ideal" knowledge base; or in other words, to produce inferences specified by the underlying truth of things.

In effect, this work considers different policies (i.e., ways to apply learned rules) for drawing inferences, and evaluates them in terms of their soundness and completeness. More precisely, we take the stance that soundness should not be compromised (beyond the approximate correctness that cannot be avoided due to learning), and seek to find policies that draw inferences in a manner that improves completeness.

In seeking such policies we ask: When is it meaningful to *learn* the predictors (for each attribute) independently of each other? When is it meaningful to *apply* them independently of each other? Can something be gained by interleaving their learning and their application? We offer formal answers to these questions, and discuss how our results apply when knowledge is acquired through learning in other typical settings, such as supervised classification / regression.

Preliminaries and Notation

Fix a set $\mathcal{A}$ of attributes x_i, with $i \in \{1, \dots n\}$. Denote by $dom[i]$ the domain of attribute x_i, and by $dom[\mathcal{A}]$ the cross product $dom[1] \times \cdots \times dom[n]$ of all attribute domains. A ***complete assignment*** of values to the attributes $\mathcal{A}$ is denoted by $\texttt{asg} \in dom[\mathcal{A}]$, and the value of attribute x_i in $\texttt{asg}$ is denoted by $\texttt{asg}[i]$. To account for partial observability, denote by $dom^*[i]$ the extended domain $dom[i] \cup \{*\}$, and by $dom^*[\mathcal{A}]$ the cross product $dom^*[1] \times \cdots \times dom^*[n]$ of all extended attribute domains; $*$ stands for "don't know". An ***(observed) assignment*** of values to the attributes $\mathcal{A}$ is denoted by $\texttt{obs} \in dom^*[\mathcal{A}]$, and the value of attribute x_i in $\texttt{obs}$ is denoted by $\texttt{obs}[i]$. Our example medical database can, then, be represented simply as a list $\mathcal{S}^*$ of assignments.

Given $\mathcal{S}^*$, we seek a new list $\mathcal{S}'$ of assignments in which some of the $*$ values are replaced with the "actual" (as determined by nature) values, from the domain of the corresponding attributes. That is, given the partial inputs and whatever relevant domain knowledge we have at our disposal, we seek to reason and infer (some of) the information that is not explicitly given in the inputs. For instance, while the height-at-birth attribute of some patient might be missing in medical database $\mathcal{S}^*$, the patient does have some value associated with that attribute, and the goal is to accurately recover it.

Although the recovery process does not have access to the underlying reality, our analysis of the recovery process will make use of it. We shall, henceforth, consider $\mathcal{S}$ to be a list of complete assignments, and assume that $\mathcal{S}^*$ is always such that the j-th observed assignment $\texttt{obs} \in \mathcal{S}^*$ is obtained by applying an unknown fixed process $\texttt{mask}$ on the j-th complete assignment $\texttt{asg} \in \mathcal{S}$. In the general case, the ***masking process*** $\texttt{mask}$ is a (possibly stochastic) mapping from $dom[\mathcal{A}]$ to $dom^*[\mathcal{A}]$. This general treatment captures cases where masking may (arbitrarily) hide or even distort parts of the underlying reality. For this work we shall restrict our attention to masking processes that only hide (without distorting) assignments, by insisting that whenever $\texttt{obs}$ is obtained from $\texttt{mask}(\texttt{asg})$, either $\texttt{obs}[i] = *$ (i.e., information is hidden) or $\texttt{obs}[i] = \texttt{asg}[i]$ (i.e., information is preserved). Such an observed assignment $\texttt{obs}$ shall be said to ***mask*** the complete assignment $\texttt{asg}$, and each attribute $x_i \in \mathcal{A}$ whose value $\texttt{obs}[i]$ is $*$ shall be said to be ***masked*** in $\texttt{obs}$.

A ***base algorithm*** is one that given access to a list $\mathcal{S}^*$ and an attribute $x_i \in \mathcal{A}$, returns a predictor P_i. A ***predictor*** P_i for x_i is a mapping from $dom[\mathcal{A} \setminus \{x_i\}]$ to $dom[i]$ that is computable in time polynomial in n (i.e., in its input size). A given assignment $\texttt{obs} \in dom^*[\mathcal{A}]$ may not necessarily offer enough information to *uniquely* determine the value of the mapping. In such a case, and only then, P_i abstains from making a prediction. Thus, the predictor P_i is naturally lifted to a mapping from $dom^*[\mathcal{A} \setminus \{x_i\}]$ to $dom^*[i]$ to account for abstaining from predictions when sufficient information is not available to determine a definite value. To reduce notation, we shall write $P_i(\texttt{obs})$ to denote the prediction of P_i on assignment $\texttt{obs} \in dom^*[\mathcal{A}]$, with the implicit understanding that only the values of attributes $\mathcal{A} \setminus \{x_i\}$ are read from $\texttt{obs}$. When P_i abstains due to insufficient information in $\texttt{obs}$ to determine a prediction, then $P_i(\texttt{obs}) = *$.

We shall restrict our attention to ***classical*** (or monotonic) predictors, which we take to mean that when they make a definite prediction on some assignment, they do not change that prediction in the presence of more information.[1] Other than that, we shall treat the base algorithm and the returned predictors as black boxes. It is exactly therein that lies the general applicability of our framework and formal results.

Dealing with Multiple Predictors

Consider a medical database $\mathcal{S}^*$ with multiple rows, and 7 columns corresponding to real-valued patient characteristics (e.g., cholesterol level). Let the complete assignment $\texttt{asg} =$

[1] Although reasoning with non-monotonic rules has a very long and rich history in KRR research (Reiter 1980), the learning of non-monotonic rules has received considerably less attention in Learning Theory research (Schuurmans and Greiner 1994) and the practical deployment of Machine Learning techniques (Mitchell 1997).

$\langle -1, 0, 5, 0, 4, -7, 9\rangle \in \mathcal{S}$ determine the actual values that these characteristics take for patient John Doe, and let the observed assignment $\texttt{obs} = \langle -1, 0, *, 0, 4, *, 9\rangle \in \mathcal{S}^*$ correspond to John Doe's (partial) medical record as found in the medical database $\mathcal{S}^*$. Suppose that a doctor reads in a medical journal that the values of certain patient characteristics can be predicted from the values of the other patient characteristics through a set $\mathbb{P}$ of the following[2] predictors:
(i) P_1 predicts according to rule "$x_3 := |\sqrt{x_5}| + |\sqrt{x_7}|$";
(ii) P_2 predicts according to rule "$x_4 := x_2 \cdot x_6$";
(iii) P_3 predicts according to rule "$x_6 := x_3 - x_5/2$".

The doctor wishes to recover reliably as much of the information missing in $\mathcal{S}^*$, and in John Doe's medical record in particular. To this end, the doctor chooses to apply the predictors in $\mathbb{P}$ *in parallel*, in the following sense: each of the predictors is applied on obs independently of the others, the prediction of each predictor is computed, and only then the predictions are used to complete the information that is missing in John Doe's medical record. By following this process, the doctor computes the predictions of P_1, P_2, P_3 to be, respectively, 5, 0, $*$. Note that predictor P_3 abstains since its prediction cannot be determined without the value of x_3. Incorporating these predictions in obs yields $\texttt{obs}' = \langle -1, 0, 5, 0, 4, *, 9\rangle$, which is sound when compared against the underlying truth asg, and more complete than obs.

Applying predictors in parallel is not the only option. For instance, the doctor could choose to apply the predictors *sequentially*, in the following sense: apply one of the predictors on obs, compute its prediction, modify obs to incorporate that prediction, and repeat the process with another predictor. Unlike the parallel process, the second predictor will now be applied not on obs, but on the new version of John Doe's medical record that has incorporated the prediction of the first predictor. By following this process, the doctor applies P_1 on $\texttt{obs} = \langle -1, 0, *, 0, 4, *, 9\rangle$, and its prediction 5 for x_3 turns obs into $\texttt{obs}' = \langle -1, 0, 5, 0, 4, *, 9\rangle$. Subsequently, the doctor considers P_2, but seing that the value of x_4 is already known in $\texttt{obs}'$, the doctor chooses to let $\texttt{obs}'' = \texttt{obs}'$, effectively ignoring the prediction of P_2. Finally, the doctor applies P_3 on $\texttt{obs}''$, and its prediction 3 for x_6 turns $\texttt{obs}''$ into $\texttt{obs}''' = \langle -1, 0, 5, 0, 4, 3, 9\rangle$.

Evidently, the two ways of applying the very same set of predictors yield different results. Indeed, in the second case, predictor P_3 has access to a value for attribute x_3 at the time of its application, and is able to make a definite prediction. The resulting medical record $\texttt{obs}'''$ is now fully complete, but unsound when compared against the underlying truth asg, since the value of x_6 is predicted, but incorrectly.

A set of predictors may be applied in other more involved ways as well; e.g., first apply $\{P_1, P_3\}$ in parallel, and subsequently apply $\{P_2\}$. It is even possible for the value of some attribute to be predicted on multiple occasions by predictors that are applied, however, at different points in the process. Henceforth, we shall assume that an ordering of predictors is implicitly prescribed by each given $\mathbb{P}$. We shall call $\mathbb{P}$ a ***policy***, and think of it as a sequence of non-empty sets of predictors, so that the predictors in each set are applied in parallel (with each predictor in the set predicting the value of a different attribute), before those in the subsequent set are applied, and so on. We shall write $\mathbb{P}(\texttt{obs})$ to mean the final assignment obtained when predictors in $\mathbb{P}$ are applied on obs in the prescribed ordering. We shall write $\mathbb{P}(\mathcal{S}^*)$ to mean the list with elements in $\{\mathbb{P}(\texttt{obs}) \mid \texttt{obs} \in \mathcal{S}^*\}$.

Note that a prediction $\mathbb{P}(\texttt{obs}) \in dom^*[\mathcal{A}]$ of a policy $\mathbb{P}$ on obs determines values in $dom^*[\mathcal{A}]$ for all attributes $\mathcal{A}$, since every attribute on which a predictor in $\mathbb{P}$ abstains will have its value persist from what is given in $\texttt{obs} \in \mathcal{S}^*$. A policy $\mathbb{P}$ is ***complete on*** $\texttt{obs} \in \mathcal{S}^*$ if $\mathbb{P}(\texttt{obs}) \in dom[\mathcal{A}]$. A policy $\mathbb{P}$ is ***sound on*** $\texttt{obs} \in \mathcal{S}^*$ ***against*** $\texttt{asg} \in \mathcal{S}$ if $\mathbb{P}(\texttt{obs})$ masks asg; soundness concisely states that the resulting value for each $x_i \in \mathcal{A}$ is either $*$, or correct according to the nature-assigned value of x_i in asg. By extension: A policy $\mathbb{P}$ is ***complete on*** a given fraction / sublist of $\mathcal{S}^*$ if it is complete on each obs in that fraction / sublist of $\mathcal{S}^*$. A policy $\mathbb{P}$ is ***sound on*** a given fraction / sublist of $\mathcal{S}^*$ ***against*** $\mathcal{S}$ if it is sound on each obs in that fraction / sublist of $\mathcal{S}^*$ against the asg in $\mathcal{S}$ that corresponds to obs.

We shall employ the notions of soundness and completeness mutatis mutandis on individual predictors as well, by thinking of them as policies that include a single predictor.

Before continuing with the technical analysis of different policies in subsequent sections, certain remarks are in order.

The first remark relates to soundness and completeness and how these notions relate to their typical use in KRR. In addition to the points we have already made in the first section, we note also the quantitative nature of the notions. We will not refer to a policy as being simply sound or complete, but we will quantify the fraction / percentage of the given inputs on which soundness and completeness are achieved.

The second remark relates to how rules are applied. Some form of sequentiality when applying rules seems to resemble proof procedures in KRR research more closely than the parallel application of rules. After all, one could argue, the latter approach seems to unnecessarily waste the opportunity to get more complete inferences by having rules interact. On the other hand, the parallel application of rules seems to be closer to what a lot of Machine Learning research has traditionally done when dealing with multiple predictors; certain exceptions of this trend are discussed in the last section.

In the sequel we shall identify the reason behind the above discrepancy, which comes down to the fact that applying *learned* (and hence approximately correct) rules naively in some sequential fashion, can reduce soundness. On the other hand, we shall show that an appropriate application of rules in some non-parallel fashion provably improves their combined soundness and completeness. The main question, then, becomes that of finding how to apply rules so that to exploit this improvement, without sacrificing their soundness.

The third remark relates to the qualification of rules. When each rule in a policy is applied, its prediction is used to update information on an attribute only if information on the attribute's value is missing. If an attribute's value is observed, then the prediction will yield to this explicit observation, and be exogenously qualified. It can be easily observed

[2]Predictors utilizing other representations for the rules according to which they make predictions are also possible. In particular, utilizing boolean attributes and a logic-based representation gives rise to rules closer to what a lot of KRR research employs.

that the same treatment accounts for the endogenous qualification of rules as well. If rules that are applied first do not abstain and do make a prediction, then those predictions will complete missing information, which will subsequently prevent the predictions of rules that are applied later from being recorded; in effect, the former rules override the latter rules.

Our investigation herein seeks to learn these rules, and the ordering in which they are applied. Since ordering captures rule qualification, the knowledge bases that we end up learning are ultimately non-classical (i.e., non-monotonic), even though each individual rule is restricted to being classical.

Benefits of Chaining Predictors

Consider a policy $\mathbb{P}$ that prescribes a sequence of d sets of predictors. We shall call d the ***depth of*** $\mathbb{P}$, and the sets the ***layers of*** $\mathbb{P}$, indexed by their order in the sequence. A policy $\mathbb{P}$ is ***flat*** if its depth d is 1, and is ***chained*** if $d \geq 2$; i.e., when at least one predictor in $\mathbb{P}$ is applied *after* some other predictor in $\mathbb{P}$. An ***empty*** policy $\mathbb{P}$ has depth $d = 0$. A policy $\mathbb{P}_1$ is a ***reordering of*** a policy $\mathbb{P}_2$ if the multi-set of the union of the layers of $\mathbb{P}_1$ matches that of $\mathbb{P}_2$; i.e., the policies share the predictors that are being used, but may differ on their depths and on how predictors are assigned to the policy layers.

We proceed to consider whether having chained policies is ever a good idea. In other words, we seek to understand if a form of sequential application of rules, as done in proof procedures in KRR, is useful when it comes to *learned* rules?

The distinction between *learned* and *programmed* rules is critical and should be emphasized. It is trivial to demonstrate the benefits (in terms of completeness) of chaining in knowledge bases with programmed rules: given the rules "if you have fever then you are sick" and "if you are sick then visit a doctor", the inference "visit a doctor" cannot be drawn from the observation "you have fever" without using both rules, since no single rule can bridge the gap between the available observation and the sought inference. However, when rules are learned, there is nothing preventing the learning process from inducing a single rule that bridges this gap (e.g., "if you have fever then visit a doctor"). In theory, then, the learning process itself could render the need for chaining superfluous.

We have revealed in the preceding section that, despite the concern above, chaining remains provably beneficial for knowledge bases with *learned* rules. To establish this claim, one must consider not only the completeness that a knowledge base can achieve (as in the case of programmed rules), but also the soundness. Indeed, the ability of a learning process to *construct* rules (in contrast to being given *prescribed* rules by a process of programming) may trivially lead to perfect completeness by constructing a constant rule for each attribute. It is only by including soundness in the evaluation of a knowledge base that we can demonstrate the benefits of chaining. This evaluation metric of combined soundness and completeness is captured by the following definition.

Definition 1 (Chaining Collapsibility). ***Chaining collapses on $\mathcal{S}^*$ against $\mathcal{S}$*** *if: for every policy $\mathbb{P}$ that is unsound on an ε fraction of $\mathcal{S}^*$ against $\mathcal{S}$, and incomplete on an ω fraction of $\mathcal{S}^*$, there exists a flat policy $\mathbb{P}'$ of predictors that is unsound on an ε' fraction of $\mathcal{S}^*$ against $\mathcal{S}$, and incomplete on an ω' fraction of $\mathcal{S}^*$ such that $\varepsilon' + \omega' \leq \varepsilon + \omega$.*

The sum of the soundness and completeness of a policy $\mathbb{P}$ on $\mathcal{S}^*$ against $\mathcal{S}$ will play an important role in our analysis, and shall be called the ***performance*** of $\mathbb{P}$ ***on $\mathcal{S}^*$ against $\mathcal{S}$***. Chaining is provably beneficial if a situation can be demonstrated where the performance of a particular policy $\mathbb{P}$ cannot be matched by *any* flat policy (not necessarily a reordering of $\mathbb{P}$). We shall demonstrate such a situation next.

Fix an assignment $\texttt{obs} \in \mathcal{S}^*$, a predictor P_t for attribute $x_t \in \mathcal{A}$, and an attribute $x_i \in \mathcal{A}$ that is masked in $\texttt{obs}$. Consider changing the $*$ value of x_i in $\texttt{obs}$ to a value in $dom[i]$. If at least one such value change causes the prediction of P_t to change, then x_i is ***relevant for*** P_t ***w.r.t.*** $\texttt{obs}$. If at least two such value changes cause the prediction of P_t to change in two different ways, then x_i is ***critical for*** P_t ***w.r.t.*** $\texttt{obs}$.

By way of illustration, consider a predictor P_3 that predicts according to the rule "$x_3 := x_1 \cdot x_2$". Given the assignment $\texttt{obs} = \langle *, 2, -1 \rangle$, the prediction of P_3 is $*$. Changing the value of x_1 in $\texttt{obs}$ to any real number causes the prediction of P_3 to change from $*$ to the double of that number. Thus, attribute x_1 is critical (and relevant) for P_3 w.r.t. $\texttt{obs}$.

Lemma 1 (Uniqueness of Critical Attributes). *Consider a predictor P_t for attribute $x_t \in \mathcal{A}$, and an assignment $\texttt{obs}$. Then: If attribute $x_i \in \mathcal{A}$ is critical for P_t w.r.t. $\texttt{obs}$, then no other attribute $x_j \in \mathcal{A}$ is relevant for P_t w.r.t. $\texttt{obs}$.*

Proof. Consider the assignment $\texttt{obs}_{u,v}$ obtained from $\texttt{obs}$ after replacing $\texttt{obs}[i]$, $\texttt{obs}[j]$ respectively with u, v. If x_i is critical for P_t w.r.t. $\texttt{obs}$, then $P_t(\texttt{obs}_{c,*})$, $P_t(\texttt{obs}_{d,*})$, $*$ are distinct for some $c, d \in dom[i]$, $c \neq d$. Assume, by way of contradiction, that x_j is relevant for P_t w.r.t. $\texttt{obs}$. Then $P_t(\texttt{obs}_{*,a}) \neq *$ for some $a \in dom[j]$. Since P_t is classical, $P_t(\texttt{obs}_{c,a}) = P_t(\texttt{obs}_{*,a}) = P_t(\texttt{obs}_{d,a}) = P_t(\texttt{obs}_{d,*}) \neq P_t(\texttt{obs}_{c,*}) = P_t(\texttt{obs}_{c,a})$; a contradiction, as needed. □

Hence, no classical predictor can have its predictions be unilaterally affected / determined by the values of more than one attribute. Although this limitation does not necessarily burden non-classical predictors, Lemma 1 bears a significant weight due to the widespread use of classical predictors in Machine Learning research. To prove that chaining does not collapse, it suffices to demonstrate any particular policy $\mathbb{P}$ that chains its predictors in a manner that effectively simulates non-classical predictors. We present such a policy next.

Consider a list $\mathcal{S}$ of complete assignments, where unbeknownst to a base algorithm, attributes x_1 and x_2 are always assigned values equal to each other, and x_3 is assigned their product. In all assignments in $\mathcal{S}^*$ available to the base algorithm, both x_3 and exactly one of x_1, x_2 are masked. Given *our* knowledge (of $\mathcal{S}$) that x_1 and x_2 share common values, we may expect the base algorithm to return a predictor P_0 for x_3 that always makes correct predictions. Using this predictor P_0, along with a predictor P_1 that predicts according to the rule "$x_1 := x_2$", and a predictor P_2 that predicts according to the rule "$x_2 := x_1$", we would be able to construct a flat policy that would be perfectly sound and complete. Alas, this scenario is not possible, as a consequence of Lemma 1.

Theorem 2 (The Benefits of Chaining). *There exist lists $\mathcal{S}^*$ and $\mathcal{S}$ such that chaining does not collapse on $\mathcal{S}^*$ against $\mathcal{S}$.*

Proof. Consider lists $\mathcal{S}^*$ and $\mathcal{S}$ as in the paragraph above.

Let policy $\mathbb{P}$ have predictors P_1 and P_2 from above in its first layer, and a predictor P_3 according to the rule "$x_3 := x_1 \cdot x_2$" in its second layer. $\mathbb{P}$ has optimal performance.

Now further specify list $\mathcal{S}^*$ and list $\mathcal{S}$ so that: At least two complete assignments $\mathtt{asg}_1, \mathtt{asg}_2 \in S$ are such that $\mathtt{asg}_1[1] \neq \mathtt{asg}_2[1]$; i.e., two different values in the domain $dom[1]$ of x_1 appear in complete assignments in $\mathcal{S}$. The assignments $\mathtt{obs}_1, \mathtt{obs}_2 \in \mathcal{S}^*$ that correspond to $\mathtt{asg}_1, \mathtt{asg}_2 \in S$ mask attribute x_2. At least one assignment $\mathtt{obs}_3 \in \mathcal{S}^*$ masks attribute x_1, and for each attribute $x_i \in \mathcal{A} \setminus \{x_1, x_2, x_3\}$, it is the case that $\mathtt{obs}_3[i] = \mathtt{obs}_1[i]$; i.e., the "context" defined by the other attributes is the same, although the values of the attributes of interest may differ.

Assume that some flat policy $\mathbb{P}'$ has optimal performance. Then, $\mathbb{P}'$ includes a predictor P_3 for x_3 that always makes a correct definite prediction. Since its predictions on $\mathtt{obs}_1$ and $\mathtt{obs}_2$ differ, then x_1 is critical for P_3 w.r.t. the assignment $\mathtt{obs}_*$ that is obtained from either $\mathtt{obs}_1$ or $\mathtt{obs}_2$ by replacing the value of x_1 with $*$. Necessarily, the prediction of P_3 when applied on $\mathtt{obs}_*$ is $*$. Assignment $\mathtt{obs}_*$ can be obtained also from $\mathtt{obs}_3$ by replacing the value of x_2 with $*$. Since predictor P_3 makes a definite prediction when applied on $\mathtt{obs}_3$, then x_2 is relevant for P_3 w.r.t. the assignment $\mathtt{obs}_*$. A contradiction by Lemma 1, as needed. □

Collectively, the predictors in policy $\mathbb{P}$ given in the proof of Theorem 2 implement a non-classical rule: "$x_3 := (x_1)^2$ if x_2 *is not known*, or $(x_2)^2$ if x_1 *is not known*". The rule's non-classical nature is evidenced by observing that if more information becomes known, the rule may retract a definite prediction it had previously made. As a matter of fact, it can be shown that any "epistemic" rule that makes predictions based not only on what holds, but also on what is *known*, can be simulated by appropriately choosing and chaining classical predictors as part of a certain policy (Michael 2008).

Can the Benefits be Realized?

Chaining predictors in some *appropriate* manner can provably enhance the combined soundness and completeness that is achieved. Are there concrete efficient algorithms that identify such a chaining? We consider two approaches below.

First Learn, Then Chain

In the first approach proceed as follows: First, call the base algorithm on list $\mathcal{S}^*$ to get one predictor P_i for each attribute $x_i \in \mathcal{A}$. Then, order the predictors to obtain a policy $\mathbb{P}$.

Such an approach captures what is implicitly assumed in certain research in KRR: that some process effectively produces the rules in the knowledge base, after which the rules can be applied in any order to draw inferences while safely ignoring the process through which the rules were produced.

Is the approach of first learning and then chaining (FLTC) the predictors ever useful? To answer this question, we need to examine more precisely the guarantees that the base algorithm is assumed to offer on the predictors it returns.

Scenario 1: Guarantees on $\mathcal{S}$

For our first scenario, assume that each P_i is guaranteed to be unsound on only an ε_i fraction of $\mathcal{S}$ against $\mathcal{S}$. Note that the predictive guarantees are w.r.t. the *complete assignments* $\mathcal{S}$, which capture the underlying reality associated with $\mathcal{S}^*$. Thus, soundness is assumed even when the predictors are applied on complete assignments in $\mathcal{S}$ (where, in particular, abstentions are disallowed); a more stringent requirement than the general case where soundness is assumed on $\mathcal{S}^*$ against $\mathcal{S}$. The problem of how such guarantees can be effectively obtained is orthogonal, and is discussed later in this work. Let $\varepsilon \triangleq \sum_{i=1}^{n} \varepsilon_i < 1$. We study the performance of $\mathbb{P}$ on $\mathcal{S}^*$.

Theorem 3 (Soundness in FLTC Setting). *Consider the predictors as described above. Any policy $\mathbb{P}$ that uses these predictors is unsound on at most an ε fraction of $\mathcal{S}^*$ against $\mathcal{S}$.*

Proof. By a union bound, since predictors are classical. □

Theorem 4 (Completeness in FLTC Setting). *Consider the predictors as described above. Let policy $\mathbb{P}$ have n layers, with each predictor appearing in every layer. Let policy $\mathbb{P}'$ be a reordering of $\mathbb{P}$. Then: If $\mathbb{P}$ and $\mathbb{P}'$ are, respectively, incomplete on an ω and ω' fraction of $\mathcal{S}^*$, then $\omega \leq \omega'$.*

Proof. The policy $\mathbb{P} + \mathbb{P}'$ that concatenates the layers of $\mathbb{P}'$ after all the layers of $\mathbb{P}$, is not less complete than $\mathbb{P}'$. It suffices to establish that $\mathbb{P}$ is not less complete than $\mathbb{P} + \mathbb{P}'$. Consider an assignment $\mathtt{obs} \in \mathcal{S}^*$. If $\mathbb{P}(\mathtt{obs}) \in dom[\mathcal{A}]$, then the claim follows. Otherwise, and since $\mathbb{P}$ has n layers, there must exist some layer in $\mathbb{P}$ that does not complete the value of any new attributes (that were not already completed by earlier layers). Since that layer is a superset of each subsequent layer in $\mathbb{P}+\mathbb{P}'$, any subsequent layer will also fail to complete the value of any new attributes. Thus, $\mathbb{P}$ and $\mathbb{P}+\mathbb{P}'$ will abstain on the same attributes. The claim follows. □

Theorems 3–4 offer a first efficient algorithm for chaining predictors, with bounded unsoundness, and optimal, in a defined sense, completeness. However, the assumed guarantees on the predictors might not always be possible to have.

Before moving on, we note that this scenario accounts for what is assumed in some KRR work: each rule truly captures the structure of the environment of interest (i.e., unsound on an $\varepsilon_i = 0$ fraction of $\mathcal{S}$), and so chaining rules in *any* order is justified without cost on the overall soundness (i.e., $\varepsilon = 0$). Further, applying each predictor (in a given knowledge base) as often as possible is optimal in terms of completeness.

Scenario 2: Guarantees on $\mathcal{S}^*$

For our second scenario, assume that each P_i is guaranteed to be unsound on only an ε_i fraction of $\mathcal{S}^*$ against $\mathcal{S}$. The predictive guarantees are w.r.t. the *observed assignments* $\mathcal{S}^*$. Let $\varepsilon \triangleq \sum_{i=1}^{n} \varepsilon_i < 1$. We study the performance of $\mathbb{P}$ on $\mathcal{S}^*$.

Although Theorem 4 holds unchanged, a naive ordering approach compromises Theorem 3 and the soundness of $\mathbb{P}$.

Theorem 5 (Unsoundness in FLTC Setting). *Consider the predictors as described above. Assume that at least one predictor is not constant on all complete assignments. Then: There exists a policy $\mathbb{P}$ (in fact, there exist exponentially in n*

many such policies) that orders these predictors that is unsound on all of $\mathcal{S}^*$ against $\mathcal{S}$, even though some reordering of $\mathbb{P}$ is unsound on at most an ε fraction of $\mathcal{S}^*$ against $\mathcal{S}$.

Proof. Let P_t be a non-constant predictor for $x_t \in \mathcal{A}$. Order $dom[\mathcal{A}]$ as a Gray Code (Knuth 2011). Choose two complete assignments $\mathtt{asg}_1, \mathtt{asg}_2 \in dom[\mathcal{A}]$ that are consecutive in the Gray Code and such that $P_t(\mathtt{asg}_1) \neq P_t(\mathtt{asg}_2)$, and let x_j be the attribute on which the assignments differ. Since P_t does not base its predictions on x_t, then $x_j \neq x_t$, and $\mathtt{asg}_1[t] = \mathtt{asg}_2[t]$. Without loss of generality, let $\mathtt{asg}_1$ be one among the two chosen complete assignments such that $P_t(\mathtt{asg}_1) \neq \mathtt{asg}_1[t]$. Construct the observed assignment $\mathtt{obs}_1$ from $\mathtt{asg}_1$ by replacing $\mathtt{asg}_1[j]$ and $\mathtt{asg}_1[t]$ with $*$.

Let $\mathcal{S}$ include only $\mathtt{asg}_1$, and let $\mathcal{S}^*$ include only $\mathtt{obs}_1$. Then, $P_t(\mathtt{obs}_1) = *$. Choose a predictor P_j for $x_j \in \mathcal{A}$ such that $P_j(\mathtt{obs}_1) = \mathtt{asg}_1[j]$. Choose a predictor P_i for each $x_i \in \mathcal{A} \setminus \{x_t, x_j\}$ such that $P_i(\mathtt{obs}_1) \in \{\mathtt{asg}_1[i], *\}$. It is consistent for all these predictors to have been returned by the base algorithm as assumed in the current scenario. A flat policy of the predictors is sound on all of $\mathcal{S}^*$ against $\mathcal{S}$.

Consider a policy $\mathbb{P}$ out of the exponentially in n many ones that order P_j before P_t. Since $P_j(\mathtt{obs}_1) = \mathtt{asg}_1[j]$, then policy $\mathbb{P}$ will apply P_t on an assignment obtained from $\mathtt{obs}_1$ by replacing $\mathtt{obs}_1[j]$ with $\mathtt{asg}_1[j]$. On this assignment, P_t will predict a value in the domain $dom[t]$ of x_t that differs from $\mathtt{asg}_1[t]$. Therefore, $\mathbb{P}(\mathtt{obs}_1)$ will not mask $\mathtt{asg}_1$, and hence $\mathbb{P}$ is unsound on all of $\mathcal{S}^*$ against $\mathcal{S}$. The flat policy given earlier, which is a reordering of $\mathbb{P}$, is trivially unsound on at most an ε fraction of $\mathcal{S}^*$ against $\mathcal{S}$. □

The severe loss of soundness in the second scenario is not due to the cumulative effect of the unsoundness of individual predictors; that happens in the first scenario. Rather, it is due to chaining *forcing* predictors to make predictions when they would have otherwise abstained. More specifically: The guarantees of each predictor P_i are w.r.t. $\mathcal{S}^*$. Yet, P_i is applied to predict on assignments that are not in $\mathcal{S}^*$, but assignments that have incorporated the predictions of predictors in earlier layers in $\mathbb{P}$. These more complete assignments are the ones that cause P_i to enhance its completeness by abstaining less, at the cost, however, of sacrificing its soundness.

SLAP-ing with Predictors

The naive FLTC approach essentially trades off soundness for completeness. To boost completeness without sacrificing soundness, the predictors need to be applied on assignments w.r.t. which they (will be made to) have guarantees.

The second approach, then, recognizes that learning and prediction cannot proceed independently, and that one needs to simultaneously learn and predict (SLAP) with predictors.

Algorithm 1, henceforth denoted by $\mathcal{L}_{\text{SLAP}}$, achieves the need to SLAP by interleaving learning and prediction to iteratively build a chained policy $\mathbb{P}$. It is given a list $\mathcal{S}^*$ of size m, a non-negative integer d indicating the desired depth of $\mathbb{P}$, and a positive value ε indicating the desired upper bound on the unsoundness of $\mathbb{P}$ on $\mathcal{S}^*$ against an *unspecified* list $\mathcal{S}$.

One should not be surprised with the requirement that algorithm $\mathcal{L}_{\text{SLAP}}$ achieves soundness guarantees on $\mathcal{S}^*$ against an unspecified, and unavailable, list $\mathcal{S}$. Meeting this seemingly inordinate requirement is critically facilitated by the oracle access that algorithm $\mathcal{L}_{\text{SLAP}}$ has to a base algorithm $\mathcal{L}_{\text{base}}$. As in the second scenario of the preceding section, we shall assume that $\mathcal{L}_{\text{base}}$ is able to provide guarantees for individual predictors on $\mathcal{S}^*$ against $\mathcal{S}$. Furthermore, we shall assume that the base algorithm $\mathcal{L}_{\text{base}}$ can be made to reduce the unsoundness of its returned predictors to any specified positive value ε_k, assuming $\mathcal{L}_{\text{base}}$ is given access to list $\mathcal{S}^*$ and is allowed time polynomial in $1/\varepsilon_k$.[3] So, as long as we do not ask for "unreasonably" small (as a function of n and m) values for ε_k, the calls to base algorithm $\mathcal{L}_{\text{base}}$ will not burden the overall computational efficiency of algorithm $\mathcal{L}_{\text{SLAP}}$.

Algorithm 1 Simultaneous Learning and Prediction

input: depth d, n attributes $\mathcal{A}$, soundness error ε, list $\mathcal{S}^*$ of m assignments, base algorithm $\mathcal{L}_{\text{base}}$ as given in text.
Set $\mathcal{S}^*_0$ to equal $\mathcal{S}^*$.
for every $k = 1, 2, \ldots, d$ **do**
  for every attribute $x_i \in \mathcal{A}$ **do**
    Call the base algorithm $\mathcal{L}_{\text{base}}$ on input x_i, $\mathcal{A}$, ε_k, δ_k, and $\mathcal{S}^*_{k-1}$, and let P_i be the predictor that it returns.
  end for
  Let $\mathbb{P}_k$ be a flat policy of the predictors $\{P_i \mid x_i \in \mathcal{A}\}$.
  Let $\mathcal{S}^*_k$ be the list of assignments given by $\mathbb{P}_k(\mathcal{S}^*_{k-1})$.
end for
Set $\mathbb{P}$ so that its j-th layer includes the predictors in $\mathbb{P}_j$.
output: Return the policy $\mathbb{P}$, and terminate.

Given oracle access to base algorithm $\mathcal{L}_{\text{base}}$ with the characteristics described above, algorithm $\mathcal{L}_{\text{SLAP}}$ calls $\mathcal{L}_{\text{base}}$ on $\mathcal{S}^*$ to obtain a predictor for each attribute, and applies these predictors in parallel to update the assignments in $\mathcal{S}^*$. It repeats the process, by calling algorithm $\mathcal{L}_{\text{base}}$ on the new assignments to obtain a new predictor for each attribute, and applying the new predictors in parallel to further update the assignments, and so on for d iterations. Algorithm $\mathcal{L}_{\text{SLAP}}$ keeps track of all predictors $\mathbb{P}_j$ obtained and applied in each iteration, and constructs the policy $\mathbb{P}$ by assigning in its j-th layer the predictors that were used during the j-th iteration.

We have purposefully left unspecified what values would be considered "reasonable" for ε_k, both in the discussion above, and in Algorithm 1. We will examine two scenarios.

For our first scenario, we assume that accessing all assignments in $\mathcal{S}^*$ is conceptually meaningful and computationally plausible; e.g., when seeking to recover missing information in all records of a database. Since time linear in n and m is already needed just for reading all of $\mathcal{S}^*$, it seems reasonable to allow ε_k to be an inverse linear function of n and m.

Observe, then, that setting $\varepsilon_k = 1/2m$ would ensure that each predictor returned by $\mathcal{L}_{\text{base}}$ during the k-th iteration of $\mathcal{L}_{\text{SLAP}}$ would be perfectly sound on $\mathcal{S}^*_{k-1}$, and so would policy $\mathbb{P}_k$. As a result, list $\mathcal{S}^*_k$ would be noiseless, and by induction the returned policy $\mathbb{P}$ would be perfectly sound on $\mathcal{S}^*$.

[3] We have already remarked on the black-box view of base algorithms taken herein, and the emphasis, instead, on their appropriate use. Nonetheless, we remind the reader that such base algorithms can be constructed under typical assumptions (Michael 2010).

For our second scenario, we assume that assignments in $\mathcal{S}^*$ can only be sampled or otherwise only partly accessed; e.g., when seeking to learn predictors from a sample of data with the aim to apply them on future data. Time linear in n is still needed just for reading a single assignment in $\mathcal{S}^*$, but no dependence of the running time on m can be generally assumed.[4] Thus, we shall not allow ε_k to depend on m.

Exactly due to the restricted access assumed on $\mathcal{S}^*$, $\mathcal{L}_{\text{base}}$ may occasionally fail to produce a predictor with the sought unsoundness ε_k. Analogous to our treatment for ε_k, we shall assume that the base algorithm $\mathcal{L}_{\text{base}}$ can be made to reduce the failure probability to any specified positive value δ_k, assuming $\mathcal{L}_{\text{base}}$ is allowed time polynomial in $1/\delta_k$. For reasons same as for ε_k, we shall not allow δ_k to depend on m.

Given a base algorithm $\mathcal{L}_{\text{base}}$ as discussed above, we can establish that SLAP preserves soundness despite chaining.

Theorem 6 (Soundness in SLAP Setting). *For any inputs and value δ, there exist values ε_k and δ_k such that the policy $\mathbb{P}$ returned by algorithm $\mathcal{L}_{SLAP}$ is, except with probability δ, unsound on at most an ε fraction of $\mathcal{S}^*$ against $\mathcal{S}$. For fixed d, algorithm $\mathcal{L}_{SLAP}$ runs in time polynomial in n, $1/\varepsilon$, $1/\delta$.*

Proof. Let $q(n, 1/\varepsilon_k, 1/\delta_k)$ be a polynomial that determines the running time of the base algorithm $\mathcal{L}_{\text{base}}$ on input $\mathcal{A}$, ε_k, δ_k. It is easy to show that algorithm $\mathcal{L}_{\text{SLAP}}$ runs in time

$$O\left(d \cdot poly(n) \cdot \sum_{k=1}^{d} q(n, 1/\varepsilon_k, 1/\delta_k)\right).$$

Set $\delta_k := \delta/2d$. Then, there is a polynomial $p(n, 1/\delta, d)$ and a constant c such that setting $\varepsilon_k := (\varepsilon/p(n, 1/\delta, d))^{c^{d-k}}$ establishes that the running time of algorithm $\mathcal{L}_{\text{SLAP}}$ is polynomial in n, $1/\varepsilon$, $1/\delta$ for fixed d. Furthermore, the choices of δ_k and ε_k guarantee that, except with probability δ, each call of $\mathcal{L}_{\text{base}}$ will return a sufficiently sound predictor such that whenever it is applied in the context of algorithm $\mathcal{L}_{\text{SLAP}}$, its predictions will be sound (i.e., the improbable event of making unsound predictions will never occur). Overall then, except with probability δ, the above effectively reproduces the noiseless intermediate predictions of the first scenario. □

Theorem 6 is to the SLAP setting what Theorem 3 is to the FLTC setting. Analogously to Theorem 4 for the FLTC setting, one can show that the completeness of algorithm $\mathcal{L}_{\text{SLAP}}$ is optimal for certain natural settings. The settings in question are those where attributes (i.e., notions whose definitions one may wish to learn) are defined in terms of one another in a hierarchical fashion, so that there are no directed cycles in the definitions. We formalize such settings below.

A list $\mathcal{S}$ of complete assignments is ***d-stratified*** if for a partitioning $\mathcal{A}_1, \ldots, \mathcal{A}_d$ of $\mathcal{A}$, and for every $x_i \in \mathcal{A}_j$, there is a predictor P_i sound on all of $\mathcal{S}$ against $\mathcal{S}$ and complete on all of $\mathcal{S}$, that does not base its predictions on $A_j \cup \ldots \cup \mathcal{A}_d$.

We shall assume that we have access to base algorithms, which we shall call ***focused***, that return predictors that do not base their predictions on attributes that are clearly irrelevant due to the stratification of $\mathcal{A}$. Such base algorithms can be constructed either by having direct access to the partitioning if it is known, or by exploiting domain-specific knowledge, attribute-efficient learning, or other relevant techniques.

By way of illustration, let $d = 4$, $x_5 \in \mathcal{A}_2$ and $x_3 \in \mathcal{A}_4$, and $\texttt{obs} = \langle 8, -3, *, 7, *, -2\rangle$. Any predictor P_5 for x_5 returned by a focused base algorithm makes a definite prediction, since none of the attributes $\mathcal{A} \setminus (A_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4)$ is masked. On the other hand, a definite prediction by a predictor P_3 for x_3 might not be possible since the value of x_3 may depend on the value of x_5, which is masked in obs. If we consider, however, a policy $\mathbb{P}$ that chains P_3 after P_5, the prediction of the latter predictor necessarily completes x_5's value before the prediction of the former predictor on x_3, *ensuring* that a definite prediction is made by P_3 for x_3.

We show that algorithm $\mathcal{L}_{\text{SLAP}}$ guarantees predictors with optimal completeness in such stratified settings.

Theorem 7 (Completeness in SLAP Setting). *Consider any integer value d, a d-stratified list $\mathcal{S}$ of complete assignments, and an associated list $\mathcal{S}^*$. Assume that algorithm $\mathcal{L}_{SLAP}$ has oracle access to a focused base algorithm $\mathcal{L}_{base}$ and returns a policy $\mathbb{P}$ of depth d. Then: $\mathbb{P}$ is complete on all of $\mathcal{S}^*$.*

Proof. The claim follows by straightforward induction. □

Theorems 6–7 offer a second algorithm for chaining predictors, with bounded unsoundness, and optimal completeness when in stratified settings. The algorithm is efficient for constant-depth policies. Unlike the first algorithm of Theorems 3–4, the second algorithm is more realistic in assuming a base algorithm with guarantees w.r.t. $\mathcal{S}^*$, instead of $\mathcal{S}$.

Although it remains open whether the efficiency guarantees of Theorem 6 can be extended to hold for super-constant values of d, several natural modifications to algorithm $\mathcal{L}_{\text{SLAP}}$ fail to resolve this question: *(i)* Employing base algorithms that produce predictors by expending sub-linear resources in the inverse error parameter $1/\varepsilon_k$ are excluded (Ehrenfeucht et al. 1989). *(ii)* Base algorithms that *identify* their inaccurate predictions (Rivest and Sloan 1994) employ a strategy that does not scale up to expressive predictors, and fails to work when instead of complete assignments one has access to arbitrary assignments. *(iii)* Training all predictors on a common set of assignments (Valiant 2000) corresponds, effectively, to the FLTC scenario and not to the SLAP scenario considered herein. *(iv)* Appealing to noise-resilient base algorithms also fails,[5] as the unsound predictions of the predictors in earlier layers of $\mathbb{P}$ can be shown (Michael 2008) to correspond to malicious noise for the predictors in later layers of $\mathbb{P}$. Such malicious noise is known to be especially intolerable by base algorithms (Kearns and Li 1993).

Overall, then, evidence suggests that interleaving learning and prediction is a solid approach when seeking to improve

[4] Under this second scenario, algorithm $\mathcal{L}_{\text{SLAP}}$ can no longer be assumed to explicitly compute the list $\mathcal{S}^*_k := \mathbb{P}_k(\mathcal{S}^*_{k-1})$ during the last step of the k-th iteration. This can be easily dealt with, by lazily computing parts of $\mathcal{S}^*_k$ if and when needed for subsequent steps.

[5] We are not suggesting that noise-resilient (or otherwise robust) base algorithms be avoided altogether, only that doing so does not improve asymptotically the achieved chaining depth. One can investigate how the noise-resilience of base algorithms relates to that of $\mathcal{L}_{\text{SLAP}}$ (as done for soundness and completeness). We have not considered noise in this work to avoid obscuring the main points.

completeness without sacrificing soundness. Even in scenarios where d-stratification is not a priori guaranteed, calling $\mathcal{L}_{\text{SLAP}}$ with increasing values for d allows the effective trade-off of time for possibly better, but never worse, performance.

Explicit Completeness

We have established in certain settings *explicit* bounds on the unsoundness of our considered policies, but only a type of optimal, not always specific, degree of completeness. Why not expect algorithms to return policies with *explicit* bounds on their degree of completeness? The answer is two-fold:

First, completeness depends externally on the masking process used to produce list $\mathcal{S}^*$ from list $\mathcal{S}$. In the interest of generality — and this happens to be the case in certain natural settings — no assumptions on how this masking process operates have been (or can be) made. The more information is missing in list $\mathcal{S}^*$ the more a policy is forced to abstain, as long as one insists, as we have done herein, that soundness should not be severely compromised to boost completeness.

Second, even if one can construct efficiently a policy that achieves either perfect soundness or perfect completeness, and even if sufficient information is available in list $\mathcal{S}^*$ for some policy to make perfectly sound and perfectly complete predictions, it is still not possible, in general, to achieve efficiency, soundness, and completeness together. We establish next a barrier to efficiently constructing a policy that simultaneously has soundness that is slightly better than a policy that predicts with constant predictors, and has completeness that is slightly better than a policy that makes no predictions.

Definition 2 (Properties of Policy-Learning Algorithms). *In what follows, **parameters** are: an integer $n > 1$, a list $\mathcal{S}$ of complete assignments over a set $\mathcal{A}$ of n attributes, and a list $\mathcal{S}^*$ of assignments. A policy-learning algorithm $\mathcal{L}$:*

*(i) is **computationally efficient** if there is a polynomial $r(\cdot,\cdot)$ such that for every choice of the parameters, $\mathcal{L}$ takes $\mathcal{A}$ and $\mathcal{S}^*$ as input, and returns a policy $\mathbb{P}$ in time $r(n, |\mathcal{S}^*|)$;*

*(ii) achieves **non-trivial soundness** if there is a polynomial $s(\cdot)$ such that for every choice of the parameters, $\mathcal{L}$ takes $\mathcal{A}$ and $\mathcal{S}^*$ as input, and returns a policy $\mathbb{P}$ that is sound on an $1/2 + 1/s(n)$ fraction of $\mathcal{S}^*$ against $\mathcal{S}$;*

*(iii) achieves **non-trivial completeness** if there is a polynomial $c(\cdot)$ such that for every choice of the parameters, $\mathcal{L}$ takes $\mathcal{A}$ and $\mathcal{S}^*$ as input, and returns a policy $\mathbb{P}$ that is complete on an $1/c(n)$ fraction of $\mathcal{S}^*$.*

Given these desirable properties for a policy-learning algorithm, the following observations are immediate: The trivial algorithm that returns the empty policy is computationally efficient and perfectly sound. The trivial algorithm that returns a policy comprising one constant predictor for each attribute is computationally efficient and perfectly complete.

Achieving perfect soundness and perfect completeness is easily shown to be impossible if information is hidden adversarially. Assume, therefore, that we are in a non-adversarial setting, where this third combination of two desirable properties is achievable. It it still the case that achieving all three properties simultaneously is impossible, even if only non-trivial soundness and non-trivial completeness is required.

Theorem 8 (Barriers to Explicit Completeness). *Under typical cryptographic assumptions (Kearns and Vazirani 1994), no algorithm that returns policies is computationally efficient and achieves non-trivial soundness and non-trivial completeness. This is so even if there is a perfectly sound and perfectly complete policy, and such policy can be approximated by a (possibly inefficient) PAC learning algorithm.*

Proof. Assume that the claim does not hold for an algorithm $\mathcal{L}$, and consider the polynomials $r(\cdot,\cdot)$, $s(\cdot)$, $c(\cdot)$ from Definition 2. Consider any concept class known not to be weakly PAC-learnable under standard cryptographic assumptions (Kearns and Vazirani 1994). For any function in the class, and every distribution $\mathcal{D}$ over examples, construct a list $\mathcal{S}$ of $m \cdot c(n) \cdot (s(n)+1)$ complete assignments by sampling examples from $\mathcal{D}$ and assigning to x_1 their label according to f. Construct the list $\mathcal{S}^*$ by masking attribute x_1 in each assignment except with probability $1/(c(n) \cdot (s(n)+1))$.

The policy that includes only predictor f for x_1 is sound on all of $\mathcal{S}^*$ against $\mathcal{S}$, and complete on all of $\mathcal{S}^*$. In addition, choosing m to be sufficiently large (but still polynomial in n and the PAC error parameters $1/\varepsilon$ and $1/\delta$), PAC learning can be achieved by finding any function in the concept class that is consistent with assignments in $\mathcal{S}^*$ that are complete.

Running algorithm $\mathcal{L}$ on $\mathcal{S}^*$ for time $r(n, |\mathcal{S}^*|)$ will return a policy $\mathbb{P}$ that is non-trivially complete on $\mathcal{S}^*$, and, therefore, complete on $|\mathcal{S}^*|/c(n)$ assignments, which is more than those already complete in $\mathcal{S}^*$. Then, $\mathbb{P}$ includes a predictor g for x_1 that makes predictions on every assignment in $\mathcal{S}^*$. Policy $\mathbb{P}$ is also non-trivially sound on $\mathcal{S}^*$ against $\mathcal{S}$, and, therefore, sound on at least $(1/2 + 1/s(n)) \cdot |\mathcal{S}^*|$ assignments against $\mathcal{S}$. Ignoring the assignments that are complete in $\mathcal{S}^*$ (and on which the prediction of g is qualified), g predicts correctly with probability $((1/2 + 1/s(n)) \cdot |\mathcal{S}^*| - m)/(|\mathcal{S}^*| - m) \geq 1/2 + 1/2s(n)$. Since $|\mathcal{S}^*|$ is polynomial, so is $r(n, |\mathcal{S}^*|)$, and therefore the concept class is weakly PAC-learnable; a contradiction, and the claim follows. □

Applications and Implications

We have chosen to motivate and present our framework in the context of a single list $\mathcal{S}^*$ of observed assignments that one seeks to complete further. The choice mostly serves to not clatter our central claims: *(i)* chaining predictors is provably beneficial, and *(ii)* the benefits can be realized by (and only by — at least for certain natural strategies) an iterative learning and prediction process. The same ideas and formal results presented herein can be applied to other settings that have been traditionally studied in Machine Learning.

The first point worth making is that the framework applies equally well to the traditional setting of *distinct* training and testing sets. There, $\mathcal{S}^*$ plays the role of the testing set, while another list $\mathcal{T}$ of assignments plays the role of the training set accessed by the base learning algorithm to provide the sought guarantees on each predictor P_i. The only modification needed to the framework is that of giving access to $\mathcal{T}$ instead of $\mathcal{S}^*$ to the learning algorithms. In the first scenario of the FLTC setting, $\mathcal{T}$ is a list of complete assignments drawn from the same distribution as those in $\mathcal{S}$. In the second scenario of the FLTC setting and in the SLAP setting, $\mathcal{T}$ is a list of assignments drawn from the same distribution as those in $\mathcal{S}^*$. Existing models of learning from complete (Valiant

1984; Michael 2011a) or observed (Michael 2010) assignments, show that these choices of $\mathcal{T}$ suffice to efficiently and reliably obtain the guarantees on P_i assumed herein.

A second important point is that the central claims of this work apply also to the traditional supervised classification / regression setting, where certain, possibly continuous and / or unobserved, features are used to inform a prediction on a *single* target attribute $x_t \in \mathcal{A}$, which is observed during training but not during testing. First, note that the proofs of our results, other than those in the SLAP section, go through even if we insist that some target attribute x_t is never observed in the testing set $\mathcal{S}^*$, and always observed in the training set $\mathcal{T}$ that one uses to produce each predictor P_i.

For the results in the SLAP section, we argue as follows: Assume that some target attribute x_t is always observed both in the testing set $\mathcal{S}^*$ and in the training set $\mathcal{T}$. Theorem 6 holds for this choice of $\mathcal{S}^*$. Now, mask x_t in all assignments in $\mathcal{S}^*$ to obtain $\overline{\mathcal{S}^*}$. The change from $\mathcal{S}^*$ to $\overline{\mathcal{S}^*}$ affects only the completeness of $\mathbb{P}$, and hence Theorem 6 still holds. To avoid affecting completeness also, and to ensure that Theorem 7 holds, choose a partitioning with $x_t \in \mathcal{A}_d$, so that no predictor can base its prediction on the value of x_t, and so that the completeness guarantees on $\mathcal{S}^*$ and on $\overline{\mathcal{S}^*}$ coincide.

So, chaining predictors is beneficial for supervised learning, and SLAP helps realize these benefits. Seeking to develop classification / regression algorithms that are robust to missing information is bound to either produce non-classical predictors or result in worse performance than SLAP.

A final point relates to using predictors to complete missing values, but never to override observed values. This view aligns with KRR work that treats observations as indefeasible, and as exogenously qualifying rules (Kakas, Michael, and Miller 2011; Michael 2013b). Although we have chosen to adopt this treatment, we note that predictors that override observed values would also be useful, especially in settings where a prediction might possibly correct an observed noisy value. This choice is inconsequential for our results herein.

Conclusions and Related Work

This work has initiated a formal investigation of interleaving learning and prediction with multiple predictors, examining the guarantees one may get under minimum assumptions.

Among works that interleave learning and prediction, Inductive Logic Programming (Muggleton 1991) typically differs from this work in: *(i)* placing less emphasis on computational efficiency; *(ii)* seeking to fit available data instead of providing statistical guarantees during prediction; *(iii)* producing predictors that never abstain (under the Closed World Assumption) instead of bounding their incompleteness.

Some form of interleaved learning and prediction is found in Transformation-Based Learning (Brill 1993), although little emphasis seems to have been placed on its formal analysis. TBL predictors aim to correct initial heuristic predictions rather than make predictions only when it is justified.

Rivest and Sloan (1994) learn predictors in a layered manner, with subsequent predictors being trained on the predictions of earlier ones. This resembles the SLAP setting, but their approach does not extend to assignments with missing information. Also, unlike us, they achieve super-constant depth of chaining, at the expense, however, of considering severely restricted classes of predictors. Attempting to integrate their and our approach is certainly worth looking into.

Valiant (2000) considers the parallel learning of predictors before their chaining, as in the first scenario under the FLTC setting. As in our approach, it is the availability of base algorithms with guarantees on complete assignments that allows the decoupling of learning and prediction. Both of our works place emphasis on efficiency and predictive guarantees. The relational learning and the two types of missing information that he employs can be carried over to our framework.

Read et al. (2009) investigate a version of multi-label classification with predictions for a label being available when training predictors for other labels. To account for their suboptimal random ordering of the predictors, they produce ensembles of such orderings. Instead of a distinguished fixed set of labels, we allow predictions on all attributes that may happen to be masked. Instead of an ensemble of orderings, we produce a single universal ordering. Finally, our emphasis is on the formal analysis and understanding of when SLAP is provably useful. To that end, the principled treatment of missing information and abstentions is essential.

Chaining learned predictors differs from learning circuits, neural networks, or other expressive functions, where the values of internal nodes are never observed during the training (or testing) phase. Furthermore, missing information (on internal nodes) is only implicitly completed during prediction, whereas we do so explicitly, and also allow abstentions.

In the spirit of implicit reasoning, Juba (2013) considers a setting of partially observed assignments closely following the one of our work (Michael 2010), and chains learned rules without, however, explicitly representing them. It remains an intriguing question to understand the comparative advantages of the implicit and explicit representations of learned rules, in terms of the performance (soundness and completeness) achieved when they are chained to draw inferences.

On the empirical front, the benefits of chaining have been demonstrated on the massive-scale extraction of commonsense knowledge from raw text (Michael and Valiant 2008) and on other language-related tasks (Doppa et al. 2011). The results obtained in this work can be seen to offer a formal basis for these experimental results, while the latter exemplify the applicability of our framework in real-world settings.

Some arguments and empirical evidence in other contexts also point to the benefits of chaining rules (Dietterich 2000; Valiant 2006), for reasons ultimately related to limitations of the base algorithms: hypothesis class expressivity, statistical and computational constraints, data availability. This work *formally* ties the benefits of chaining to a limitation of rules (cf. Lemma 1), and seeks ways to realize them. Certain ideas from our work have been argued (Michael 2009) to be important in the Recognizing Textual Entailment task (Dagan, Glickman, and Magnini 2005). The design of base algorithms has been considered (Michael 2010), with applications (Michael 2013a), and open problems (Michael 2011b).

The goal of enhancing completeness resembles the goal of "boosting" from Machine Learning, where one seeks to improve a weakly sound predictor. A typical boosting approach (Schapire 1990) proceeds by learning a sequence of predic-

tors, each trained on those assignments on which the preceding predictor was unsound. At this abstract level, the parallels with our work are immediate: SLAP attempts to improve a weakly complete predictor set by learning a sequence of such predictor sets, each trained on those assignments on which the preceding predictor set was incomplete. Of course there are critical differences as well, not the least of which being that an unsound prediction cannot be identified during testing, whereas an incomplete one can. Understanding the parallels and differences in more technical detail may allow boosting techniques to be adapted and adopted for the problem of improving the completeness of predictors.

Regarding discussions pointing out that Machine Learning research has focused on the construction of flat (in our defined sense) predictors (Dietterich 2003), this work offers a glimpse of what one may expect when moving to a chained (and more structured) use of predictors. The work presented herein offers a concrete basis upon which a large scale investigation of this intriguing phenomenon may be carried out.

References

Brill, E. D. 1993. *A Corpus-Based Approach to Language Learning*. Ph.D. Dissertation, University of Pennsylvania, U.S.A.

Dagan, I.; Glickman, O.; and Magnini, B. 2005. The PASCAL Recognizing Textual Entailment Challenge. In *Proc. of 1st PASCAL Challenges Workshop for Recognizing Textual Entailment (RTE'05)*, 1–8.

Dietterich, T. G. 2000. Ensemble Methods in Machine Learning. In *Proc. of 1st International Workshop on Multiple Classifier Systems (MCS'00)*, 1–15.

Dietterich, T. G. 2003. Learning and Reasoning. *Unpublished Manuscript*.

Doppa, J. R.; Nasresfahani, M.; Sorower, M. S.; Irvine, J.; Dietterich, T. G.; Fern, X.; and Tadepalli, P. 2011. Learning Rules from Incomplete Examples via Observation Models. In *Proc. of Workshop on Learning by Reading and its Applications in Intelligence Question-Answering (FAM-LbR/KRAQ'11)*.

Ehrenfeucht, A.; Haussler, D.; Kearns, M. J.; and Valiant, L. G. 1989. A General Lower Bound on the Number of Examples Needed for Learning. *Information and Computation* 82(3):247–261.

Juba, B. 2013. Implicit Learning of Common Sense for Reasoning. In *Proc. of 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, 939–946.

Kakas, A.; Michael, L.; and Miller, R. 2011. Modular-E and the Role of Elaboration Tolerance in Solving the Qualification Problem. *Artificial Intelligence* 175(1):49–78.

Kearns, M. J., and Li, M. 1993. Learning in the Presence of Malicious Errors. *SIAM Journal on Computing* 22(4):807–837.

Kearns, M. J., and Vazirani, U. V. 1994. *An Introduction to Computational Learning Theory*. The MIT Press.

Knuth, D. E. 2011. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms*. Addison-Wesley.

Michael, L., and Valiant, L. G. 2008. A First Experimental Demostration of Massive Knowledge Infusion. In *Proc. of 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, 378–388.

Michael, L. 2008. *Autodidactic Learning and Reasoning*. Ph.D. Dissertation, Harvard University, U.S.A.

Michael, L. 2009. Reading Between the Lines. In *Proc. of 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, 1525–1530.

Michael, L. 2010. Partial Observability and Learnability. *Artificial Intelligence* 174(11):639–669.

Michael, L. 2011a. Causal Learnability. In *Proc. of 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, 1014–1020.

Michael, L. 2011b. Missing Information Impedements to Learnability. In *Proc. of 24th Annual Conference on Learning Theory (COLT'11), JMLR: Workshop and Conference Proceedings*, volume 19, 825–827.

Michael, L. 2013a. Machines with WebSense. In *Proc. of 11th International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense'13)*.

Michael, L. 2013b. Story Understanding... Calculemus! In *Proc. of 11th International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense'13)*.

Mitchell, T. M. 1997. *Machine Learning*. McGraw Hill Series in Computer Science. McGraw-Hill.

Muggleton, S. H. 1991. Inductive Logic Programming. *New Generation Computing* 8(4):295–318.

Read, J.; Pfahringer, B.; Holmes, G.; and Frank, E. 2009. Classifier Chains for Multi-label Classification. In *Proc. of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD'09)*, 254–269.

Reiter, R. 1980. A Logic for Default Reasoning. *Artificial Intelligence* 13(1–2):81–132.

Rivest, R. L., and Sloan, R. 1994. A Formal Model of Hierarchical Concept Learning. *Information and Computation* 114(1):88–114.

Schapire, R. E. 1990. The Strength of Weak Learnability. *Machine Learning* 5(2):197–227.

Schuurmans, D., and Greiner, R. 1994. Learning Default Concepts. In *Proc. of 10th Canadian Conference on Artificial Intelligence (CSCSI'94)*, 99–106.

Valiant, L. G. 1984. A Theory of the Learnable. *Communications of the ACM* 27:1134–1142.

Valiant, L. G. 2000. Robust Logics. *Artificial Intelligence* 117(2):231–253.

Valiant, L. G. 2006. Knowledge Infusion. In *Proc. of 21st National Conference on Artificial Intelligence (AAAI'06)*, 1546–1551.

Valiant, L. G. 2013. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books.

Tackling Winograd Schemas by Formalizing Relevance Theory in Knowledge Graphs

Peter Schüller
Department of Computer Engineering, Marmara University, Goztepe Kampusu
34722 Kadikoy, Istanbul, Turkey
schueller.p@gmail.com

Abstract

We study disambiguating of pronoun references in Winograd Schemas, which are part of the Winograd Schema Challenge, a proposed replacement for the Turing test. In particular we consider sentences where the pronoun can be resolved to both antecedents without semantic violations in world knowledge, that means for both readings of the sentence there is a possible consistent world. Nevertheless humans will strongly prefer one answer, which can be explained by pragmatic effects described in Relevance Theory. We state formal optimization criteria based on principles of Relevance Theory in a simplification of Roger Schank's graph framework for natural language understanding. We perform experiments using Answer Set Programming and report the usefulness of our criteria for disambiguation and their sensitivity to parameter variations.

Introduction

The Winograd Schema Challenge (Levesque, Davis, and Morgenstern 2012) is an anaphora resolution task which was proposed as a challenging but more reasonable alternative for the Turing Test. The challenge has received interest in the Artificial Intelligence community and beyond (Marcus 2013) as a chance to push forward natural language *understanding* capabilities of computers.

Winograd Schemas (WSs) are pairs of sentences that are referential ambiguous, i.e., they can be semantically interpreted in at least two distinct ways. Only one of these interpretations is correct and humans easily identify the correct one.

$$\begin{array}{l}\text{There is [a gap]}_g \text{ in [the wall]}_w.\\ \text{You can see the garden through it}_{w,g?}.\end{array} \quad (1)$$

$$\begin{array}{l}\text{There is [a gap]}_g \text{ in [the wall]}_w.\\ \text{You can see the garden behind it}_{w,g?}.\end{array} \quad (2)$$

(1) and (2) show such a pair of sentences, where we denote phrases with square brackets and coreference using subscripts. Here the pronoun 'it' is ambiguous between noun phrases 'a gap' and 'the wall'. In (1) the solution is 'it_w' — we can see the garden through the gap, in (2) the solution is 'it_g' — we can see the garden behind the wall. Note that only a single word differs between (1) and (2). This WS can be disambiguated by reasoning over world knowledge such as 'walls are solid objects', 'solid objects are nontransparent', and 'one cannot see through nontransparent objects'. An interesting challenge of WSs is that they seem to *require deep reasoning* beyond the surface of the text, in particular beyond the current state of the art of (mostly statistical) natural language processing technology.

Some schemas can be disambiguated using linguistic theories of text structure, for example the following.

$$\begin{array}{l}\text{[Pete]}_p \text{ envies [Martin]}_m\\ \text{because he}_{p,m?} \text{ is very successful.}\end{array} \quad (3)$$

$$\begin{array}{l}\text{[Pete]}_p \text{ envies [Martin]}_m\\ \text{although he}_{p,m?} \text{ is very successful.}\end{array} \quad (4)$$

In (3) the word 'because' indicates that '$\text{he}_{p,m?}$ is very successful' is a plausible reason for 'Pete envies Martin'. As a result we can rule out the coreference 'he_p' because Martin's success is a more plausible cause for Pete's envy than Pete's success, hence the solution is 'he_m'.

Such reasoning is already challenging, however it is not always sufficient: there are WSs where both possible coreferences correspond to a world where the sentence makes sense — text structure and world knowledge are not sufficient for tackling Winograd Schemas.

One example for such a schema is the following.

$$\begin{array}{l}\text{[Sam's drawing]}_s \text{ was hung just above [Tina's } \emptyset]_t \text{ and}\\ \text{it}_{s,t?} \text{ did look much better with another one below it.}\end{array} \quad (5)$$

$$\begin{array}{l}\text{[Sam's drawing]}_s \text{ was hung just above [Tina's } \emptyset]_t \text{ and}\\ \text{it}_{s,t?} \text{ did look much better with another one above it.}\end{array} \quad (6)$$

(The '$\emptyset$' indicates an omission (ellipsis) of 'drawing'.)

In these sentences we can imagine a world where the drawing 'below it' or 'above it' is a third drawing, neither Sam's nor Tina's: then the pronoun can be either 'it_s' or 'it_t'.

Humans intuitively chose 'it_s' for (5) and 'it_t' for (6), intuitively because these readings of the sentences are much *more relevant* and *less far-fetched* than their alternatives.

In this work we focus on the disambiguation of such Winograd Schemas, schemas where we have to go beyond semantics and text structure because they are not sufficient for disambiguation.

We will approach this problem using Relevance Theory (Wilson and Sperber 2006), which measures relevance in

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

terms of *positive cognitive effect* – intuitively defined as the amount of inferences that are possible – minus the *processing effort* of achieving that effect. Relevance Theory is based on informal notions of human reasoning, hence integrating this theory into formal reasoning is not straightforward.

We propose a method for disambiguating WSs based on a formalization of criteria of Relevance Theory and present the following contributions.

- We formally describe a *knowledge graph* data structure which is a simplification of Roger Schank's psychological model of natural language understanding (Schank 1972).
- We formally describe a *reasoning process* that combines an input knowledge graph with background knowledge graphs, including constraints on certain intuitively necessary graph properties.
- Based on notions from Relevance Theory we propose a *relevance fitness function* on knowledge graphs.
- We report on *experiments* for *evaluating parameters* of that fitness function using Answer Set Programming.

Preliminaries

Winograd Schemas (WSs) were described by Winograd (1972), they are twin sentences with an ambiguous pronoun (see Hirst (1981) for a survey) and a *special word* such that exchanging this word with its *alternate word* changes the reference to another entity in the sentence. For example in the pair (1)/(2), the pronoun 'it' is ambigous between antecedent noun phrases 'a gap' and 'the wall', the special word is 'through' with its alternate word 'behind', the correct coreference is 'it_g' for (1) and 'it_w' for (2).

Levesque, Davis, and Morgenstern (2012) proposed the Winograd Schema Challenge as an alternative for the Turing Test, arguing that disambiguating WS requires true understanding of text on the one hand, and is more practical and more reasonable than the Turing Test on the other hand because (i) it does not require deception by the computer, (ii) does not require conversation, (iii) permits objective and automatic scoring, and (iv) requires deep reasoning over the input and the utilization of world knowledge.

They describe two important conditions on WSs used in such a challenge: (1) it shall not be possible to disambiguate the coreference by *selectional restrictions*, i.e., both antecedents must be of the same gender and number, moreover *straightforward statistical methods* shall not allow a clear disambiguation.[1] Moreover, (2) untrained human subjects shall identify the *correct* answer *without effort*.

The aim of the first condition is to require true text understanding to achieve success in the challenge, the second condition aims at a strong methodology with repeatable tests.

The original proposal of the challenge contains sentence pairs with a question, where the answer of the question asks for the referent of the pronoun. We will omit that question and instead check if the pronoun was resolved correctly.

Pragmatics is the linguistic field that analyses the effects of *context* on utterances. Note that'utterance' in linguistics denotes written, spoken, and signed natural language, similarly 'speaker' resp., 'hearer' denotes producers resp., consumers of natural language in all modalities. Moreover 'context' denotes linguistic, historical, cultural, and situative context.

From the perspective of WSs, pragmatics can provide useful insights about how the first part of a WS (the context) influences anaphora resolution in the second part (where the pronoun is located).

Relevance Theory (RT) is a popular theory within pragmatics, developed by Wilson and Sperber (2006). Their *Communicative Principle of Relevance* states that every utterance raises the expectation of its own optimal relevance, i.e., a hearer may interpret an utterance under the assumption that it is most relevant among all possible alternative utterances.

RT aims to 'explain in cognitively realistic terms what these expectations of relevance amount to, and how they might contribute to an empirically plausible account of comprehension.' This aim led to cognitive results that we will carry over to terms of knowledge representation, automated reasoning, and computation.

We will focus on three definitions of RT: relevance, processing effort, and positive cognitive effect.

Relevance of an utterance depends on two factors: other things being equal, (a) the greater the *processing effort* spent on interpreting an utterance, the lower its relevance, and (b) the greater the *positive cognitive effects* achieved by interpreting an utterance, the greater its relevance.

Processing effort is defined as the effort spent on perception, memory, and inference. Different contexts have different levels of accessibility (salience) for certain concepts such that processing effort for the same utterance can differ greatly between contexts.

Positive Cognitive Effect is 'a worthwhile difference to the individual's representation of the world'. Concretely it is defined wrt. the knowledge that can be concluded by a hearer using some mechanism of inference in human cognition.

According to RT, the most important cognitive effect is *contextual implication*: knowledge inferred from context C and utterance U that is not inferrable from C or U alone. Further positive cognitive effects are strengthening, revision, or abandonment of available *assumptions*.

Roger Schank's Knowledge Representation (1972) aims to provide a theory of human natural language understanding based on a graph formalism of concepts and dependencies that represent utterances and whole discourses.

A concept is either a nominal (e.g., 'book' or 'John'), an action (e.g., most verbs), or a modifier (e.g., adjectives). Dependencies can be of several types and directions, for example 'John $\leftrightarrow$ sleep' means that John sleeps. They can be annotated with temporal information, e.g., 'I $\leftrightarrow^p$ hit $\leftarrow^o$ John' means that John was hit by me (o indicates the object role and p the past).

Reasoning is described as looking up graph snippets from a library when encountering related input, a 'conceptual processor' that connects graph snippets according to conceptual rules, a 'syntactic processor' that ensures no syntactic constraints are violated. Later work by Schank (1980) makes several of these ideas more concrete (in particular memory

[1] As an example, if the antecedents are 'women' and 'pills' and the special words are 'pregnant' and 'carcinogenic', then the schema is too obvious to resolve, it is 'not Google-proof'.

organization) but is still far away from a formal account of how that reasoning is performed.

Related Work

Rahman and Ng (2012) proposed a machine-learning approach for tackling Winograd Schemas. They take into regard features such as narrative chains, FrameNet information, semantic compatibility, and polarity biases, and report a correctness result of 73% compared to 55% obtained via the state-of-the-art Stanford resolver (Lee et al. 2011). Some of their features are related to biases in language cognition.

Such biases are described by Kehler et al. (2008) who describe that pronoun interpretation is influenced by which coherence relations are likely to appear and which expectations about following entities are likely to occur. This is explained in more detail by Hartshorne (2013): e.g., a pronoun in a 'result' clause will most likely refer to the previous clauses' affected entity, while a pronoun in an 'explantory' clause most likely refers to entities that are causes.

As WSs are twin sentences, these biases should not make disambiguation easier. However, note that this is not true, if the word that evokes the bias is the special word itself (i.e., 'because' and 'although' should not be special words).

Relevance Formalizations. Gärdenfors (1978) discusses a logical account of relevance that formalizes whether some piece of logical evidence can influence the truth value of a sentence. Delgrande and Pelletier (1998) go a step further and define relevance as a property of sentences in relation to conditionals in conditional logic (which allows nonmonotonic inferences). They state that relevance in a logical formalism must be defined on the meta-level instead of within that formalism. We similarly apply our relevance fitness function on a meta-level to evaluate the reasoning process and its potential outcomes.

Understanding natural language requires taking assumptions and truly engaging in a process of *interpreting* a given linguistic input. This makes natural language so efficient but allows for complex misunderstandings. Relevance Theory, and the notion of relevance we pursue in this work, aims to pinpoint the *correct assumptions* a hearer has to make about a linguistic input such that the hearer obtains the interpretation intended by the speaker. (E.g., a flight attendant can use 'seat 3a' to refer to a passenger or to the purchase of a passenger at that seat.) Hence our notion of relevance is related to Gärdenfors, Delgrande and Pelletier's approaches but it is about human interpretation rather than about logical truth.

Hobbs et al. (1993) interpret natural language by abducing as few as possible assumptions to explain as much as possible observed input. The approach we introduce in the following can be considered to be abduction, but we evaluate relevance not only on the amount of abduced pieces of knowledge but in a more holistic way that considers global as well as local graph properties. Moreover our approach considers how well background knowledge integrates into the input during the reasoning process.

Textual Structure. As noted in the introduction, many Winograd Schemas can be solved by identifying expectations evoked by the textual structure of the input, and verifying if these expectations are violated by the semantic content of one or the other reading of the input.

One theory explaining such expectations is Rhetorical Structure Theory (RST) (Mann and Thompson 1988; Taboada and Mann 2006) which describes text organization as relations between parts of text, mostly between pairs of adjacent text spans. Such relations can be explicit in an utterance, e.g., the words 'if' or 'because', but they can be implicit, which creates the challenge of detecting them.

RST argues with similar principles as Relevance Theory, explaining textual structure as *providing relevant knowledge* that can be *integrated readily* into context and also *strengthening or justifying assumptions*. Both theories are likely to be necessary in a system that can disambiguate WSs.

Graph-Based Knowledge Representation. Important works related to our graph knowledge representation are knowledge bases and ontologies for natural language processing such as the Component Library (Barker, Porter, and Clark 2001), Generalized Upper Model (Bateman et al. 2010), and YAGO (Suchanek, Kasneci, and Weikum 2007). These and similar ontologies use formal logic (often Description Logics (Baader et al. 2003)) to categorize and relate linguistic terms, they are systematically engineered to cover a broad range of domains and applications. Contrary to that we use a lightweight graph framework to represent linguistic input and background knowledge, with the aim of studying the formalization of relevance. We hope that the results of this work can be applied also to these more systematic ontologies, as they have natural representations as graphs. As stated by Chaudhri, Dinesh, and Inclezan (2013), to be useful in natural language processing an ontology must be *linguistically motivated*.

Conceptual Blending (Pereira 2008) is an approach that is structurally similar to our knowledge graphs, however Blending has the goal to create *creativity* in AI systems. This is achieved by 'blending' knowledge graphs using associations of concepts that have similar properties, giving rise to 'metaphorical' and 'creative' computational reasoning.

Relevance in Conceptual Blending is formulated as a fitness *with respect to a reasoning goal*, for example the 'horse' and the 'bird' microtheory can be blended in different ways, and if we impose the goal 'flying' then a blend that is compatibly with flying is more relevant.

Knowledge Graphs and Reasoning

Schank aimed to provide a psychological model of how human reasoning works for understanding text. We next define a graph framework inspired by this model and formalize reasoning based on combining nodes of knowledge graphs without violating certain constraints of these graphs.

We simplify the framework to a form that is sufficient for our purposes. Most importantly we distinguish dependencies only by label, not by drawing direction or arrow style.

Knowledge Graphs

We define the knowledge graph data structure based on two vocabularies $\mathcal{C}_{cont}$ and $\mathcal{C}_{dep}$ which are the domains for labels of nodes and dependencies in our knowledge graph, resp.

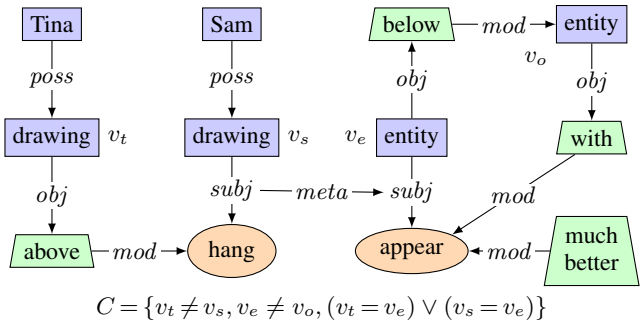


Figure 1: Knowledge graph structure corresponding to (5).

Example 1. *Figure 1 shows as an example the knowledge graph corresponding to input sentence* (5) *(we omit temporal information and interpret 'X's drawing' as 'the drawing owned by X'). Node labels are from the set* $\mathcal{C}_{cont} = \{$Tina, Sam, drawing, entity, hang, appear, above, below, with, much better$\}$. *Some intuitions about this graph: the* obj *dependency between* v_e *and 'below' denotes that* v_e *is the argument of 'below'; the 'below* v_e*' concept modifies* v_o *such that* v_o *denotes 'an entity that is below* v_e*'; moreover* v_e *has a* $subj$ *role in 'appear(ing)' which is modified by 'much better' and by 'with* v_o*'. The* $meta$ *dependency denotes an underspecified textual structure relation.*

In the following we implicitly extend this vocabulary as needed. The dependency vocabulary $\mathcal{C}_{dep} = \{subj, obj, obj2, mod, poss, at, meta\}$ is fixed.

We define an oracle function $compat : 2^{\mathcal{C}_{cont}} \to \{0, 1\}$ that evaluates whether a subset $C \subseteq \mathcal{C}_{cont}$ of the vocabulary contains mutually compatible content. Intuitively $compat$ abstracts from a (possibly intricate) reasoning in an ontological knowledge base that determines whether linguistically distinct concepts can denote the same world concept. For example, a drawing and an entity can be present in one concept node, while a drawing and Sam cannot: they must stay in separate concept nodes.

In Schank's graphs, dependencies occur between nodes and between dependencies, hence we reify dependencies: we represent a dependency as a graph node and two edges.

Definition 1. *A* knowledge graph structure $G = (V, E, l_{cont}, l_{type}, C)$ *on vocabularies* $\mathcal{C}_{cont}$ *and* $\mathcal{C}_{dep}$ *is a directed graph with nodes* V*, edges* $E \subseteq V \times V$*, node labelling functions* $l_{cont} : V \to 2^{\mathcal{C}_{cont} \cup \mathcal{C}_{dep}}$ *and* $l_{type} : V \to 2^{\{\mathbf{act}, \mathbf{obj}, \mathbf{mod}, \mathbf{dep}\}}$*, and a set of reasoning constraints* C *of two forms: inequalities between vertices and disjunctions of equalities between vertices.*

Figure 1 depicts a knowledge graph structure. Here and in the following we show nodes of type **act**, **obj**, and **mod**, as ellipses, rectangles, and trapezes, respectively. Dependencies (nodes of type **dep**) are shown without frame, and we omit arrows into such nodes unless they are $meta$-dependencies. Note in particular the $meta$ dependency between two $subj$ dependencies.

Given a set C of reasoning constraints we denote by $e(C)$ the set of disjunctions of equalities and by $n(C)$ the set of inequalities in C. In our example, $e(C) = \{(v_t = v_e) \vee (v_s = v_e)\}$ and $n(C) = \{v_t \neq v_s, v_e \neq v_o\}$.

We next impose constraints on knowledge graph structures to define knowledge graphs. The constraints ensure that knowledge graphs have certain properties, motivated methodologically (e.g., nodes have only one type, dependencies have only one incoming and one outgoing edge to a concept node) or linguistically (e.g., a limitation of object and possessor dependencies for concept nodes).

Equality constraints are inspired by constraints in Minimal Recursion Semantics (Copestake et al. 2005) which is a formalism for parsing natural language that allows ambiguous representations for unresolved pronouns as in WSs. Inequality constraints arise from syntactic principles of natural language, for example Principle B of Government and Binding (Haegeman 1994) states that a pronoun must not be bound within the same linguistic clause.

The following definition restricts each node $v \in V$ to have exactly one type $l_{type}(v)$. Using this restriction, we denote by $\mathbf{dep}(V) = \{v \in V \mid l_{type}(v) = \{\mathbf{dep}\}\}$ the set of *dependency nodes* and by $\mathbf{con}(V) = \{v \in V \mid l_{type}(v) \in \{\{\mathbf{act}\}, \{\mathbf{obj}\}, \{\mathbf{mod}\}\}\}$ the set of *concept nodes*.

Definition 2. *A* knowledge graph $G = (V, E, l_{cont}, l_{type}, C)$ *on vocabularies* $\mathcal{C}_{cont}$ *and* $\mathcal{C}_{dep}$ *is a knowledge graph structure such that*

(A) for every $v \in V$ *it holds that* $|l_{type}(v)| = 1$*;*
(B) for every $v \in \mathbf{dep}(V)$ *it holds that* $l_{cont}(v) \subseteq \mathcal{C}_{dep}$ *and* $|l_{cont}(v)| = 1$*;*
(C) for every $v \in \mathbf{con}(V)$ *it holds that* $l_{cont}(v) \subseteq \mathcal{C}_{cont}$ *and* $compat(l_{cont}(v)) = 1$*;*
(D) for $r \in \{obj, obj2, poss\}$ *and for every* $v \in V$ *it holds that* $|\{u \in V \mid (u, v) \in E \text{ and } l_{cont}(u) = \{t\}\}| \leq 1$.
(E) for every $v \in \mathbf{dep}(V)$ *it holds that* $|\{u \in \mathbf{con}(V) \mid (u, v) \in E\}| \leq 1$ *and* $|\{u \in \mathbf{con}(V) \mid (v, u) \in E\}| \leq 1$.
(F) for every $u \in \mathbf{con}(V)$*, it holds that there is at most one pair* (v, w)*,* $v \in \mathbf{dep}(V)$*,* $l_{cont}(v) = \{at\}$*,* $w \in \mathbf{con}(V)$ *such that* $\{(u, v), (v, w)\} \subseteq E$.
(G) there is no pair (u, v) *with* $u \in \mathbf{con}(V)$*,* $v \in \mathbf{dep}(V)$*, and* $\{(u, v), (v, u)\} \subseteq E$.
(H) for every pair $u, v \in \mathbf{con}(V)$*,* $u \neq v$ *it holds that* $|\{w \in V \mid \{(u, w), (w, v)\} \subseteq E\}| \leq 1$ *and* $|\{w \in V \mid \{(v, w), (w, u)\} \subseteq E\}| \leq 1$.

Note that Figure 1 is a knowledge graph.

These conditions become useful next where we define a reasoning task that combines nodes in knowledge graphs. (A) ensures single types for nodes. (B) ensures that dependency nodes have a singleton content from $\mathcal{C}_{dep}$. (C) ensures that concept nodes have as content a subset of the vocabulary that can refer to the same object or activity or modifier in the world. (D) ensures that a node can have only a single dependency from an object (obj) second object ($obj2$) and possessor ($poss$). This requirement is related to a linguistic principle known as 'theta criterion' which states that if a word assigns theta roles (types of arguments) then each role must be filled by a unique object (for example the word 'between' requires two arguments of type obj and $obj2$ for tangible or numerical entities). (E) ensures that a dependency node has no more than one incoming and no more than one

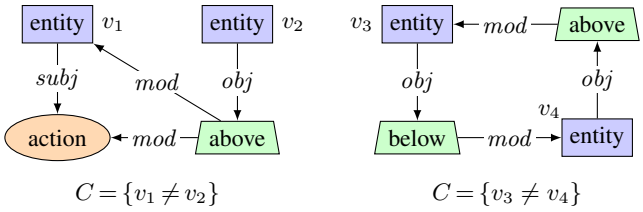


Figure 2: Background Knowledge Graphs related to (5).

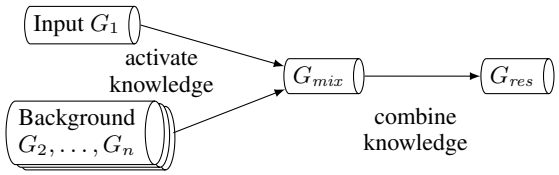


Figure 3: Reasoning with Knowledge Graphs

outgoing edge from (resp., to) a concept node. This causes dependency nodes to be like arcs between concept nodes (as visible in the figure) with the additional possibility of dependencies to dependencies. (F) ensures that a concept is at a single location only. (G) ensures that no concept depends on itself. (H) ensures that two concept nodes are never connected via more than one distinct dependency (in each direction). This avoids redundancies which will become important for realizing relevance in knowledge graphs.

From here on we leave vocabularies implicit.

Reasoning

We next define a notion of reasoning based on activating background knowledge graphs and combining them by collapsing subsets of nodes into single nodes. In the next section we define criteria for comparing all possible graphs resulting from such reasoning, yielding only those graphs that are most relevant wrt. principles of Relevance Theory.

Example 2. *Figure 2 depicts two knowledge graphs that are background knowledge related to our example. The left graph expresses that 'entity v_1 doing something above entity v_2' is interrelated with 'entity v_1 being above v_2'. The right graph expresses that 'entity v_3 being below entity v_4' is interrelated with 'entity v_4 being above v_3'. Note that these graphs does not imply a 'directionality', there are no 'conditions' or 'effects', just multiple dependency chains between concepts.*

Figure 3 gives an overview of our reasoning process: we activate knowledge in G_{mix} and then combine the activated graphs into result graph G_{res}.

Activating Knowledge. We start with a knowledge graph G_1 representing the input and a set $BG_{act} = \{G_2, \ldots, G_n\}$ of *activated knowledge graphs* from a background knowledge collection BG ($BG_{act} \subseteq BG$). For this work it was sufficient to activate knowledge $BG_{act} \subseteq BG$ according to matches of node contents that are more specific than 'entity' or 'action'. For a larger-scale system, methods that rely on salience and semantic distance should be used, for example Distributional Semantics (Padó and Lapata 2007).

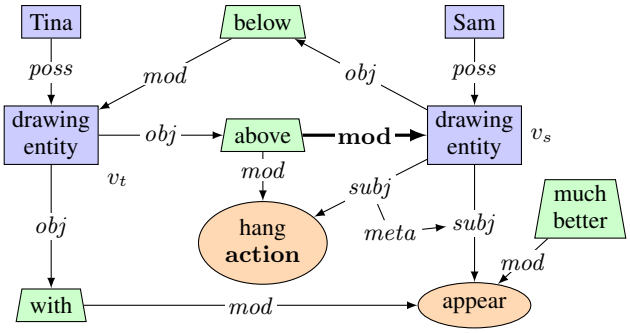


Figure 4: Reasoning: combining graphs from Figure 1 and 2.

Background knowledge is activated because it is somehow related to the input, however some of that knowledge might not be useful for reasoning. Hence we select a subset $BG_{use} \subseteq BG_{act}$ of *used knowledge graphs* from activated knowledge. The next section specifies how this selection is realized.

For combining knowledge we put all graphs in BG_{use} into one graph structure. For that we define a union operator: given two knowledge graphs G_1 and G_2 with disjoint sets of nodes, $G_i = (V_i, E_i, l_{cont_i}, l_{type_i}, C_i)$, $i \in \{1, 2\}$ we obtain $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2, l_{cont_1} \cup l_{cont_2}, l_{type_1} \cup l_{type_2}, C_1 \cup C_2)$. Due to disjoint sets of nodes the result is again a knowledge graph. If necessary, we can achieve disjointness by renaming nodes.

Combining Knowledge. We transform a knowledge graph by combining sets of nodes into single nodes. For that we use a function that maps nodes to other nodes.

Definition 3. *Given a knowledge graph structure $G = (V, E, l_{cont}, l_{type}, C)$, a* target function *$target : V \to V$ is a total function such that $target(v) = u$ with $v \neq u$ implies $target(u) = u$.*

A target function induces an equivalence relation over V: nodes that are mapped to themselves are representatives of the equivalence class and all other nodes map to such nodes. We use a function representation because it is more practical for our purposes.

The effect of applying a target function to a knowledge graph is defined next.

Definition 4. *Given a knowledge graph structure $G = (V, E, l_{cont}, l_{type}, C)$ and a target function $target$ wrt. G, we denote by $G[target]$ the* application of $target$ to G *and define $G[target] = (V', E', l_{cont}', l_{type}', C')$ with*

- $V' = \{target(v) \mid v \in V\}$,
- $E' = \{(target(u), target(v)) \mid (u, v) \in E\}$,
- $l_{cont}'(v) = \bigcup\{l_{cont}(u) \mid u \in V,\ target(u) = v\}$,
- $l_{type}'(v) = \bigcup\{l_{type}(u) \mid u \in V,\ target(u) = v\}$, *and*
- $C' = \{c$ *with vertices mapped using* $target \mid c \in C\}$.

The graph $G[target]$ is a knowledge graph structure, however it might not be a knowledge graph (e.g., a vertex might obtain two types). We next restrict $target$ such that

$G[target]$ is again a knowledge graph structure, and define the effect of reasoning constraints.

Reasoning constraints enforce structural properties of knowledge graphs: inequality constraints ensure that two nodes remain distinct nodes during reasoning, while equality constraints ensure that certain nodes are collapsed with others. In our example sentence (5) the word 'another' causes the constraint $v_e \neq v_o$ in Figure 1 while the word 'it' must bind to a predecessor phrase which can either be 'Sam's drawing' or 'Tina's [drawing]', hence we have the constraint $(v_t = v_e) \vee (v_s = v_e)$.

Definition 5. *Given a knowledge graph structure $G = (V, E, l_{cont}, l_{type}, C)$, a target function $target$ wrt. G is a* reasoning function *if*

(a) $G[target]$ is a knowledge graph,
(b) for each inequality constraint $(u \neq v) \in n(C)$ it holds that $target(u) \neq target(v)$, and
(c) for each equality constraint $c \in e(C)$ there is an equality $(u = v)$ in c such that $target(u) = target(v)$.

As an example consider Figure 4 where we combine knowledge graphs. If we disregard the bold **mod** edge and **action** concept then the graph is just a reorganized input graph (Figure 1) with $target(v_o) = v_t$ and $target(v_e) = v_s$. Combining the left graph in Figure 2 with the input graph is possible by combining 'action' with 'hang', and other nodes with nodes containing the same content, which leads to an additional dependency mod between 'above' and node v_s. Now we can see that the right graph from Figure 2 is already contained in the resulting graph, precisely we can combine v_3 with v_s and v_4 with v_t and then nodes containing 'below' and 'above', and this generates no additional edges or nodes. Note that these operations do not violate reasoning constraints, and that in the resulting set of constraints C' we still have $v_t \neq v_s$ but we omit the equivalence constraint because it is satisfied as v_s and v_e becoming the same node.

Observe that the result graph depicted in Figure 4 is a correct solution to the Winograd Schema (5) because in that schema 'it' must refer to Sam's drawing and the graph identifies v_e with v_s.

Finally we formally define a reasoning process based on which we will next formulate criteria of relevance.

Definition 6. *Given input graph G_1 and background knowledge BG, our reasoning process deterministically obtains BG_{act}, then chooses BG_{use} and $target$, and thereby obtain the result graph $G_{res} = G_{mix}[target]$ where*

$$G_{mix} = \bigcup \big(\{G_1\} \cup BG_{use}\big).$$

Relevance (Theory) in Knowledge Graphs

The previous section defined a reasoning process that leaves two important points open: (i) how to select BG_{act} and (ii) how to select $target$. In this section we use concepts from Relevance Theory to *assigning a relevance fitness value* to a reasoning process. This fitness value induces a preference between possible choices for BG_{use} and $target$.

By comparing fitness values wrt. BG_{act} and $target$ we can find the *most relevant* reasoning outcomes for a given input. This approach corresponds well with RT, which states that relevance cannot be measured in absolute terms, but it is always *in comparison with alternatives*.

RT originally is a theory of human cognition of natural language. We next justify and sketch possible formalizations of important principles from RT, then we propose a fitness function based on these ideas.

Nodes in G_{res} that originate from at least one node in the input graph G_1 we call *input nodes* (V_{res}^{in}), while nodes that originate only from nodes in background knowledge graphs we call *background nodes* (V_{res}^{bg}). In our example in Figure 4, the only background node is the bold **mod** dependency node, all other nodes are input nodes because the remaining (seven) nodes from background knowledge are combined with input nodes.

We consider the following formalizations of relevance.

• The amount of nodes from background knowledge that can been combined with input nodes indicates how well background knowledge integrates with the input. If all nodes from a background knowledge graph are combined with input nodes, then that background graph *supports existing knowledge* or *strengthens existing assumptions*. Both is defined as positive cognitive effect in RT.

• Background nodes can form *contextual implications* in the graph, also contributing to positive cognitive effect: they can connect parts of the input graph (as in Figure 4) that are not connected originally. Such connections are like inferences using background and input graph knowledge.

• Background nodes can be close to input nodes or far away in the graph (in terms of the shortest path to some input node). The former nodes are closely related to input, while the latter are more 'far-fetched', they can be seen as *additional assumptions*.

Assuming additional concepts can be necessary to make sense of an input, however this imposes additional *processing effort* and therefore reduces relevance relative to alternatives. In our example, we can combine the right graph in Figure 2 with Figure 1 in a more far-fetched manner: instead of combining v_3 with v_s and v_4 with v_t we could combine v_3 with v_t and not combine v_4 with any input node. In that case our reasoning process makes the *assumption* that there is a third object that is below Tina's drawing.

• Another effect of reasoning with background knowledge is that it can connect nodes from the input graph and thus causes the overall graph to become more connected. To measure how well the result of reasoning matches expectations in background knowledge, and how well-integrated the result of reasoning is, it can therefore be useful to measure the *radius* of the resulting knowledge graph.

The graph radius is the smallest value r such that from some node of the graph we can reach all other nodes traversing less than r edges. Our input graph in Figure 1 has radius 7 (from the left $subj$ node), while the result graph in Figure 4 has radius 5 (from the bold mod node). Recall that dependencies are nodes, so from 'Tina' to 'meta' we traverse 8 edges.

Based on our definition of reasoning process, we next define a fitness function with constant parameters C_{in}^{com}, C_{bg}, C_{bg}^{com}, C_{bg}^{dis}, C^{om}, and C^{rad}.

Definition 7. *Given G_1, BG, BG_{use}, and target as in Definition 5, we define fitness function f_{rel} as*

$$f_{rel}(G_1, BG, BG_{use}, target) = C_{in}^{com} \cdot \sum_{n \in V_{res}^{in}} com(n) + C_{bg} \cdot |V_{res}^{bg}| + C_{bg}^{com} \cdot \sum_{n \in V_{res}^{bg}} com(n) + C_{bg}^{dis} \cdot \sum_{n \in V_{res}^{bg}} dis(n) + C^{om} \cdot |BG_{act} \setminus BG_{use}| + C^{rad} \cdot rad(G_{res})$$

where

$$\begin{aligned} V_{res}^{in} &= \{target(v) \mid v \in V_1\}, \\ V_{res}^{bg} &= V_{res} \setminus V_{res}^{in}, \\ com(n) &= |\{v \in V_{res} \mid target(n) = target(v)\}|, \\ dis(n) &= \textit{the length of the shortest path from } n \\ &\quad \textit{to some node } n' \in V_{res}^{in}, \textit{ and} \\ rad(G_{res}) &= \textit{the radius of graph } G_{res}. \end{aligned}$$

Informally, V_{res}^{in} denotes input nodes, V_{res}^{bg} denotes nodes background nodes, $com(n)$ denotes the number of nodes combined to form node $n \in V_{res}$, and $dis(n)$ denotes the distance of node $n \in V_{res}^{bg}$ to the nearest node $n' \in V_{res}^{in}$.

This fitness function contains all points stated informally before, with (so far) unspecified constant factors as parameters. C_{in}^{com} allows to measure how many nodes were combined to yield input nodes. For background nodes, C_{bg} gives uniform weight to each one, C_{bg}^{com} gives weight proportional to the amount of combined nodes, and C_{bg}^{dis} gives weight proportional to the distance from input nodes. Moreover C^{om} allows for giving a weight to each activated background knowledge graph that was not used, and C^{rad} gives a weight to the radius of the result graph G_{res}.

Note that C_{in}^{com}, C_{bg}, and C_{bg}^{com}can be computed locally for each node of G_{res}, while C_{bg}^{dis} requires consideration of more than one node, C^{rad} is about a property of the whole graph, and C^{om} is not about the graph but about which background knowledge is used to build G_{res}. These locality properties can have significant influence on implementability and computational efficiency of an optimization procedure that computes graphs with optimal fitness. In this work we consider only small inputs to show qualitative applicability of the fitness function, therefore we will only briefly touch on matters of computational efficiency.

Experimental Evaluation

We designed our relevance fitness function based on principles of RT, moreover we know that disambiguation of certain WSs can be explained in terms of relevance.

In this section we perform experiments on such schemas in order to show which parameters lead to a correct disambiguation, and how well our fitness function performs in general, i.e., whether it captures the requirements on disambiguating our example schemas.

For our empirical experiments we selected Winograd Schemas numbered 2, 5, 15, and 17 from (Levesque, Davis, and Morgenstern 2012), encoded them in both of their variants as input graphs, and encoded some relevant background knowledge (including Figure 2). We repeat these schemas as (7)+(8), (5)+(6), (9)+(10), and (11)+(12), respectively.

[Tom]$_t$ threw his schoolbag down to [Ray]$_r$ after [he]$_{t,r?}$ reached the top of the stairs. (7)

[Tom]$_t$ threw his schoolbag down to [Ray]$_r$ after [he]$_{t,r?}$ reached the bottom of the stairs. (8)

The [drain]$_d$ is clogged with [hair]$_h$. [It]$_{d,h?}$ has to be removed. (9)

[The drain]$_d$ is clogged with [hair]$_h$. [It]$_{d,h?}$ has to be cleaned. (10)

There is [a pillar]$_p$ between me and [the stage]$_s$, and I can't see [it]$_{p,s?}$ (11)

There is [a pillar]$_p$ between me and [the stage]$_s$, and I can't see around [it]$_{p,s?}$. (12)

We chose these schemas because they do not violate text structure or world knowledge in their wrong reading, therefore we think Relevance Theory is a major ingredient in solving them.[2]

For each schema we computed the reasoning outcome(s) that were ranked as most relevant by the fitness function f_{rel}. We measured the number of result graphs with correct disambiguation (i.e., graphs where the node corresponding to the ambiguous pronoun is combined with the node of the input that corresponds to the correct answer) versus the number of result graphs with an incorrect disambiguation.

We performed this experiment on the following parameter combinations.

- $C_{in}^{com} \in \{0, 1, 2, 3\}$ varies the importance of combining background knowledge with input nodes.
- $(C_{bg}, C_{bg}^{com}, C_{bg}^{dis}) \in \{(0,0,0), (-1,0,0), (0,0,-1), (0,1,-2), (0,1,-1), (0,2,-1)\}$ varies the importance of several features of background nodes. In particular (0,0,0) omits any cost or benefit, (-1,0,0) gives a uniform cost to every background node, (0,0,-1) imposes a cost corresponding to distance from the nearest input node, (0,1,-1) additionally gives a benefit for combining with other nodes, finally (0,1,-2) and (0,2,-1) vary the ratio between cost and benefit of distance from input nodes and combination with background knowledge.
- $C^{om} \in \{0, -10, -20, -30\}$ varies the cost of not using an activated background knowledge graph.
- $C^{rad} \in \{0, -5, -10, -15\}$ gives cost to the graph radius.

These parameter ranges were established from estimates of parameter importance and in preliminary experiments. The seemingly natural setting $(C_{bg}, C_{bg}^{com}, C_{bg}^{dis}) = (0, 1, 0)$ was excluded because it facilitates the use of background knowledge that is neither combined with other background knowledge nor with input knowledge.

The above parameter combinations yield 384 overall parameter combinations and we have 8 sentences (2 for each schema), yielding 3072 overall experimental runs.

We encoded the reasoning process in Answer Set Programming (ASP) (Lifschitz 2008; Brewka, Eiter, and

[2]There can be two stairs, a plumber might remove a clogged drain, and there can be an object blocking the view to the pillar.

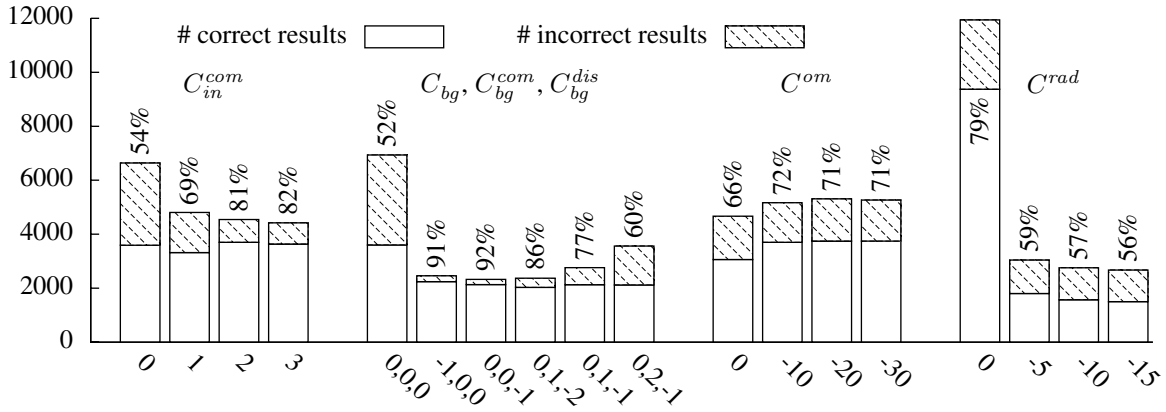


Figure 5: Experimental results (for each parameter group we accumulated over all combinations of remaining parameters).

Truszczynski 2011) which is a logic programming formalism suitable for representing knowledge and guess-and-check computations with integer optimization. Encodings, graph visualizations, and results are available online.[3]

In our experiments we used the solver clingo 4.2.1 (Gebser, Kaufmann, and Schaub 2013) with a timeout of 1 hour for finding the optimal value and 1 hour for enumerating a maximum of 25 graphs of optimum cost. Note that our encodings are not tuned for good performance, hence some parameter combinations and inputs cause timeouts and we only consider results from runs without timeouts. Such timeouts happened only for inputs (7) and (8), due to a high number of activatable background knowledge graphs. To gain confidence in our results, we checked that all trends we report in the following are similar if we use a timeout of only 20 min, hence timeouts do not invalidate the following discussion.

Results

Figure 5 shows results for each of the four parameter groups described previously, accumulated over all possible combinations of the other three parameter groups. We display the number of enumerated reasoning graphs, split into correctly and incorrectly disambiguated graphs, we also show the percentage of correct disambiguations.

We aim for a relevance fitness function that has the following properties: (i) firstly it gives higher fitness to graphs that contain the correct disambiguation, and (ii) it select as few as possible best results. The benefit of (i) is obvious, while (ii) is desireable for reasons of relevance: a low total number of results means that few assumptions needed to be taken for achieving these results, and few assumptions are necessary for subsequent reasoning.

From Figure 5 we can observe the following.

- Increasing C_{in}^{com} improves the correctness and reduces the number of result graphs.
- Giving weight to the second group of parameters $(C_{bg}, C_{bg}^{com}, C_{bg}^{dis})$ is important for correctness and for reducing the number of result graphs. The best correctness results are achieved with (-1,0,0), (0,0,-1), and (0,1,-2).
- A nonzero C^{om} increases correctness and increases the number of results (due to encouraging usage of background graphs).
- Giving weight to C^{rad} reduces correctness from 79% to 59% and drastically reduces the number of results.

These accumulated results give an overview of the impact of our fitness function constants. We next look into details.

Table 1 shows the best and worst individual parameter combinations. Each row is the result of 8 runs (one for each sentence of each schema).

In the upper part of the table we first show the 11 best combinations, they yield 100% correct answers and only 10 answers for 8 inputs. We can observe that C_{in}^{com} and C^{rad} are never 0, furthermore the combinations (0,0,0) and (0,2,-1) do not appear in the second column, which indicates that these parameter values are not favorable for correctness. C^{om} seems to be least important. The table row with 11 correct answers and 100% correctness actually contains 30 different parameter configurations, these combinations are not shown for space reasons, but we note that these combinations contain neither zero constants for C_{in}^{com}, C^{om}, and C^{rad}, nor combinations (0,0,0) or (0,2,-1) for the other constants. This suggests that all possible parameter values except (0,0,0) and (0,2,-1) and zero values contribute to successful disambiguation.

In the lower part of the table we show the 8 worst settings. (C_{bg},C_{bg}^{com},C_{bg}^{dis}) is either (0,0,0) which gives no weight to background nodes, or (0,1,-1), which gives cost to distance from input nodes but also gives the same amount of benefit for combination with other background knowledge nodes. C_{in}^{com} is zero in all but configuration. C^{om} and C^{rad} vary across their possible values. The total number of solutions (correct plus incorrect) is higher in the lower part of the table, except for two rows with 7 incorrect and 4 correct solutions: in these rows the parameters make graph radius the dominating cost and optimal fitness is achieved by not using any background knowledge (which is undesired and also leads to bad correctness).

In summary we see that all coefficients of f_{rel} are important, and many parameter values and combinations achieve correct disambiguation with few reasoning result. On the

[3] www.peterschueller.com/winograd/kr14/

C_{in}^{com}	(C_{bg},C_{bg}^{com},C_{bg}^{dis})	C^{om}	C^{rad}	# incorr.	# corr.	% corr.
3	(0,1,-2)	0	-5	0	10	100
3	(0,1,-2)	0	-10	0	10	100
3	(0,1,-2)	-10	-5	0	10	100
3	(0,1,-2)	-10	-10	0	10	100
3	(0,0,-1)	0	-5	0	10	100
1	(0,0,-1)	-10	-5	0	10	100
3	(0,0,-1)	-10	-5	0	10	100
3	(0,0,-1)	-10	-10	0	10	100
2	(-1,0,0)	0	-5	0	10	100
3	(-1,0,0)	0	-5	0	10	100
3	(0,1,-1)	0	-5	0	10	100
— 30 configurations (see text) —				0	11	100
.	.	.	.	...	...	...
0	(0,1,-1)	0	-15	7	4	36
0	(0,1,-1)	0	-10	7	4	36
1	(0,0,0)	-20	0	102	58	36
0	(0,0,0)	0	-15	100	50	33
0	(0,0,0)	0	-5	100	50	33
0	(0,0,0)	0	-10	100	50	33
0	(0,1,-1)	-10	0	114	46	29
0	(0,0,0)	-30	-5	154	24	14

Table 1: Best and worst parameter settings (8 runs per row).

negative side we see that zero values are unfavorable, and that the benefit of (0,1,-1) depends on other parameters.

Conclusion and Future Work

In order to formalize important concepts from Relevance Theory we introduced a formal simplification of Roger Schank's framework for knowledge graphs. We started from this framework because it promised an easy formulation of principles from RT as properties of graphs. Moreover, our knowledge graphs have similar properties to other graphs used for representing knowledge.

Therefore we hope that our work can inspire applications of Relevance Theory also in other graph-based reasoning formalisms.

Negative Results. Preliminary experiments showed that certain graph parameters are not useful: minimizing the number of *bridges* (edges whose removal disconnects the graph) does not yield promising results because the input graph might already be without any bridges. Also, maximizing the number of dependencies per node encourages buildup of additional dependencies between input nodes (think: contextual implications) but this measure can introduce redundancies in the graph that do not violate basic graph conditions and certainly do not follow the intended notion of relevance.

Possible Extensions. An important possible extension of our approach are graphs that act as *constraints*, i.e., they invalidate a reasoning result if a specific part of them *can* match the result graph. Preliminary experiments show that adding only one background knowledge graph with such capabilities increases the correctness of many parameter combinations tested in this work. Due to prohibitively high computation times we conducted only preliminary experiments.

Possible extensions of our relevance fitness function are the Cheeger constant (a graph parameter that measures *bottlenecks* in a graph; it is mainly used in research about social or computer networks) and a variable cost for unused activated graphs, depending on such graphs' size.

Winograd Schemas contain short sentences and they are meant to be interpreted without additional context. Therefore wrt. processing effort we mainly considered effort of assuming additional entities, a characterization that is shared by RT and by Hobbs et al. (1993). Future work should also consider salience of concepts and semantic distance between concepts as factors that contribute to processing effort.

Obtaining Knowledge Graphs. We here considered only knowledge graphs and abstracted from the process of *parsing natural language* to create knowledge graphs — certainly a challenging task by itself. Most existing parsing systems resolve ambiguities during parsing, a notable exception is Minimal Recursion Semantics (Copestake et al. 2005) which inspired equality constraints in this work. For obtaining and processing knowledge graphs several related works exist, e.g., the Boxer system (Bos 2008) which is based on Discourse Representation Theory, work by Baral and Dzifcak (2012) who interpret natural language and inferring unknown annotations using inverse lambda calculus, and the approaches of Chen et al. (2013) who combine existing open knowledge bases in a system for human-robot interaction.

Probabilistic approaches are not included in this work, in a general and robust system such methods will be necessary, and metrics relevant to probabilistic NLP will become relevant for evaluating such a system, for example MUC (Message Understanding Conference) metrics.

Computational Efficiency. We largely disregarded the efficiency of reasoning in our experiments and discussion. Due to the high complexity class of our reasoning method which is a typical guess & check & optimize task, reasoning efficiency will certainly become an important topic in practice. Several possibilities for tuning performance exist, the most obvious being an optimization of the ASP encoding, using general principles as described, e.g., by Balduccini and Lierler (2012). Moreover there is the possibility of using a hybrid reasoning formalism such as HEX-programs (Eiter et al. 2005) that combine ASP with domain-specific C++ code (e.g., for graph processing). Finally it is possible to realize the reasoning method from this paper in a formalism that is independent from ASP.

Incremental Progress. While our experiments indicate that certain parameter combinations lead to correct disambiguation, they are limited because they have been performed on a small set of examples. Levesque, Davis, and Morgenstern (2012) pointed out that *incremental progress* is possible in the WSC, by creating libraries of questions that can be solved by children, up to questions solvable by university-educated people. They also propose using a limited vocabulary, which will make testing more objective, and avoid solutions that are targeted to certain sentences.

For this paper we created a few background knowledge graphs from which several become activated and used in multiple of our test sentences. Nevertheless we are convinced that for achieving truly general results, it will be *nec-*

essary to create larger-scale testsuites on standard vocabularies. (Otherwise background knowledge used in experiments might just be tuned to achieve the right experimental outcome, making the experimental results meaningless.)

Creating such a corpus of Winograd Schema on a limited vocabulary contains several research challenges such as vocabulary selection, building the schemas, and evaluating empirically whether each schema is correctly disambiguated by a significant majority of human subjects.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback, and Roland Kaminski for finishing clingo 4.2 at exactly the right time and for fast reactions to bug reports.

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Balduccini, M., and Lierler, Y. 2012. Practical and Methodological Aspects of the Use of Cutting-Edge ASP Tools. In *International Symposium on Practical Aspects of Declarative Languages (PADL)*, 78–92.

Baral, C., and Dzifcak, J. 2012. Solving Puzzles Described in English by Automated Translation to Answer Set Programming and Learning How to Do that Translation. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 573–577.

Barker, K.; Porter, B.; and Clark, P. 2001. A library of generic concepts for composing knowledge bases. In *International Conference on Knowledge Kapture (K-CAP)*, 14.

Bateman, J. a.; Hois, J.; Ross, R.; and Tenbrink, T. 2010. A linguistic ontology of space for natural language processing. *Artificial Intelligence* 174(14):1027–1071.

Bos, J. 2008. Wide-coverage semantic analysis with Boxer. In *Conference on Semantics in Text Processing*, 277–286.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Comm ACM* 54(12).

Chaudhri, V.; Dinesh, N.; and Inclezan, D. 2013. Three lessons in creating a knowledge base to enable reasoning, explanation and dialog. In *Workshop on Natural Language Processing and Automated Reasoning (NLPAR)*, 7–26.

Chen, X.; Xie, J.; Ji, J.; and Sui, Z. 2013. Toward Open Knowledge Enabling for Human- Robot Interaction. *Journal of Human-Robot Interaction* 1(2):100–117.

Copestake, A.; Flickinger, D.; Pollard, C.; and Sag, I. A. 2005. Minimal Recursion Semantics: An Introduction. *Research on Language and Computation* 3(2-3):281–332.

Delgrande, J. P., and Pelletier, J. 1998. A Formal Analysis of Relevance. *Erkenntnis* 49(2):137–173.

Eiter, T.; Ianni, G.; Schindlauer, R.; and Tompits, H. 2005. A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer-Set Programming. In *International Joint Conference on Artificial Intelligence*, 90–96.

Gärdenfors, P. 1978. On the Logic of Relevance. *Synthese* 37:351–367.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2013. Advanced Conflict-Driven Disjunctive Answer Set Solving. In *International Joint Conference on Artificial Intelligence*, 912–918.

Haegeman, L. 1994. *Introduction to Government and Binding Theory*. Blackwell.

Hartshorne, J. K. 2013. What is implicit causality? *Language, Cognition and Neuroscience*.

Hirst, G. 1981. *Anaphora in Natural Language Understanding: A Survey*. Springer-Verlag.

Hobbs, J. R.; Stickel, M.; Martin, P.; and Edwards, D. 1993. Interpretation as abduction. *Artif Intell* 63(1-2):69–142.

Kehler, A.; Kertz, L.; Rohde, H.; and Elman, J. 2008. Coherence and coreference revisited. *Journal of Semantics* 25(1):1–44.

Lee, H.; Peirsman, Y.; Chang, A.; Chambers, N.; Surdeanu, M.; and Jurafsky, D. 2011. Stanfords Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task. In *CoNLL: Shared Task*, 28–34.

Levesque, H.; Davis, E.; and Morgenstern, L. 2012. The Winograd Schema Challenge. In *International Conference on Principles of Knowledge Representation and Reasoning*.

Lifschitz, V. 2008. What Is Answer Set Programming? In *AAAI Conference on Artificial Intelligence*, 1594–1597.

Mann, W. C., and Thompson, S. A. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text* 8(3):243–281.

Marcus, G. 2013. Why can't my computer understand me? *The New Yorker*. August 18.

Padó, S., and Lapata, M. 2007. Dependency-Based Construction of Semantic Space Models. *Computational Linguistics* 33(2):161–199.

Pereira, F. C. 2008. *Creativity and AI: A Conceptual Blending approach*. Walter de Gruyter.

Rahman, A., and Ng, V. 2012. Resolving complex cases of definite pronouns: The Winograd Schema Challenge. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNPL)*, 777–789.

Schank, R. 1972. Conceptual dependency: A theory of natural language understanding. *Cognitive Psychol* 3:552–631.

Schank, R. C. 1980. Language and Memory. *Cognitive Science* 4(3):243–284.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In *International Conference on World Wide Web (WWW)*, 697–706.

Taboada, M., and Mann, W. C. 2006. Rhetorical Structure Theory: looking back and moving ahead. *Discourse Studies* 8(3):423–459.

Wilson, D., and Sperber, D. 2006. Relevance theory. In Horn, L., and Ward, G., eds., *The Handbook of Pragmatics*. Blackwell. 607–632.

Winograd, T. 1972. *Understanding Natural Language*. Academic Press.

Generalized Multi-Context Systems

Shahab Tasharrofi, Eugenia Ternovska
Simon Fraser University, email: {sta44,ter}@cs.sfu.ca

Abstract

Multi-context systems (MCSs) define a versatile framework for integrating and reasoning about knowledge from different (heterogeneous) sources. In MCSs, different types of non-monotonic reasoning are characterized by different semantics such as equilibrium semantics and grounded equilibrium semantics [Brewka and Eiter, 2007].

We introduce a novel semantics of MCSs, a supported equilibrium semantics. Our semantics is based on a new notion of support. The "strength" of supports determines a spectrum of semantics that, in particular, contains the equilibrium and grounded equilibrium semantics. In this way, our supported equilibrium semantics generalizes these previously defined semantics. Moreover, the "strength" of supports gives us a measure to compare different semantics of MCSs.

Introduction

Plato defined knowledge as justified true beliefs but it took thousands of years before this philosophical standpoint was used to form the intuitionistic view of Brouwer and the S4 provability logic of Godel. Since then, the progress towards more justified belief system have accelerated. Artemov (Artemov 1995) introduced Logic of Proofs (LP) as a formalization that internalizes justifications for statements and several justification logics were defined based on LP by Brezhnev (Brezhnev 2001). Knowledge representation and reasoning (KRR) field has also both benefited from and contributed to this direction. Fitting (Fitting 2005) defined an epistemic semantics for LP and Cabalar (Cabalar 2011) used LP to define causal logic programming under stable model semantics. Inspired by the utility that justification systems bring to true logical statements in general and to non-monotonic logic in particular, this paper uses justifications to remove unintended models of multi-context systems (MCSs) (Brewka and Eiter 2007) in a new semantics that we call *supported*. A MCS is a collection of contexts that are linked using bridge rules. Each context has its own way of representing knowledge (i.e., its own syntax and semantics). Bridge rules define how knowledge can be transferred between contexts. In MCSs, a model has the form of a collection of belief sets (called a *belief state*). The semantics of MCSs that inters us here are as follows.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

1. *Equilibrium semantics (ES)* defines intended models as exactly those belief states that, if viewed operationally, remain unchanged after first applying bridge rules and then applying contexts, hence the name of an equilibrium.
2. *Minimal equilibrium semantics (MES)* defines intended models as those equilibriums that are also minimal.
3. *Grounded equilibrium semantics (GES)* defines intended models as the minimal equilibriums of a positive MCS obtained by *reducing* the original MCS. Reducing MCSs is similar (both methodically and intent-wise) to the procedure Gelfond and Lifschitz (Gelfond and Lifschitz 1988) use to define stable model semantics. Unfortunately, this also means that GES is not applicable to MCSs with non-reducible logics, that is logics for which the reduction procedure cannot be applied.

The above semantics are motivated by everyday reasoning about a collection of contexts or agents. In MCSs, some knowledge is shared between different knowledge bases, while some knowledge is kept private/confidential.

Note that that justifications and, in particular, avoiding self-justifications was the main motivation behind the introduction of grounded equilibrium semantics for MCSs (Brewka and Eiter 2007). However, grounded equilibrium semantics (GES) is defined over MCSs in which all contexts are reducible. Thus, even one non-reducible context is enough to render GES non-applicable. Below, example 1 demonstrates this point.

The main goal of this work is to extend MCSs with the notions of justifications and support so that

- a spectrum of semantics for MCSs is obtained by varying one parameter, the support;
- the existing semantics of MCSs are in that spectrum.

We relate variations in support with the level of selectivity of the semantics, and argue that intermediate points in the spectrum are just as useful as the previously known semantics, – ES, MES, GES. We also show the usefulness of our new justification-aware semantics in diagnosis of MCSs.

Recall that normal answer set programs are sets of rules of form $h \leftarrow b_1, \cdots, b_k, \textbf{not}\ b_{k+1}, \cdots, \textbf{not}\ b_n$ such that $h, b_1, \cdots, b_n$ are first order literals (i.e., negated or non-negated atoms with variables). Also, recall that S is a stable model of a normal answer set program P if and only if S is consistent and deductively closed under the positive program obtained by reducing P according to S.

Example 1. *Let $M := (C_1, C_2, C_3)$ be*[1] *a multi-context system with $C_i := (L_i, kb_i, br_i)$ (for $i \in \{1, 2, 3\}$), L_1, L_2 be the logic of normal answer set programs under stable model semantics, L_3 be first-order logic (with $BS_3 := \{\{\}, \{comedian(jk)\}\}$ as possible belief sets), and kb_i, br_i be as follows:*

$$kb_1 := \left\{ \begin{array}{l} forbes400(bg). \\ wealthy(X) \leftarrow forbes400(X). \\ wealthy(X) \leftarrow celebrity(X). \end{array} \right\}$$
$$br_1 := \{celbrity(X) \leftarrow C_2 : famous(X).\}$$
$$kb_2 := \left\{ \begin{array}{l} actor(bp). \\ famous(X) \leftarrow actor(X). \end{array} \right\}$$
$$br_2 := \left\{ \begin{array}{l} famous(X) \leftarrow C_1 : wealthy(X). \\ famous(X) \leftarrow C_3 : comedian(X). \end{array} \right\},$$
$$kb_3 := \{\}$$
$$br_3 := \{\bot \leftarrow C_3 : comedian(X), \textbf{not}\ C_2 : famous(X).\}$$

with X ranging over the four possibilities of "bg" (stands for Bill Gates), "bp" (stands for Brad Pitt), "jk" (stands for Jimmy Kimmel), and "aj" (stands for Average Joe).

Note that MCS M can only use the equilibrium semantics because context C_3 is not reducible. According to equilibrium semantics, M has equilibriums $S_1, \cdots, S_6$ as follows:

for $i \in \{1, \cdots, 6\} : S_i := (bs_1^i, bs_2^i, bs_3^i)$ where,

$$bs_1^1 := \left\{ \begin{array}{l} forbes400(bg), wealthy(bg), wealthy(bp), \\ celebrity(bg), celebrity(bp) \end{array} \right\},$$
$$bs_1^2 := bs_1^1 \cup \{wealthy(aj), celebrity(aj)\},$$
$$bs_1^3 := bs_1^4 := bs_1^1 \cup \{wealthy(jk), celebrity(jk)\},$$
$$bs_1^5 := bs_1^6 := bs_1^2 \cup bs_1^3,$$

$$bs_2^1 := \{actor(bp), famous(bg), famous(bp)\},$$
$$bs_2^2 := bs_2^1 \cup \{famous(aj)\},$$
$$bs_2^3 := bs_2^4 := bs_2^1 \cup \{famous(jk)\},$$
$$bs_2^5 := bs_2^6 := bs_2^2 \cup bs_2^3,$$

$$bs_3^1 := bs_3^2 := bs_3^3 := bs_3^5 := \{\},$$
$$bs_3^4 := bs_3^6 := \{comedian(jk)\}.$$

If the names of this example are to be taken literally, among all the six equilibrium models above, only S_4 is a reasonable belief state. This is because, first, "Average Joe," by definition, is not famous, wealthy, or a celebrity, and, second, "Jimmy Kimmel" is a famous comedian and a wealthy celebrity.

Note that, in Example 1, grounded equilibrium semantics is not applicable because context C_3 uses the non-reducible first-order logic. However, since kb_3 is empty, one might wonder what would have happened if L_3 was a reducible logic. Example 2 considers that situation.

Example 2. *In Example 1, let M be as before except that L_3 is the reducible logic of normal answer set programs under stable model semantics. Then, equilibrium models of M are as before (i.e., $S_1, \cdots, S_6$) but, now, M also has exactly one grounded equilibrium: S_1.*

[1] We use := to represent "denotes" or "equals by definition".

Note that, as we discussed before, in Example 1, S_4 is the only reasonable equilibrium model. However, as we saw in Examples 1 and 2, neither the equilibrium semantics nor the grounded equilibrium semantics can capture the set of intended models. We saw that equilibrium semantics accepts too many equilibriums, i.e., all of $S_1, \cdots, S_6$, and grounded equilibrium semantics (even if definable) rejects our intended equilibrium model S_4.

Summary of the Paper – First, this paper defines a novel semantics for MCSs that we call supported equilibrium semantics. We show that our semantics generalizes both the equilibrium semantics and the grounded equilibrium semantics under a natural instantiation of justification functions for contexts. For instance, we show that, for our running Example 1, depending on how justification functions are instantiated, our supported equilibrium semantics can either work similar to equilibrium semantics and accept all equilibrium $S_1, \cdots, S_6$, or it can work similar to grounded equilibrium semantics and accept only S_1.

Second, this paper also shows that our supported equilibrium semantics characterizes many interesting cases that cannot be captured by either the equilibrium semantics or by the grounded equilibrium semantics. For instance, we show that under a very natural and easy instantiation of justification functions, our supported equilibrium semantics accepts both equilibrium models S_1 and S_4 of MCS M in Example 1. That is, using such natural justifications, supported equilibrium semantics is neither as relaxed as equilibrium semantics that accepts all $S_1, \cdots, S_6$, nor as restrictive as grounded equilibrium semantics that rejects our intended equilibrium model S_4. In this sense, supported equilibrium semantics properly generalizes its predecessors and better captures our intended equilibrium models.

Finally, the last section of this paper is dedicated to a short discussion on a plausible application of supported equilibrium semantics to detect and repair faulty MCSs. There, we discuss how supported equilibrium semantics helps us to remove equilibrium model S_1 of MCS M from Example 1. This type of debugging and repairing has not previously been possible in MCSs, and its future formal investigation will help MCSs be more suited to modelling practical KRR applications.

Contributions

Our contributions in this paper are as follows:

Supported Equilibrium Semantics – We introduce a novel semantics for multi-context systems by adding the concept of support to all logics and then extending the semantics of equilibrium models so that the concept of support is respected when choosing intended equilibria of a MCS.

Unifying Equilibrium Semantics with Grounded Equilibrium Semantics – We show that our new semantics of supported equilibria naturally extends both the equilibrium semantics and the grounded equilibrium semantics. That is, we show that, by carefully choosing supports of logics, both equilibrium semantics and grounded equilibrium semantics become special cases of supported equilibrium semantics.

Broadening Applications of MCSs – Our supported equilibrium semantics extends the applicability of MCSs to ap-

plications in which some/all of justifications cannot be revealed due to privacy, confidentiality or other reasons. This is achieved by allowing contexts to justify their beliefs at varying degrees, and by designing a semantics that works independently of how detailed justifications are. Of course, the more justifications are provided, the better our semantics becomes.

Better Diagnosis and Repair – We show that supported equilibrium semantics paves the way for better diagnoses and repairs in a faulty MCS.

Background

This section briefly reviews multi-context systems using the exposition of (Fink, Ghionna, and Weinzierl 2011) and (Brewka and Eiter 2007). We also use the two following notations here and in other places throughout this paper:

Notation 1 (Negating a Set). *For a set of belief literals X, we use "***not** *X" to denote a set that contains the negation of literals in X, i.e.,* **not** $X := \{\textbf{not}\ b \mid b \in X\}$.

Notation 2 (Un-pairing). *For a pair $P := (X, Y)$, we use* $\mathrm{fst}(P)$ *to denote X and* $\mathrm{snd}(P)$ *to denote Y.*

In MCSs (Brewka and Eiter 2007), a *logic* L is a triple $L := \langle KB_L, BS_L, ACC_L\rangle$, where KB_L is a set of knowledge bases (syntactic part of L), BS_L is a set of belief sets (semantic part of L), and $ACC_L : KB_L \mapsto 2^{BS_L}$ maps each knowledge base to a set of acceptable belief sets (the semantics of L). A *multi-context system MCS* $:= (C_1, \cdots, C_n)$ is a collection of contexts $C_i := (L_i, kb_i, br_i)$ with logic L_i, knowledge base $kb_i \in KB_{L_i}$ and bridge rules br_i. In MCSs, bridge rule $r \in br_i$ has the following form:

$$(i:s) \quad \leftarrow \quad (c_1:p_1), \cdots, (c_j:p_j), \ \textbf{not}\ (c_{j+1}:p_{j+1}), \cdots, \textbf{not}\ (c_m:p_m). \tag{1}$$

where $hd(r) := s$; $body^+(r) := \{(c_k : p_k) \mid 1 \le k \le j\}$; $body^-(r) := \{(c_k : p_k) \mid j+1 \le k \le m\}$; and, $body(r) := body^+(r) \cup (\textbf{not}\ body^-(r))$.

A *belief state* $S := (S_1, ..., S_n)$ is a collection of belief sets, i.e., $S_i \in BS_{L_i}$. Also, we use $X(S)$ for the disjoint union of beliefs in all S_i's, i.e., $X(S) := \{(i : b) \mid 1 \le i \le n \text{ and } b \in S_i\}$. A bridge rule r of form (1) is *applicable* wrt. S, denoted by $S \models body(r)$, iff $p_l \in S_{c_l}$ for $1 \le l \le j$ and $p_l \notin S_{c_l}$ for $j < l \le m$. We define $app_i(S) := \{hd(r) \mid r \in br_i \wedge S \models body(r)\}$ to obtain heads of all applicable bridge rules of context C_i. Belief state S is an *equilibrium* of *MCS* if, for all i, $S_i \in ACC_{L_i}(kb_i \cup app_i(S))$.

Logic L is *monotone* if, for all $kb, kb' \in KB_L$, (1) $ACC(kb)$ is a singleton set $\{S\}$, and, (2) if $ACC(kb) = \{S\}$, $ACC(kb') = \{S'\}$, and $kb \subseteq kb'$ then $S \subseteq S'$. Also, L is *reducible* if (1) subset $KB^*_L \subseteq KB_L$ exists s.t. $L^* := \langle KB^*_L, BS_L, ACC_L\rangle$ is monotone, and, (2) reduction function $red_L : KB_L \times BS_L \mapsto KB^*_L$ exists s.t. a. $red_L(kb, S) = kb$ if $kb \in KB^*_L$, b. $red_L(kb, S') \subseteq red_L(kb, S)$ if $S \subseteq S'$, and, c. $S \in ACC_L(k)$ iff $ACC_L(red_L(k, S)) = \{S\}$.*Context* $C := (L, kb, br)$ *is reducible* if L is reducible and $red_L(kb \cup H, S) = red_L(kb, S) \cup H$ for all $H \subseteq \{hd(r) \mid r \in br\}$. MCS M is reducible if all of its contexts are reducible.

A reducible MCS $M := (C_1, \ldots, C_n)$ is *definite* if all bridge rules r of M are positive, i.e., $body^-(r) = \emptyset$, and, for all i, $kb_i \in KB^*_{L_i}$. Definite MCSs guarantee monotonic inference and, thus, always have a unique minimal equilibrium (Brewka and Eiter 2007). Also, for reducible MCS $M := (C_1, \cdots, C_n)$, reduction of M under belief state S, denoted by M^S, is a definite MCS $M' := (C^S_1, \cdots, C^S_n)$ where $C^S_i := (L_i, red_i(kb_i, S_i), br^S_i)$ and $br^S_i := \{hd(r) \leftarrow body^+(r) \mid r \in br_i \text{ and } S \models body(r)\}$. Finally, S is a *grounded equilibrium* of reducible MCS M if S is the unique minimal equilibrium of the M^S. For non-reducible MCSs, grounded equilibria are not defined.

Note that other semantics for MCSs and their extensions (such as managed MCSs (Brewka et al. 2011)) exist, but are not considered here because they are not relevant to this paper.

Generalized MCSs

Examples 1 and 2 showed a case where both equilibrium and grounded equilibrium semantics fail to capture our intended models. This section first defines justification functions that justify beliefs of an acceptable belief set with other beliefs of that belief set and formulas in the knowledge base. For example, the belief that a product is unavailable for sale is justified by a combination of other beliefs and formulas as follows. Among the beliefs that contribute towards justifying unavailability of this product, we can mention the belief that all instance of this product in the store are marked as "sold" and that this product is also unavailable in the warehouse. Moreover, a formula also contributes as a justification of unavailability of this product: it says that if a product is available for sale, then it should be either in the warehouse or in the store without being labeled as "sold."

Secondly, in this section, we lift the notion of justification from a particular acceptable belief set in a particular logic to contexts in general, and then to multi-context systems. We show the naturality of our definitions using our running example and leave the formal expressiveness results to the next section.

Support for Logics

In order to define supported equilibrium semantics, we need to introduce the concept of support at the level of logics. To this end, we define *justifications* for acceptable belief sets and, then, extend them to *support for logics*. While we use very natural definitions, we also heavily use our running example to give more intuitions about our definitions.

Consider logic $L := \langle KB, BS, ACC\rangle$, belief set $bs \in BS$, and knowledge base $kb \in KB$ such that bs is an acceptable belief set for kb in L. By a *justification* for bs, we mean a possible explanation of why belief in bs are believed.

Definition 1 (Justification). *Let $L := \langle KB, BS, ACC\rangle$, $kb \in KB$ and $bs \in ACC(kb)$. Then, function $j : bs \mapsto (\mathcal{P}(bs) \times \mathcal{P}(kb))$ is called a* justification *for bs if j is non-circular, i.e., a well-ordering $<_{bs}$ on beliefs in bs exists such that, for all $b, b' \in bs$ with $b \in \mathrm{fst}(j(b'))$, we have $b <_{bs} b'$.*

A justification function as in Definition 1 provides possible explanations for belief sets in a non-circular way. In the

following, Example 3 shows a natural justification function in the context of our running example.

Example 3. *Consider multi-context system M from Example 1 and equilibrium models $S_1, \cdots, S_6$. By definition of equilibrium models, we know that, for $i \in \{1,2,3\}$ and $j \in \{1,\cdots,6\}$, belief set bs_i^j is an acceptable belief set for knowledge base $kb_i \cup app_i(S_j)$, i.e., $bs_i^j \in ACC_i(kb_i \cup app_i(S_j))$. In this example, we give two natural justification functions j_1 and j_2 for two belief sets bs_1^1 and bs_1^2 respectively. Note that, by Definition 1, each j_i (non-circularly) maps bs_1^i to a pair consisting of a subset of bs_1^i and a subset of $kb_1 \cup app_1(S_i)$, i.e., $j_i : bs_1^i \mapsto (\mathcal{P}(bs_1^i) \times \mathcal{P}(kb_1 \cup app_1(S_i)))$. Our j_i's are as follows:*

$$
\begin{aligned}
&j_{1,2}(forbes400(bg)) := (\{\}, \{forbes400(bg)\}),\\
&j_{1,2}(wealthy(bg)) := (\{forbes400(bg)\},\\
&\qquad \{wealthy(bg) \leftarrow forbes400(bg)\}),\\
&j_{1,2}(wealthy(bp)) := (\{celebrity(bp)\},\\
&\qquad \{wealthy(bp) \leftarrow celebrity(bp)\}),\\
&j_2(wealthy(aj)) := (\{celebrity(aj)\},\\
&\qquad \{wealthy(aj) \leftarrow celebrity(aj)\}),\\
&j_{1,2}(celebrity(bg)) := (\{\}, \{celebrity(bg)\}),\\
&j_{1,2}(celebrity(bp)) := (\{\}, \{celebrity(bp)\}),\\
&j_2(celebrity(aj)) := (\{\}, \{celebrity(aj)\}).
\end{aligned}
$$

Note that j_1 and j_2 agree on their shared domain, i.e., bs_1^1 and that they both have very intuitive meanings. For instance, $j_1(forbes400(bg)) = (\{\}, \{forbes400(bg)\})$ means that, our belief in $forbes400(bg)$ is independent of all other beliefs and is justified only by a fact from the knowledge base, i.e., $forbes400(bg)$. It is noteworthy that, here, $forbes400(bg)$ appears once as a belief and another time as a formula. Thus, $forbes400(bg)$ does not constitute a self-justification because, for self-justifications to occur, a belief should (directly or indirectly) depend on itself (as a belief).

In Example 3, justification coincides with the consequence relation. That is, if $j_i(b) = (bs, kb)$, $bs \subseteq bs'$, $kb \subseteq kb'$, and $bs' \in ACC_1(kb')$ then $b \in bs'$. However, we want to emphasize that, despite the proximity of the two notions of consequence and justification, the latter is much more flexible than the former. Therefore, as shown later in this paper, the same logic can have many different justification functions (unlike consequence relation which is closely tied to the semantics of a logic). Moreover, since justifications are more flexible, as this paper shows, they can be used to characterize a range of different semantics for the same MCS (which would have been impossible using consequence relation because of their rigidity).

Now, we use justifications to define the support for a logic. Intuitively, $Sup_L(kb, bs)$ denotes a (usually non-exhaustive) set of possible *justifications* for belief set bs.

Definition 2 (Support for a Logic L). *Let $L := \langle KB_L, BS_L, ACC_L\rangle$, $kb \in KB_L$ and $bs \in BS_L$. Then, $Sup_L(kb, bs)$ is a set of justifications for bs according to kb such that if $bs \notin ACC_L(kb)$ then $Sup_L(kb, bs) = \emptyset$.*

Example 4 applies Definition 2 to our running example:

Example 4. *Continuing Example 1, we define Sup_{L_1} to be such that $Sup_{L_1}(kb_1, bs_1^2) = \emptyset$ (because $bs_1^2 \notin ACC_1(kb_1)$) and $Sup_{L_1}(kb_1 \cup app_1(S_2), bs_1^1) = \{j_2, j_3\}$ where j_2 is from Example 3 and $j_3 : bs_1^2 \mapsto (\mathcal{P}(bs_1^2) \times \mathcal{P}(kb_1 \cup app_1(S_2)))$ is as below:*

$$
\begin{aligned}
&j_3(wealthy(bg)) := (\{celebrity(bg)\},\\
&\qquad \{wealthy(bg) \leftarrow celebrity(bg)\}),\\
&j_3(b) := j_2(b) \quad (\textit{for } b \in bs_1^2 \setminus \{wealthy(bg)\}).
\end{aligned}
$$

As discussed before, justifications do not uniquely correspond to a logic L of a context. Moreover, as seen in Example 4, different justification functions contribute towards defining different supports for a logic. Among all possible supports for a logic, two trivial but important ones are the *unit support* and the *empty support* (defined below). Unit and empty supports are definable for all logics and, as will be seen later on, they respectively form the minimum and the maximum of a lattice on supports for a logic.

Definition 3 (Unit and Empty Supports). *Consider logic L and two supports uSup_L, eSup_L for L such that, for all kb and bs: (1) $\mathrm{eSup}_L(kb, bs) = \emptyset$, and, (2) $\mathrm{uSup}_L(kb, bs) = \{u^{bs}\}$ where, for all $b \in bs$, $u^{bs}(b) = (\{\}, \{\})$. Also, we call uSup_L and eSup_L respectively as the* unit support for L *and the* empty support for L.

Note that, neither the unit support nor the empty support for a logic does not provide any information about the internal knowledge base. As we show later on, the unit and empty support for a logic define the two ends of an spectrum in which the unit support is the most relaxed support for a logic and empty support is the most rigid support for a logic. In other words, we show later on in this paper that (1) unit support for a logic corresponds to the case where all equilibriums are supported, and, (2) empty support corresponds to the case that no equilibriums is supported.

While empty and unit supports are definable for all logics, they do not give us any insight into why something is (or is not) believed. Thus, one might wonder if more insightful notions of support can be developed for interesting KR logical frameworks. Here, we emphasize that, for many interesting logical frameworks, such as many non-monotonic logics, a more rigid notion of support has already been developed (Lifschitz 2010; Tasharrofi 2013). The following example uses a well-established notion of support for normal logic programs in combination with MCS M from Example 1 to show how supports provide insight into a knowledge base.

Example 5. *Consider context C_1 of Example 1. Since kb_1 is a normal logic program under answer set semantics, one can use a support for logic L of C_1 that is based on the notion of support as defined in (Lifschitz 2010). Using this support function, both justification functions j_1 and j_3 from Examples 3 and 4 belong to $Sup_L(bs_1^1, kb_1 \cup app_1(S_1))$.*

Support for Contexts

Using the definition of supports on the level of logic, we define support on the level of contexts as in Definition 4 that follows. Unlike supports for logics that worked with syntactic (knowledge base formulas) and semantic (beliefs) objects simultaneously, in Definition 4, supports at the level of contexts work solely on semantic objects (beliefs).

Definition 4 (Support for Contexts). *Consider MCS $M := (C_1, \cdots, C_n)$, its context $C_i := \langle L_i, kb_i, br_i \rangle$, and its belief state $S := (S_1, \cdots, S_n)$. Also, let Sup_{L_i} be the support function for logic L_i and recall that $X(S) := \{(i : b) \mid b \in S_i\}$. Support of belief set S_i (from context C_i) under S, denoted by Sup_i^S, is the set of functions $f : S_i \mapsto \mathcal{P}(X(S))$ that are computed by taking a function $g \in Sup_{L_i}(kb_i \cup app_i(S), S_i)$ and tracing the reason for the inclusion of the knowledge that comes from bridge rules. More formally, a function f is included in Sup_i^S if and only if functions $g \in Sup_{L_i}(kb_i \cup app_i(S), S_i)$ and $R : S_i \mapsto \mathcal{P}(br_i)$ exist such that, for all $b \in S_i$, we have:*

$$f(b) := \{i : b' \mid b' \in \mathrm{fst}(g(b))\} \cup \bigcup_{r \in R(b)} body^+(r).$$

and $R(b) \subseteq \{r \mid r \in br_i \text{ and } S \models body(r)\}$ is a minimal subset of applicable bridge rules that justifies the knowledge that comes from bridge rules, i.e., for all formulas $k \in (\mathrm{snd}(g(b)) \setminus kb)$, a rule $r \in R(b)$ exists with $hd(r) = k$.

Note that, in Definition 4, if $\mathrm{snd}(g(b)) \setminus kb = \emptyset$ for some belief b, then $R(b) = \emptyset$. Intuitively, it means that support from other contexts is required only when existing knowledge of a context is not sufficient for supporting a belief. Following example applies the notion of support at the context-level to our running Example 1.

Example 6. *Consider MCS M of Example 1 and equilibrium model S_2 of M. Also, assume that support for logic of C_1 is as in Example 5. Using that support, let us compute one of the functions $f \in Sup_1^{S_2}$. According to Definition 4, such a function f supports beliefs in bs_1^2. Also, by Definition 4, to construct such a function f, we need function $g \in Sup_{L_1}(kb_1 \cup app_1(S_2), bs_1^2)$. By Example 4, we know $Sup_{L_1}(kb_1 \cup app_1(S_2), bs_1^2) = \{j_2, j_3\}$, thus either $g = j_2$ or $g = j_3$. In this example, we take $g = j_2$.*

By Definition 4, to construct f, formulas that are justified (generated) by bridge rules should be justified. In our example, these are formulas of form $celebrity(X)$. Fortunately, each of these formulas appear as the head of has exactly one bridge rule. Therefore, the minimal set $R(b)$ in the case of each of these formulas contains exactly one rule with that belief in its head. For example, when $b = celebrity(aj)$, $R(b)$ only contains rule "$celebrity(aj) \leftarrow C_2 : famous(aj)$". Thus, f is as follows:

$$f(celebrity(X)) = \{2 : famous(X)\} \text{ (for all } X),$$
$$f(wealthy(X)) = \{1 : celebrity(X)\} \text{ (for } X \neq bg),$$
$$f(wealthy(bg)) = \{1 : forbes400(bg)\},$$
$$f(forbes400(bg)) = \emptyset.$$

The numbers 1 and 2 in the function f above refer to contexts C_1 and C_2 respectively.

Example 6 shows how support functions can be extended beyond the boundaries of knowledge bases and belief sets of a logic and into the beliefs from the belief sets of other contexts. According to Definition 4 and as shown in Example 6, this task is achieved by following bridge rules of a context.

Supported Equilibrium Semantics

Now, we use supports at the level of contexts to define supported equilibrium semantics for MCSs. In the following, Definition 5 gives our main notion of a *supported equilibrium*. Informally speaking, a belief state is called a supported equilibrium if all beliefs are well-justified, i.e., they are justified and nothing justifies itself (either directly or indirectly). Similar to Definition 1, self-justifications are avoided by requiring the existence of a well-ordering on the beliefs.

Definition 5 (Supported Equilibrium). *A belief state $S := (S_1, \cdots, S_n)$ of MCS is a* supported equilibrium w.r.t. *$(Sup_{L_1}, \cdots, Sup_{L_n})$ if functions $f_1 \in Sup_1^S, \cdots, f_n \in Sup_n^S$ and well-founded strict partial ordering $<$ on $X(S)$ exist s.t. if $p \in S_i$ and $(j : q) \in f_i(p)$ then $(j : q) < (i : p)$.*

First, note that Definition 5 does not put any special requirement on contexts and works for all contexts and all supports. Therefore, unlike grounded equilibrium semantics of (Brewka and Eiter 2007), introspection in supported equilibrium semantics does not come at the cost of excluding non-reducible contexts.

Second, note that Definition 5 tests a belief state for being supported but not for being an equilibrium. So, one might reasonably suspect that a belief state S might exist such that S is a supported equilibrium (according to Definition 5) but not an equilibrium (according to the original definition of equilibrium semantics (Brewka and Eiter 2007)). However, the following Theorem 1 states that if S is a supported equilibrium then it has to be an equilibrium as well. Therefore, the term "supported equilibrium" is indeed an appropriate and reasonable name for belief states that satisfy the condition of Definition 5.

Theorem 1 (Supported Equilibria $\subseteq$ Equilibria). *Let $M := (C_1, \cdots, C_n)$ and also let $S := (S_1, \cdots, S_n)$ be a supported equilibrium of M w.r.t. $(Sup_{L_1}, \cdots, Sup_{L_n})$. Then, S is also an equilibrium of M.*

Proof. Assume that S is not an equilibrium of M. Then, context C_i should exist such that $S_i \notin ACC(kb_i \cup app_i(S))$. Therefore, by Definition 2, we know that $Sup_{L_i}(kb_i \cup app_i(S), S_i) = \emptyset$. Thus, by Definition 4, $Sup_i^{L_i}$ is also empty. Hence, by Definition 5, S cannot be a supported equilibrium of M which contradicts our assumption. So, S has to be an equilibrium of M. □

Now, let us return to our running example and see if supported equilibrium semantics can be introspective enough to reject equilibrium model S_2 of Example 1 as not supported.

Example 7. *Consider MCS M and equilibrium model S_2 of Example 1. In Example 6, we saw that $Sup_1^{L_1}$ contains two support functions for belief set bs_1^2 of S_2. We also computed support function $f \in Sup_1^{L_1}$. For S_2 to be a supported equilibrium, by Definition 5, functions $f_i \in Sup_i^{L_i}$ (for $i \in \{1, 2, 3\}$) should be found so that beliefs in bs_i^2 ($i \in \{1, 2, 3\}$) are justified and nothing justifies itself. In this example, we show that, indeed, if f_1 is the support function f from Example 6, then a self-justification is inevitable.*

In order to do so, we take f_1 to be function f from Example 6 and show that ordering $<$ cannot exist. We know that $(1 : celebrity(aj)) \in f_1(wealthy(aj))$ and $(2 : famous(aj)) \in f_1(celebrity(aj))$. Thus, ordering $<$ should satisfy $(2 : famous(aj)) < (1 : celebrity(aj)) < (1 : wealthy(aj))$. Now, we look at the possible supports for belief $famous(aj)$ in bs_2^2. According to C_2, belief $famous(X)$ can have three possible supports: $actor(X)$, $comedian(X)$ or $wealthy(X)$. However, for $X = aj$, the only possibility is $wealthy(X)$. Thus, inevitably, we have $(1 : wealthy(aj)) \in f_2(famous(aj))$ and, therefore, ordering $<$ should also satisfy $(1 : wealthy(aj)) < (2 : famous(aj))$ that contradicts the previous constraint on $<$.

Hence, if f_1 is function f from Example 6, ordering $<$ cannot exist and a self-justification inevitably occurs. The reader can check that choosing f_1 differently does not rectify the situation either. So, S_2 is not a supported equilibrium w.r.t. support functions $(Sup_{L_1}, Sup_{L_2}, \mathrm{uSup}_{L_3})$.

Example 7 demonstrates how supported equilibrium semantics avoids undesirable equilibrium models. Moreover, Example 7 shows that supported equilibrium semantics can also single out unfounded beliefs. In Example 7, one can check that all beliefs in S_2 are founded except those about "Average Joe". Even more importantly, Example 7 shows that supported equilibrium semantics allows us to trace back the reason for inclusion of all beliefs in a belief state.

Moreover, recall that grounded equilibrium semantics was not applicable to MCS M from Example 1. Thus, up to now, it was not possible to avoid unintended equilibrium models such as S_2 for MCSs such as M. However, as Example 7 shows, supported equilibrium semantics both (1) applies to M, and, (2) rejects unintended equilibrium S_2 as not supported. Hence, supported equilibrium semantics can indeed express cases that are not expressible using either the equilibrium model semantics or the grounded equilibrium model semantics. It should also be noted that Example 7 concerned a very simple MCS in which all contexts except one were reducible. However, as shown in the next section, the cases that supported equilibrium semantics can express (but equilibrium semantics or grounded equilibrium semantics cannot) are not limited to these easy cases. In the next section, we show that supported equilibrium semantics generalizes and unifies a spectrum of different semantics that can be ordered according to their selectivity.

Degree of Selectivity in Supports

In the previous sections, we defined supported equilibrium semantics and showed that all supported equilibriums are indeed also an equilibrium while vice versa is not necessarily true. That is, as shown in Example 7, there exist equilibriums that are not supported. Moreover, in various parts of this paper, we noted that supported equilibrium semantics is a unifying semantics that can define a spectrum of different semantics ordered according to their flexibility and/or selectivity. In this section, we want to further elaborate on the relationship between different semantics of a MCS.

In this section, we partially order different supports for logics and show that this ordering directly corresponds to the degree of selectivity in supported equilibrium semantics. That is, if support S is less than support S' (according to the ordering in this section), then supported semantics w.r.t. S' is more selective than supported semantics w.r.t. S.

We first define an ordering over justification functions:

Definition 6 (Ordered Justifications). *Let bs be a belief set for a logic L and j_1, j_2 be two justification functions for bs. Then, we say that j_1* is less selective than *j_2, denoted by $j_1 \leq j_2$, if, for all $b \in bs$, we have:*

$$\mathrm{fst}(j_1(b)) \subseteq \mathrm{fst}(j_2(b)) \text{ and } \mathrm{snd}(j_1(b)) \subseteq \mathrm{snd}(j_2(b)).$$

Intuitively, $j_1 \leq j_2$ means that j_1 requires less reasons than j_2 for supporting beliefs b in a belief set. Therefore, since supported equilibriums disallow self-justifying loops, requiring more reasons makes such a loop more probable and, so, shrinks the set of supported equilibriums.

Definition 7 (Ordered Supports). *Let $L := \langle KB, BS, ACC\rangle$ be a logic and Sup_L, Sup'_L be two supports for logic L. Then, we say that Sup_L* is less selective than *Sup'_L, denoted by $Sup_L \leq Sup'_L$ if, for all $kb \in KB$, $bs \in BS$, and $j' \in Sup'_L(kb, bs)$, a justification $j \in Sup_L(kb, bs)$ exists such that $j \leq j'$.*

Informally speaking, Definition 7 says that S_1 is a less selective support than S_2 if and only if, for all justification functions in S_2, a less selective justification function in S_1 can always be found. Hence, again, if $S_1 \leq S_2$ then using S_1 leads to less circularity and thus more supported equilibriums. The following theorem formalizes this reasoning:

Theorem 2. *Let $M := (C_1, \cdots, C_n)$ be a MCS and $S := (S_1, \cdots, S_n)$ be a supported equilibrium of M w.r.t. supports $(Sup_1, \cdots, Sup_n)$. Also, let supports $Sup'_1, \cdots, Sup'_n$ be such that $Sup'_i \leq Sup_i$ (for $1 \leq i \leq n$). Then, S is also a supported equilibrium of M w.r.t. supports $(Sup'_1, \cdots, Sup'_n)$.*

Proof. Let ordering $<$ and functions $f_i \in Sup_i^S$ witness S being a supported equilibrium. Since $f_i \in Sup_i^S$, by Definition 4, functions $g_i \in Sup_i(kb_i \cup app_i(S), S_i)$ and $R_i : S_i \mapsto \mathcal{P}(br_i)$ exist such that, for all $b \in S_i$: (1) $R_i(b)$ is a minimal subset of applicable bridge rules in br_i (with respect to S) that satisfies $\{hd(r) \mid r \in R_i(b)\} \supseteq \mathrm{snd}(g_i(b)) \setminus kb_i$, and, (2) $f_i(b) = \mathrm{fst}(g_i(b)) \cup \bigcup_{r \in R_i(b)} body^+(r)$.

Now, since $Sup'_i \leq Sup_i$, functions $g'_i \in Sup'_i(kb_i \cup app_i(S), S_i)$ exist such that $g'_i \leq g_i$. Also, construct R'_i such that $R'_i(b) = \{r \mid r \in R_i(b) \text{ and } hd(r) \in (\mathrm{snd}(g'_i(b)) \setminus kb_i)\}$. It is easy to check that $R'_i(b)$ satisfies minimality conditions of Definition 4 and, so, functions $f'_i \in Sup'^S_i$ exist such that $f'_i(b) = \mathrm{fst}(g'_i(b)) \cup \bigcup_{r \in R'_i(b)} body^+(r)$.

Therefore, in order to prove that S is supported w.r.t. $(Sup'_1, \cdots, Sup'_n)$, we only need to show that functions f'_i are non-circular. Note that, since $R'_i(b) \subseteq R_i(b)$ and $\mathrm{fst}(g'_i(b)) \subseteq \mathrm{fst}(g_i(b))$, we have $f'_i(b) \subseteq f_i(b)$ (for all b). Hence, the same ordering $<$ also shows the non-circularity of functions f'_i because:

$$(j : b') \in f'_i(b) \Rightarrow (j : b') \in f_i(b) \Rightarrow (j : b') < (i : b).$$

□

Theorem 2 shows that our ordering on supports is indeed relevant to our supported equilibrium semantics in the sense that greater supports are related to more selective semantics. The following proposition shows that our ordering relation on supports has a minimum and a maximum that respectively coincides with our unit and empty supports.

Proposition 1 (Least and Greatest Supports). *For all logics L and supports Sup_L, we have:* $\mathrm{uSup}_L \leq Sup_L \leq \mathrm{eSup}_L$.

Proof. Showing $Sup_L \leq \mathrm{eSup}_L$ is easy because $\mathrm{eSup}_L(kb, bs) = \emptyset$. Showing $\mathrm{uSup}_L \leq Sup_L$ is also easy because function $u^{bs} \in \mathrm{uSup}_L(kb, bs)$ is less than all $j \in Sup_L(kb, bs)$. This is because $\mathrm{fst}(u^{bs}(b)) = \emptyset \subseteq \mathrm{fst}(j(b))$ and $\mathrm{snd}(u^{bs}(b)) = \emptyset \subseteq \mathrm{snd}(j(b))$. □

According to Proposition 1, unit support is the most flexible support. In the next section, we show that if all logics use the unit support then supported equilibrium semantics coincides with equilibrium semantics. That is, equilibrium semantics is, in fact, a semantics in which beliefs are believed without really knowing why. On the other hand, we will also show in the next section that grounded equilibrium semantics is a particular case of supported equilibrium semantics in which having detailed justifications for beliefs are extremely important.

Generalizing Normal and Grounded Equilibrium Semantics

One of the promises of this paper was that our new supported equilibrium semantics generalizes both of the original semantics for multi-context systems, i.e., the equilibrium semantics and the grounded equilibrium semantics. In this section, we give Theorems 3 and 5 that, respectively, prove that equilibrium semantics and grounded equilibrium semantics are both special cases of supported equilibrium semantics (just using different support functions).

Theorem 3. *Let $M := (C_1, \cdots, C_n)$ be a MCS and $S := (S_1, \cdots, S_n)$ be a belief state of M. Then, S is an equilibrium of MCS if and only if S is a supported equilibrum of MCS w.r.t.* $(\mathrm{uSup}_{L_1}, \cdots, \mathrm{uSup}_{L_n})$.

Proof. ($\Leftarrow$) Directly follows Theorem 1.
($\Rightarrow$) Since S is an equilibrium model of M, we have that $S_i \in ACC(kb_i \cup app_i(S))$ for all $1 \leq i \leq n$. Therefore, by Definition 3, we know that $\mathrm{uSup}_{L_i}(kb_i \cup app_i(S), S_i) = \{u^{S_i}\}$. Now, by Definition 4, $Sup_i^S = \{f^{S_i}\}$ where $f^{S_i}(b) := \emptyset$ for all $b \in S_i$. Hence, we take $(f_1, \cdots, f_n) \in (Sup_1^S \times \cdots \times Sup_n^S)$ such that $f_i := f^{S_i}$. Now, by Definition 5, S is a supported equilibrium because functions $f_1, \cdots, f_n$ do not induce any circular justification. □

Theorem 3 shows that supported equilibrium semantics w.r.t. unit supports naturally extends equilibrium semantics. Previously, we discussed that unit support is associated with a complete black-box view of contexts since it does not provide any information about why something is believed. Hence, by Theorem 3, we know that equilibrium semantics indeed corresponds to the view of contexts as black-boxes.

Next, we show that, if support functions are chosen carefully, grounded equilibrium semantics can also be defined in terms of supported equilibrium semantics. In order to achieve our goal, we first characterize the unique minimal equilibrium of definite MCSs and use this definition to characterize the grounded equilibrium semantics in terms of supported equilibrium semantics.

Definition 8 (Monotonicity-based Support). *Let $L := \langle KB, BS, ACC\rangle$ be a monotone logic, $kb \in KB$ be a knowledge base and $bs \in BS$ be the unique acceptable belief set of kb, i.e., $\{bs\} = ACC(kb)$. Also, let $<$ be a total and well-founded ordering on kb. A function $j_< : bs \mapsto (\mathcal{P}(bs) \times \mathcal{P}(kb))$ is said to be a* monotonicity-based justification *according to ordering $<$ if, for all $b \in bs$, we have $j_<(b) := (bs_1, kb_2)$ where kb_1, kb_2, bs_1 and bs_2 are so that:*

- *either $kb_1 = bs_1 = \emptyset$ or $kb_1 \in KB$, $bs_1 \in BS$ and $\{bs_1\} = ACC(kb_1)$,*
- *$kb_2 \in KB$, $bs_2 \in BS$ and $\{bs_2\} = ACC(kb_2)$,*
- *formulas $k_1, k_2 \in kb$ exist such that $k_1 \leq k_2$, $kb_1 = kb_{<k_1}$, $kb_2 = kb_{\leq k_2}$, and, for all k' with $k_1 < k' \leq k_2$, we have $kb_{<k'} \notin KB$,*
- *$b \notin bs_1$ but $b \in bs_2$.*

We also define the monotonicity-based support of L, *denoted by* mSup_L, *as follows:*

If $\{bs\} \neq ACC(kb)$ then $\mathrm{mSup}_L(kb, bs) = \emptyset$;
Otherwise, $\mathrm{mSup}_L(kb, bs)$ is the set of monotonicity-based justifications $j_<$ according to ordering $<$.

In the following, Theorem 4 shows that, for definite multi-context systems, supported equilibrium semantics and grounded equilibrium semantics coincide.

Theorem 4. *Let $M := (C_1, \ldots, C_n)$ be a definite MCS and $S := (S_1, \cdots, S_n)$ be a belief state of M. Then, S is a supported equilibrium of M w.r.t.* $(\mathrm{mSup}_{L_1}, \cdots, \mathrm{mSup}_{L_n})$ *if and only if S is a grounded equilibrium of M.*

Proof. ($\Leftarrow$) Let S be a grounded equilibrium of M. By Proposition 1 of (Brewka and Eiter 2007), we know that $\{S_i\} = ACC_{L_i}(kb_i^\infty)$ where $kb_i^\infty := \bigcup_\alpha kb_i^\alpha$ and kb_i^α is defined as follows:

$$kb_i^\alpha := \begin{cases} kb_i & \text{if } \alpha = 0, \\ kb_i^\beta \cup app_i(E^\beta) & \text{if } \alpha = \beta + 1, \\ \bigcup_{\beta<\alpha} kb_i^\beta & \text{if } \alpha \text{ is a limit ordinal.} \end{cases}$$

where, for all ordinals α, we have $E^\alpha := (E_1^\alpha, \cdots, E_n^\alpha)$ and $\{E_i^\alpha\} = ACC(kb_i^\alpha)$. So, for ordinals α and β, if $\alpha < \beta$ then $E_i^\alpha \subseteq E_i^\beta$ (for all i). Thus, for all $k \in kb_i^\infty$ (respectively, for all $b \in S_i$), by $rank_i^{kb}(k)$ (respectively, by $rank_i^{bs}(b)$), we denote the minimum ordinal α such that $k \in kb_i^\alpha$ (respectively, $b \in E_i^\alpha$). Now, define orderings $<_i$ (for $i \in \{1, \cdots, n\}$) on kb_i^∞ to be any total ordering that respects the ranks of knowledge in kb_i^α's, i.e., for $k_1, k_2 \in kb_i^\infty$, we have $k_1 <_i k_2$ if $rank_i^{kb}(k_1) < rank_i^{kb}(k_2)$.

Next, we show that functions $f_1 \in \mathrm{mSup}_1^S, \cdots, f_n \in \mathrm{mSup}_n^S$ exist such that $f_1, \cdots, f_n$ avoid self-justification. In order to do that, consider support functions $g_i \in \mathrm{mSup}_{L_i}(kb_i^\infty \cup app_i(S), S_i)$ (for $i \in \{1, \cdots, n\}$) that are

generated according to orderings $<_i$ (respectively). Now, construct functions $f_i \in \mathrm{mSup}_i^{L_i}$ using g_i's so that, for $b \in S_i$, we have:

$$f_i(b) := \{(i : b') \mid b' \in \mathrm{fst}(g_i(b))\} \cup \bigcup_{r \in R_b} body^+(r),$$

where R_b is a minimal subset of applicable bridge rules of C_i such that, for each $k \in (\mathrm{snd}(g_i(b)) \setminus kb_i)$, there is a rule $r \in R$ with $hd(r) = k$ and $E_i^{rank_i^{kb}(k)-1} \models body(r)$. Note that $rank_i^{kb}(k) - 1$ is indeed an ordinal because, if $k \notin kb_i$, $rank_i^{kb}(k)$ is always a successor ordinal.

It can be easily checked that, for $i \in \{1, \cdots, n\}$, we have $f_i \in \mathrm{mSup}_i^{L_i}$. We just need to show that ordering $<^*$ exists such that $(i : p) \in f_j(q)$ implies $(i : p) <^* (j : q)$. Define $<^*$ as follows:

$$(i : p) <^* (j : q) \Longleftrightarrow \begin{cases} \text{either } i = j \text{ and } p \in g_i(q), \\ \text{or, } rank_i^{bs}(p) < rank_j^{bs}(q). \end{cases}$$

We leave it to the reader to check that $<^*$ is a well-founded ordering. Now, if $(i : p) \in f_j(q)$, we know that either $i = j$ and $p \in \mathrm{fst}(g_j(q))$ or $(i : p) \in body^+(r)$ for a rule $r \in app_i(S)$ with $hd(r) \in \mathrm{snd}(g_j(q))$. In the former case, by construction of $<^*$, we have $(i : p) <^* (j : q)$ as required. In the latter case, let $\alpha := rank_j^{bs}(q)$. By construction of $\mathrm{snd}(g_j(q))$, we know that α is a successor ordinal and that r is applicable under $E^{\alpha-1}$. Thus, $p \in E_i^{\alpha-1}$. Hence, $rank_i^{bs}(p) \leq \alpha - 1 < \alpha = rank_j^{bs}(q)$ and, so, by construction of $<^*$ and as required, $(i : p) <^* (j : q)$. Hence, S is a supported equilibrium of M.

($\Rightarrow$) Since S is a supported equilibrium, functions $f_i \in \mathrm{mSup}_i^{L_i}$ and well-ordering $<_{X(S)}$ exist such that $(i : p) \in f_j(q)$ implies $(i : p) <_{X(S)} (j : q)$. Also, by Definition 4, we know that each f_i is constructed according to some $g_i \in \mathrm{mSup}_{L_i}(kb_i \cup app_i(S), S_i)$. Also, since M is a definite MCS, it has a unique minimal equilibrium. Let $E := (E_1, \cdots, E_n)$ be that minimal equilibrium.

By Theorem 1, we know that S is an equilibrium of M. Thus, we only need to show that S is minimal. Assuming otherwise means that i and p exist such that $p \in S_i$ but $p \notin E_i$. Choose a minimal such i and p w.r.t. ordering $<_{X(S)}$. By Definition 8, $f_i(p) \supseteq \bigcup_{r \in R} body^+(r)$ where R satisfies $\{hd(r) \mid r \in R\} \supseteq (\mathrm{snd}(g_i(p)) \setminus kb_i)$. Now, since $(i : p)$ is minimal w.r.t. $<_{X(S)}$, if $(j : q) \in f_i(p)$ then $q \in E_j$. Thus, for all $r \in R$, we have $E \models body(r)$ and, so, $app_i(E) \supseteq \{hd(r) \mid r \in R\} \supseteq (\mathrm{snd}(g_i(p)) \setminus kb_i)$. Therefore, $(kb_i \cup app_i(E)) \supseteq \mathrm{snd}(g_i(p))$. Now, assume that bs is the unique acceptable belief set of $\mathrm{snd}(g_i(p))$. By Definition 8, $p \in bs$. Moreover, since L_i is monotone and $\{E_i\} = ACC(kb_i \cup app_i(E))$, we have that $E_i \supseteq bs$ and, henceforth, $p \in E_i$. This contradicts our assumption that $p \notin E_i$. Therefore, S is a minimal equilibrium and, since M has only one minimal equilibrium, $S = E$. □

Theorem 4 shows that, for definite MCS, grounded equilibrium semantics can be defined in terms of supported equilibrium semantics. In the following, we use monotonicity-based support of M and Theorem 4 to show that, indeed, grounded equilibrium semantics can always be defined in terms of supported equilibrium semantics.

Definition 9 (Reducibility-based Supports). *Let $L := \langle KB, BS, ACC\rangle$ be a reducible logic. Also, let $kb \in KB$ and $bs \in ACC(kb)$. Moreover, let $L^* := \langle KB^*, BS, ACC\rangle$ be the monotone part of logic L. The* reducibility-based support of L, *denoted by* $\mathrm{rSup}_L(kb, bs)$, *is the set of functions $f : bs \mapsto (\mathcal{P}(bs) \times \mathcal{P}(kb))$ such that:*

$$\exists f' \in \mathrm{mSup}_{L^*}(red_L(kb, bs), bs) \text{ s.t.} \\ \forall b \in bs : \ f(b) = (\mathrm{fst}(f'(b)), \mathrm{snd}(f'(b)) \cap kb).$$

Needless to say that $\mathrm{rSup}_L(kb, bs) = \emptyset$ *if* $bs \notin ACC(kb)$.

The following lemma suggests that the difference between monotonicity-based supports and reducibility-based supports disappear when we move to the supports to the level of contexts.

Lemma 1. *For reducible MCS $M := (C_1, \cdots, C_n)$ and belief state $S := (S_1, \cdots, S_n)$ of M, let $C_i' := (L_i^*, red_{L_i}(kb_i, S_i), br_i^*)$ where L_i^* is the monotone part of L_i and br_i^* is the reduct of br_i under S. Then, for all i, $\mathrm{mSup}_i^S = \mathrm{rSup}_i^S$.*

Theorem 5. *Let $M := (C_1, \ldots, C_n)$ be a reducible MCS and $S := (S_1, \cdots, S_n)$ be a belief state of M. Then, S is a grounded equilibrium of M if and only if S is a supported equilibrium of M w.r.t. $(\mathrm{rSup}_{L_1}, \cdots, \mathrm{rSup}_{L_n})$.*

Proof. ($\Rightarrow$) Let S be a ground equilibrium of M. Then, S is the unique grounded equilibrium of M^S and, hence, a supported equilibrium of M w.r.t. $(\mathrm{mSup}_{L_1^*}, \cdots, \mathrm{mSup}_{L_n^*})$ (by Theorem 4 and where L_i^*'s are the monotone part of L_i's). Therefore, $f_i^* \in \mathrm{mSup}_i^S$ and strict well-ordering $<$ exist such that $p \in S_i \wedge (j : q) \in f_i(p) \Rightarrow (j : q) < (i : p)$. By Lemma 1, $f_i \in \mathrm{rSup}_i^S$. Thus, S is a supported equilibrium of M w.r.t. $(\mathrm{rSup}_1, \cdots, \mathrm{rSup}_n)$.

($\Leftarrow$) Let S be a supported equilibrium of M. Then, by Lemma 1, S is also a supported equilibrium of M^S (using the same support functions and well-ordering). Therefore, S is the unique grounded equilibrium of M^S (by Theorem 4). Hence, S is a grounded equilibrium of M. □

Theorem 5 shows that supported equilibrium semantics naturally extends grounded equilibrium semantics. We also showed previously in Theorem 3 that supported equilibrium semantics also naturally extends equilibrium semantics of MCSs. Thus, our supported equilibrium semantics indeed extends and unifies both the equilibrium semantics and the grounded equilibrium semantics.

Application: Better Diagnoses

In this section, we discuss one of possible applications of supported equilibrium semantics for MCSs: the detailed diagnosis of MCSs in the presence of supports.

Recall the example of shops and warehouses from the beginning of the section on generalized MCSs. There, a product was believed to be unavailable for sale because (1) the product was believed not to be in the warehouse, (2) all

the available instances of that product in the store were believed to be "sold," and, (3) our knowledge base asserted that a product can be available only if it is available in the warehouse or if it is available in the store and not marked as "sold." Now, consider an extension of this example in which this product is also believed to be on route from factory to warehouse and our knowledge asserts that a product cannot be both unavailable and on route to warehouse. Here, using supports to trace back the reason for conflict, a more meaningful diagnosis can be obtained: to change our knowledge so that we also believe a product is available if it is on route from factory to warehouse. Note that our diagnosis here proposes a change to the knowledge base of a context; a diagnosis that was impossible without using justifications.

In (Bögl et al. 2010) and (Eiter et al. 2010), authors rightly argue that, due to their distributed behavior and complexity, MCSs are prone to two types of errors: those that originate inside a context and those that originate in the interaction between contexts. Moreover, the authors introduce *inconsistency explanations* and *diagnoses* to repair inconsistent MCSs. For MCS M, an inconsistency-explanation is a pair (E_1, E_2) so that $E_1, E_2 \subseteq br_M$ which, informally speaking, means that if all bridge rules in E_1 are kept and all bridge rules in E_2 are non-applicable then M is inconsistent. Thus, (E_1, E_2) explains why M is inconsistent: because bridge rules in E_1 are all applicable or because bridge rules in E_2 are all non-applicable. Also, authors of (Eiter et al. 2010) study the dual notion of diagnoses (D_1, D_2) which means that excluding bridge rules in D_1 and excluding enough conditions from the body of bridge rules in D_2 makes M consistent. Hence, a diagnosis tells us a way to restore the consistency of M.

The authors of (Eiter et al. 2010) study inconsistency explanations and diagnoses and prove their duality. However, diagnoses and inconsistency explanations only take bridge rules into consideration and cannot deal with inconsistencies that arise from errors internal to a knowledge base. However, we now know that supported equilibrium semantics provides the right means to look inside the contexts of a MCS. Therefore, a mistake can be traced back to its source via justifications. In this section, we consider an example of how better repair is achievable through support functions and leave the full treatment of this subject to a future research.

Moreover, and even more importantly, supported equilibrium semantics enables us to focus on incorrectness of MCSs rather than their inconsistency. It should be clear that inconsistency and incorrectness are different notions. For example, consider MCS M from Example 1. M has two equilibrium models and, thus, is consistent. However, M accepts unintended equilibrium model S_2 and, thus, is incorrect. Similarly, one could argue that some inconsistent multi-context systems are indeed correct. For instance, a tour of American cities that passes each city exactly once might not exist. The non-existence of such a path might make a MCS inconsistent but it does not make it incorrect.

Example 8. *Consider MCS M of Example 1 and its equilibrium model S_1. We know that S_1 is a supported equilibrium model of M w.r.t. the same support functions as in Example 7. According to S_1, "Jimmy Kimmel" is not famous, not wealthy and not a celebrity which is contradictory to the real world. So, we want to guarantee that M should assert $celebrity(jk)$. Under this assumption, M is incorrect because it allows S_1 to be supported. Thus, M is consistent but not correct (because it does not represent what it is intended to represent). Now, using support functions, we understand that one possible repair for our MCS is to guarantee in C_3 that "Jimmy Kimmel" is a comedian. In this case, a non-circular chain of justifications would guarantee that $celebrity(jk)$, $wealthy(jk)$ and $famous(jk)$ will all be true. Thus, we would add knowledge $comedian(jk)$ to the knowledge base of C_3 to repair this example and disallow S_1 from being a supported equilibrium.*

Hence, supported equilibrium semantics enables us to focus on correctness (rather than consistency), and better repair MCSs (by proposing internal changes to contexts).

Conclusion and Future Directions

In this paper, we introduced the notion of justifications and supports to MCSs and invented the novel semantics of supported equilibria for MCSs. Moreover, we showed that supported equilibrium semantics is useful in two important ways: (a) from the theoretical aspect, supported semantics generalizes the two main semantics proposed for multi-context systems, i.e., the equilibrium semantics and the grounded equilibrium semantics, and, (b) from the application point of view, supported equilibrium semantics provides the means to look inside the contexts of MCSs and to better diagnose and repair faulty MCSs. Hence, we believe that supported equilibrium semantics is the most suitable semantics available for MCSs.

Moreover, in this paper, we only argued about two very specific supports, i.e., the trivial unit supports and the reducibility-based supports. We also discussed that supported equilibrium semantics allows the inclusion of many more possible supports that, from the viewpoint of selectivity degree, lie between these two semantics. We believe that many other useful supports exist that have not been investigated in this paper. For example, an interesting and unanswered question about supported equilibrium semantics is to find necessary and/or sufficient conditions under which it is guaranteed that supported equilibria of a MCS will form an anti-chain. Answering such a question gives us natural ways to define a semantics similar to stable model semantics for arbitrary new languages.

References

Artemov, S. 1995. Operational modal logic. Technical report, Cornell University.

Bögl, M.; Eiter, T.; Fink, M.; and Schüller, P. 2010. The MCS-IE system for explaining inconsistency in multi-context systems. In Janhunen, T., and Niemelä, I., eds., *Logics in Artificial Intelligence - 12th European Conference, JELIA 2010, Helsinki, Finland, September 13-15, 2010. Proceedings*, volume 6341 of *Lecture Notes in Computer Science*, 356–359. Springer.

Brewka, G., and Eiter, T. 2007. Equilibria in heterogeneous nonmonotonic multi-context systems. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI'07) - Volume 1*, 385–390. AAAI Press.

Brewka, G.; Eiter, T.; Fink, M.; and Weinzierl, A. 2011. Managed multi-context systems. In Walsh, T., ed., *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 786–791. IJCAI/AAAI.

Brezhnev, V. 2001. On the logic of proofs. In Striegnitz, K., ed., *Proceedings of the Sixth ESSLLI Student Session, 13th European Summer School in Logic, Language and Information (ESSLLI'01)*, 35 – 46.

Cabalar, P. 2011. Logic programs and causal proofs. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI.

Eiter, T.; Fink, M.; Schüller, P.; and Weinzierl, A. 2010. Finding explanations of inconsistency in multi-context systems. In Lin, F.; Sattler, U.; and Truszczynski, M., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*. AAAI Press.

Fink, M.; Ghionna, L.; and Weinzierl, A. 2011. Relational information exchange and aggregation in multi-context systems. In Delgrande, J. P., and Faber, W., eds., *Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings*, volume 6645 of *Lecture Notes in Computer Science*, 120–133. Springer.

Fitting, M. 2005. The logic of proofs, semantically. *Annals of Pure and Applied Logic* 132(1):1 – 25.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceeding of the 5th International Conference and Symposium on Logic Programming (ICLP/SLP'88)*, 1070–1080.

Lifschitz, V. 2010. Thirteen definitions of a stable model. In Blass, A.; Dershowitz, N.; and Reisig, W., eds., *Fields of logic and computation*. Berlin, Heidelberg: Springer-Verlag. 488–503.

Tasharrofi, S. 2013. A rational extension of stable model semantics to the full propositional language. In Rossi, F., ed., *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 1118 – 1124. IJCAI.

Qualitative Spatial Representation and Reasoning in Angry Birds: The Extended Rectangle Algebra

Peng Zhang and **Jochen Renz**
Research School of Computer Science
The Australian National University
Canberra, Australia
{u4969566, jochen.renz}@anu.edu.au

Abstract

Angry Birds is a popular video game where the task is to kill pigs protected by a structure composed of different building blocks that observe the laws of physics. The structure can be destroyed by shooting the angry birds at it. The fewer birds we use and the more blocks we destroy, the higher the score. One approach to solve the game is by analysing the structure and identifying its strength and weaknesses. This can then be used to decide where to hit the structure with the birds.

In this paper we use a qualitative spatial reasoning approach for this task. We develop a novel qualitative spatial calculus for representing and analysing the structure. Our calculus allows us to express and evaluate structural properties and rules, and to infer for each building block which of these properties and rules are satisfied. We use this to compute a heuristic value for each block that corresponds to how useful it is to hit that block. We evaluate our approach by comparing the suggested shot with other possible shots.

Introduction

Qualitative spatial representation and reasoning has numerous applications in Artificial Intelligence including robot planning and navigation, interpreting visual inputs and understanding natural language (Cohn and Renz 2008). In recent years, plenty of formalisms for reasoning about space were proposed (Freksa 1992; Frank 1992; Renz and Mitra 2004). An emblematic example is the RCC8 algebra proposed by Randell *et al*. (1992). It represents topological relations between regions such as "x overlaps y", or "x touches y"; however, it is unable to represent direction information such as "x is on the right of y" (Balbiani, Condotta, and del Cerro 1999). The Rectangle Algebra (RA) (Mukerjee and Joe 1990; Balbiani, Condotta, and del Cerro 1999), which is a two-dimensional extension of the Interval Algebra (IA) (Allen 1983), can express orientation relations and at the same time represent topological relations, but only for rectangles. If we want to reason about topology and direction relations of regions with arbitrary shapes, we could for example combine RCC8 and RA, which was analysed by (Liu, Li, and Renz 2009). However, if we only consider the minimum bounding rectangles (MBR) of regions, RA is expressive enough to represent both direction and topology.

What is common in basically all qualitative spatial representations that have been proposed in the literature, and in particular for two-dimensional calculi such as the RA, is that the represented world is assumed to be represented from a birds-eye view, that is from above like a typical map. This is particularly remarkable since qualitative representation and qualitative reasoning are usually motivated as being very close to how humans represent spatial information. But humans typically don't perceive the world from a birds-eye view, the "human-eye view" is to look at the world sideways! When looking at the world sideways, the most important feature of the world is the distinction of "up" and "down" and the concept of gravity. What goes up must come down, everything that is not supported by something and is not stable will invariably fall to the ground. This gives a whole new meaning to the concept of consistency which is typically analysed in the context of qualitative spatial reasoning: a qualitative representation Θ is consistent if there is an actual situation θ that can be accurately represented by Θ. But if any possible θ is not stable and would fall apart under gravity, then Θ cannot be consistent.

In this paper we are concerned with the human-eye view of the world, i.e., we want to be able to have a qualitative representation that takes into account gravity and stability of the represented entities.

Ideally, we want a representation that allows us to infer whether a represented structure will remain stable or whether some parts will move under the influence of gravity or some other forces (e.g. the structure is hit by external objects). Additionally, if the structure is regarded as instable, we want to be able to infer the consequences of the instability, i.e., what is the impact and resulting movement of the instable parts of the structure. Gravity and stability are essential parts of our daily lives and any robot or AI agent that is embedded in the physical world needs to be able to reason about stability. Obvious examples are warehouse robots who need to be able to safely stack items, construction robots whose constructions need to be stable, or general purpose household robots who should be able to wash and safely stack dishes. But in the end, any intelligent agent needs to be able to deal with stability, whenever something is picked up, put down or released. While this problem is clearly of

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

general interest, the motivating example we use in this paper is the popular Angry Birds game. There we have a two-dimensional snapshot of the world, ironically in the human-eye view and not the birds-eye view, and inferring how objects move and fall down under gravity and other forces is of crucial importance when trying to build an AI agent that can successfully play the game.

The Rectangle Algebra is not expressive enough to reason about the stability or consequences of instability of a structure in a human-eye view. While it is possible to some degree, we could simply impose that a structure can only be consistent if each rectangle is supported from below, this is only a necessary condition, it is clearly not sufficient. For example, in Fig. 1(a) and (b), assume the density of the objects is the same and object 2 is on the ground. The RA relation between object 1 and object 2 in these two figures are both <start inverse, meet inverse>, but obviously the structure in Fig. 1(a) is stable whereas object 1 in (b) will fall. In order to make such distinctions, we need to extend the granularity of RA and introduce new relations that enable us to represent these differences. In this paper, we introduce an *Extended Interval Algebra* (EIA) which contains 27 relations instead of the original 13. We use the new algebra as a basis for an *Extended Rectangle Algebra* (ERA), which is obtained in the same way as the original RA. Depending on the needs of an application, we may not need to extend RA to 27 relations in each dimension. Sometimes we only need the extended relations in one axis. Thus, the extended RA may include 13×27, 27×13 or 27×27 relations depending on the requirement of different tasks.

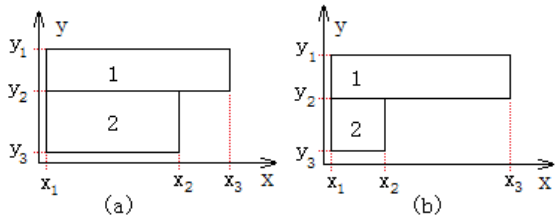


Figure 1: Two structures with the same RA relation (si,mi)

In the following, we will introduce EIA and ERA and then develop a rule-based reasoning method that allows us to infer the stability of a structure and the consequences of the detected instability. Our method is focused towards our target application, namely, to build an agent that can play the Angry Birds game automatically and rationally. This requires us to identify a target that will make a given stable structure maximally instable and will lead to the greatest damage to the structure. In order to do so, we take a stable structure (typically everything we see is by default stable) and for each possible target object we remove the object and analyse the consequences of the structure minus this object. Other applications may require a slightly different method, but the basic stability detection we develop is a general method. The results of our evaluation show that the agent based on this method is able to successfully detect stability and to infer the consequences of instability.

Interval Algebra and Rectangle Algebra

Allen's Interval Algebra defines a set $\mathcal{B}_{int}$ of 13 basic relations between two intervals (see Fig.2). It is an illustrative model for temporal reasoning. The set of all relations of IA is the power set $2^{\mathcal{B}_{int}}$ of $\mathcal{B}_{int}$. The composition ($\circ$) between basic relations in IA is illustrated in the transitivity table in Allen [1983]. The composition between relations in IA is defined as $R \circ S = \cup\{A \circ B : A \in R, B \in S\}$.

Relation	Illustration	Interpretation
X b Y Y a X	X Y	X takes place before Y
X m Y Y mi X	X Y	X meets Y(*i* stands for *inverse*)
X o Y Y oi X	X Y	X overlaps with Y
X s Y Y si X	X Y	X starts Y
X d Y Y di X	X Y	X during Y
X f Y Y fi X	X Y	X finishes Y
X = Y	X Y	X is equal to Y

Figure 2: The 13 basic relations of the Interval Algebra

RA is an extension of IA for reasoning about 2-dimensional space. The basic objects in RA are rectangles whose sides are parallel to the axes of some orthogonal basis in 2-dimensional Euclidean space (see Fig. 1). The basic relations of RA can be denoted as $\mathcal{B}_{rec} = \{(A, B)|A, B \in \mathcal{B}_{int}\}$. The composition between basic RA relations is defined as $(A, B) \circ (C, D) = (A \circ C, B \circ D)$.

The Extended Rectangle Algebra (ERA)

In order to express the stability of a structure and reason about the consequences of the instability in a situation which observes physical rules, we extend the basic relations of IA from 13 relations to 27 relations denoted as $\mathcal{B}_{eint}$ (see Fig.3). The relations consider the importance of the centre of mass of an object for its stability, which corresponds to its centre point for rectangular objects.

Definition 1 (The extended IA relations). *We introduce the centre point of an interval as a new significant point in addition to the the start and end points. For an interval a, denote centre point, start point and end point as c_a, s_a and e_a, respectively.*
1. The 'during' relation has been extended to 'left during', 'centre during' and 'right during' (ld, cd & rd).

- *"x ld y" or "y ldi x" :* $s_x > s_y, e_x \leq c_y$
- *"x cd y" or "y cdi x":* $s_x > s_y, s_x < c_y, e_x > c_y, e_x < e_y$
- *"x rd y" or "y rdi x" :* $s_x \geq c_y, e_x < e_y$

2. The 'overlap' relation has been extended to 'most overlap most', 'most overlap less', 'less overlap most' and 'less overlap less'(mom, mol, lom &lol).

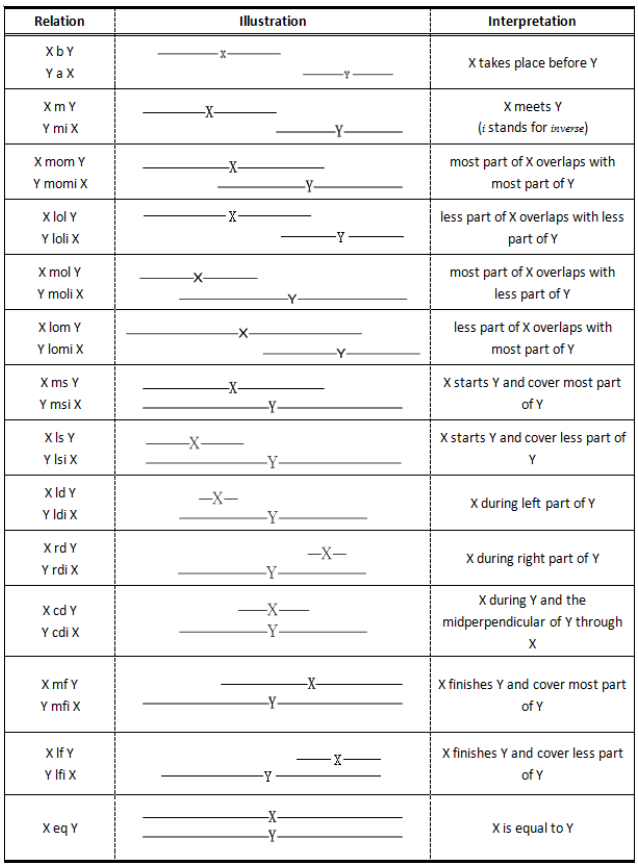


Figure 3: The 27 basic relations $\mathcal{B}_{eint}$ of the extended IA

- *"x mom y" or "y momi x"* : $s_x < s_y, c_x \geq s_y, e_x \geq c_y, e_x < e_y$
- *"x mol y" or "y moli x"* : $s_x < s_y, c_x \geq s_y, e_x < c_y$
- *"x lom y" or "y lomi x"* : $c_x < s_y, e_x \geq c_y, e_x < e_y$
- *"x lol y" or "y loli x"* : $c_x < s_y, e_x > s_y, e_x < c_y$

3. The 'start' relation has been extended to 'most start' and 'less start' (ms & ls).

- *"x ms y" or "y msi x"* : $s_x = s_y, e_x \geq c_y$
- *"x ls y" or "y lsi x"* : $s_x = s_y, e_x > s_y, e_x < c_y$

4. Similarly, the 'finish' relation has been extended to 'most finish' and 'less finish' (mf & lf).

- *"x mf y" or "y mfi x"* : $s_x > s_y, s_x \leq c_y, e_x = e_y$
- *"x lf y" or "y lfi x"* : $s_x > c_y, s_x < e_y, e_x = e_y$

Denote the set of relations of extended IA as the power set $2^{\mathcal{B}_{eint}}$ of the basic relation set$\mathcal{B}_{eint}$. Denote the set of relations of extended RA as the power set $2^{\mathcal{B}_{erec}}$ of the basic relation set$\mathcal{B}_{erec}$. The composition table of EIA and ERA is straightforward to compute.

Note that EIA can be expressed in terms of INDU relations (Pujari, Kumari, and Sattar 2000) if we split each interval x into two intervals x_1 and x_2 that meet and have equal duration. However, this would make representation of spatial information very awkward and unintuitive. There is also some similarity with Ligozat's general intervals (Ligozat 1991) where intervals are divided into zones. However, the zone division does not consider the centre point.

Inferring stability using ERA rules

In the example in Figure 1 we have seen a simple case where a situation is instable. However, since instability is not a concept which is contained in ERA itself, but is rather a consequence of certain ERA cases, compositional reasoning which is typically used in qualitative spatial reasoning is not helpful for determining instability. Instead, we express standard physical stability rules using ERA relations and use rule-based reasoning to infer stability.

For our analysis we assume that the centre of mass of all objects we represent is at the centre of their MBR. This is always the case for circles and rectangles with a uniform density, even if the rectangles are not parallel to the ground. This covers most objects used in Angry Birds, with the exception of triangular shaped objects and non-structurally significant objects. The MBRs of all Angry Birds objects are obtained using the object recognition module of the basic Angry Birds game playing software that is provided by the organisers of the Angry Birds AI competition (AIBIRDS). In order to get the real shapes of objects, we used an improved version of the object recognition module (Ge et al. 2014), which will be added to the basic software package soon. This module also detects the ground, which is important for stability analysis. In accordance with Ge and Renz (2013) we call rectangles that are equivalent to their MBRs *regular* rectangles and otherwise *angular* rectangles. Figure 4 gives an overview of the different kinds of angular rectangles that Ge and Renz distinguish. The distinctions are mainly whether rectangles are fat or slim, or lean to the left or right. Interestingly, these distinctions can be expressed using ERA relations by comparing the projections of each of the four edges of the angular rectangle with the projections of the corresponding MBR.

We first translate all objects we obtain using the object recognition modules into a set of ERA constraints Θ. Since all these objects are from real Angry Birds scenes, all ERA constraints in Θ are basic relations and the identified objects are a consistent instantiation of Θ. This is a further indication that compositional reasoning is not helpful for determining stability. The next step is that we extract all contact points between the objects. This is possible for us to obtain since we have the real shapes of all objects. The contact points allow us to distinguish different types of contact between two objects, which we define as follows:

Definition 2 (Contact Relation). *Two rectangles R_1 and R_2 can contact in three different ways:*

- *If one corner of R_1 contacts with one edge of R_2, then R_1 has a* point to surface contact *with R_2, denoted as $CR(R_1, R_2) = ps$*
- *If one edge of R_1 contacts with one edge of R_2, then R_1 has a* surface to surface contact *with R_2, denoted as $CR(R_1, R_2) = ss$*
- *If one corner of R_1 contacts with one corner of R_2, then R_1 has a* point to point contact *with R_2, denoted as $CR(R_1, R_2) = pp$*

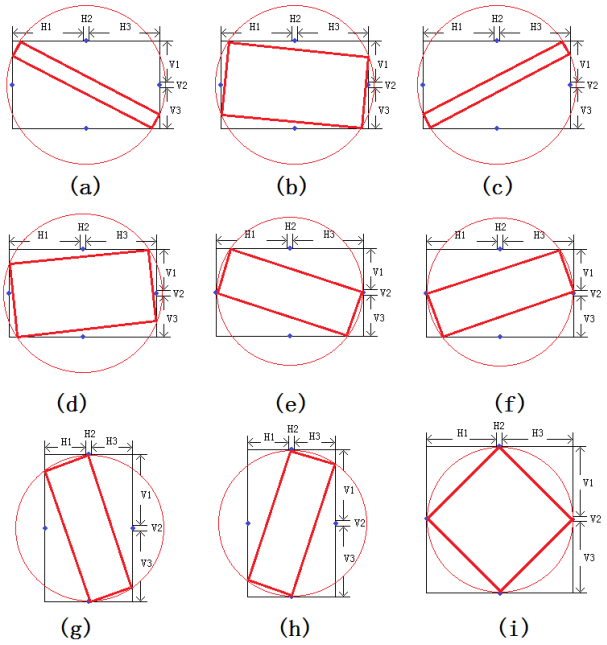


Figure 4: 9 types of angular rectangles (Ge and Renz 2013)

- *If R_1 does not touch R_2, then R_1 has* no contact *with R_2, denoted as $CR_{R_1,R_2} = n$*

Definition 3 (Contact Dimension). *The* contact dimension *expresses in which dimension (horizontal or vertical) two rectangles R_1 and R_2 contact each other. Specifically, if $CR(R_1, R_2) \in \{ps, ss, pp\}$, then:*

- *R_1 and $_2$* contact horizontally *(denoted as $CD(R_1, R_2) = hc$), if $ERA(R_1, R_2) \in (\{m, mi\}, U_{EIA})$, where $\mathcal{U}_{EIA}$ is the set of all EIA relations.*
- *In all other cases R_1 and R_2* contact vertically*, denoted as $CD(R_1, R_2) = vc$.*

If $CR(R_1, R_2) = n$, then $CD(R_1, R_2) = n$, too.

Stability of Regular Rectangles

We start with stability for regular rectangles as the rules are obvious consequences of Newtonian physics. Throughout this paper, we will never consider what is happening *above* an object when determining its stability. As an example how this can influence stability, refer again to Figure 1(b). Suppose we put a very heavy object 3 on object 1, then object 1 might be stable if the combined centre of mass of objects 1 and 3 is above object 2. Since this requires numerical calculations and cannot be obtained qualitatively, we will always ignore these cases, and consequently will only be able to obtain an approximation of stability. We therefore call our stability concept *ERA-stability*. ERA-stability is based on the simple physical rule that an object is stable and will not topple if the vertical projection of the centre of mass of an object falls into the area of support base.

Given a set of ERA-constraints Θ and a consistent instantiation θ, where the centre of mass of all elements $x \in \theta$ coincides with the centre of the MBR of x. ERA-stability for regular rectangles $x \in \theta$ is determined recursively using the following rules, until no more objects are determined ERA-stable:

A regular rectangle $x \in \theta$ is ERA-stable, if one of the following rule applies:

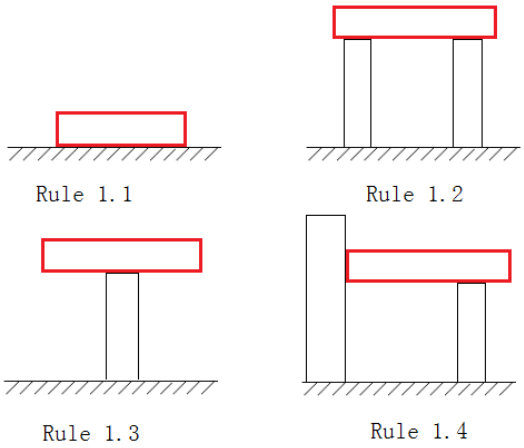


Figure 5: Examples for ERA-stable cases of regular rectangles

Rule 1.1: x lies on the ground.

Rule 1.2: $\exists y, z \in \theta$ such that y and z are both ERA-stable and $CD(y, x) = CD(z, x) = vc$: $ERA_x(x, y) \in \{momi, moli, lomi, loli, msi, lsi, ldi\} \wedge ERA_x(x, z) \in \{mom, mol, lom, lol, mfi, lfi, rdi\}$.

Rule 1.3: $\exists y \in \theta$ such that y is ERA-stable and $CD(y, z) = vc$: $ERA_x(x, y) \in \{ms, mf, msi, ls, mfi, lf, cd, cdi, ld, rd, mom, momi, lomi, mol\}$.

Rule 1.4: $\exists y, z \in \theta$ such that y and z are both ERA-stable, $CD(x, y) = hc$ and $CD(x, z) = hc$: $((ERA_x(x, y) \in \{mi\} \wedge ERA_x(x, z) \in \{mom, mol, lom, lol, mfi, lfi, rdi\}) \vee (ERA_x(x, y) \in \{m\} \wedge ERA_x(x, z) \in \{momi, moli, lomi, loli, msi, lsi, ldi\}))$

Rule 1.2 covers the cases where the vertical projection of the mass centre of an object is located between two ERA-stable supporters y and z. Rule 1.3 covers the cases where it is located above the contact area of an ERA-stable supporter y. Rule 1.4 covers cases where x is horizontally contacting with an ERA-stable object on one side and has an ERA-stable supporter on the other side. Fig. 5 shows examples of the four ERA-stable cases for a regular rectangle. In all four rules x is always ERA-stable.

Stability of Angular Rectangles

The case of angular rectangles is more complicated than regular rectangles, but it follows the same physical rule that if an object is stable, the vertical projection of the centre of mass must falls into a support area. While the support area

for regular rectangles is defined by two supporting objects, for angular rectangles it can be defined by the ground, acting as a supporting object, and one supporting object. Angular rectangles always have a momentum to become regular rectangles. The rectangles in Figure 4(a,b,e) have a momentum in counter-clockwise direction, the rectangles in Figure 4(c,d,f) have a momentum in clockwise direction. The three rectangles in the bottom row probably never occur in reality as the top point has to be exactly above the bottom point and we ignore these cases. The momentum means that an angular rectangle always needs support to prevent it from becoming a regular rectangle. In cases where the bottom point of an angular rectangle is supported, for example by the ground, the second support must always be on the opposite side of the centre of mass. This leads to the following rules, which again are applied recursively, until no more objects are determined ERA-stable:

An angular rectangle $x \in \theta$ is ERA-stable, if one of the following rule applies:

Rule 1.5: $\exists y, z \in \theta$ such that y and z are both ERA-stable and $CD(y,x) = CD(z,x) = vc$: $ERA_x(x,y) \in \{momi, moli, lomi, loli, msi, lsi, ldi\} \wedge ERA_x(x,z) \in \{mom, mol, lom, lol, mfi, lfi, rdi\}$.

Rule 1.6: $\exists y \in \theta$ such that y is ERA-stable and $CD(y,x) = vc$ and $CR(x,y) = ss$: $ERA_x(x,y) \in \{ms, mf, msi, ls, mfi, lf, cd, cdi, ld, rd, mom, momi, lomi, mol\}$.

Rule 1.7: $\exists y, z \in \theta$ such that y and z are both ERA-stable and $CD(y,x) = hc$ and $CD(z,x) = vc$: $((ERA_x(x,y) \in \{mi\} \wedge ERA_x(x,z) \in \{mom, mol, lom, lol, mfi, lfi, rdi\}) \vee (ERA_x(x,y) \in \{m\} \wedge ERA_x(x,z) \in \{momi, moli, lomi, loli, msi, lsi, ldi\}))$.

Rule 1.8: $\exists y, z \in \theta$ such that y and z are both ERA-stable and $CD(y,x) = CD(z,x) = hc : ERA_x(x,y) \in \{m\} \wedge ERA_x(x,z) \in \{mi\}$

The principle of Rule 1.5 is similar to Rule 1.2 which tests if the centre of mass falls between two vertical supporters. Rule 1.6 is similar to Rule 1.3, the difference is that it requires the two objects to be surface-to-surface contact, otherwise the angular rectangle will topple. The supporter in this case also needs sufficient support to remain stationary. Rule 1.7 is a similar rule as Rule 1.4 which describes horizontal support for angular rectangles. Rule 1.8 describes a different property of angular rectangles, which is an angular rectangle can remain stable with the support of two horizontally contacting objects on both left and right sides. This is because the two objects on the sides can provide upward frictions to support the angular rectangle. Fig. 6 shows example cases of the above four rules where angular rectangles are ERA-stable.

All these rules only consider objects as support that are already stable. Cases where two rectangles are stable because they support each other are not considered. Applying rules 1.1-1.8 recursively to all objects in θ until no more objects are detected as ERA-stable identifies all ERA-stable objects. If all objects in θ are ERA-stable, then θ is ERA-stable. The recursive assignment of ERA-stability to objects

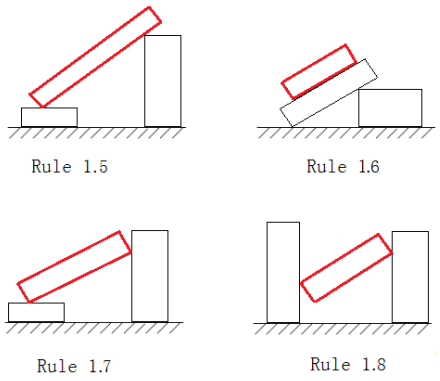


Figure 6: ERA-stable cases of angular rectangles

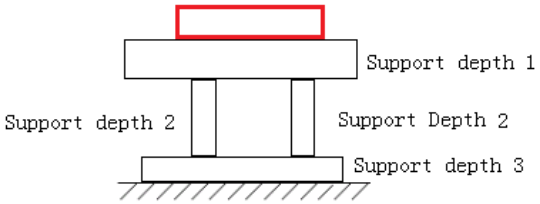


Figure 7: Illustration of support structure

allows us to build a *stability hierarchy*. Regular objects that lie on the ground are objects of *stability level* 1. Then, objects that are stable because they are supported by an object of stability level ℓ and possible objects objects of stability level smaller than ℓ are considered to be of level $\ell + 1$. Conversely, we could define the *support depth* of a target object (see Figure 7). Those object that are directly responsible for the ERA-stability of a target object x as they support it have support depth 1. Objects that are directly responsible for the ERA-stability of an object y of support depth ℓ as they support it have support depth $\ell + 1$. This can be calculated while recursively determining stability of objects. The support level can be helpful in order to separate the support structure of an object from the larger structure, or when determining which object could cause the most damage to a structure when destroyed.

Applying ERA-stability to identify good shots in Angry Birds

The goal of the Angry Birds game is to shoot birds using a slingshot in a way that all pigs are killed with the fewest number of used birds. The pigs are typically supported by complicated structures that have to be destroyed in order to kill the pigs. Instead of shooting at a pig directly, it might be better to destroy the whole structure that supports the pig. This could be optimised by using the support structure we defined in the previous section.

Another useful substructure is the shelter of the pigs. The

reason is straightforward, if a pig is not reachable, there must be some objects that protect it; these objects form the sheltering structure of the pigs. Destroying the sheltering structure can either kill the pig or make the pig directly reachable to the bird. We define the sheltering structure of a pig as the closest objects that protect the pig from a direct hit from each direction. Specifically, a sheltering structure of an object (usually a pig) consist of left, right and top sheltering objects. In order to obtain the sheltering structure of a certain object, the first step is to get the closest object from the left side of the queried object; then, get the supportee list of the object (similar process as getting the supporter list); after that, get the right closest object with its supportee list. The next step is to check if the two supportee lists have objects in common. If so, pick the one with smallest depth as the roof object of the sheltering structure; if not, there is no sheltering structure for the queried object. If a roof object is found, also put the supportees of both the left and right closest objects with smaller depth than the roof object into the sheltering structure. Finally, put the supporters of both left and right closest objects which are not below the queried object into the sheltering structure. This can also be described using rules expressed in ERA (actually RA is sufficient).

The integration of the rules to evaluate a shot

With the rules described above, we are able to dynamically analyse the possible consequences after a shot has been made. In order to predict the final consequence of an external influence on the structure (usually the impact from a bird), the direct consequence and its following subsequences should be analysed in detail. Funt (1987) suggested a similar method to simulate the consequence of a structure with a changed object which assumes that the changed object disappears and chooses the most significant instable object to simulate the consequence.

In the following, we analyse different cases of how a structure can be impacted from an external force and, in particular, what other objects will be affected by an impact to the target object. We separately analyse *regular rectangles* and *angular rectangles*, each of which can be affected in four different ways. First we consider the four cases where the affected objects are regular rectangles:

Case 1: The target object is directly hit by another object. The direct consequence to the hit object will be one of three types: **destroyed, toppling, remaining stationary**. Empirically, the way to determine the consequence of the hit depends on the height and width ratio of the target. For example, if an object hits a target with the height and width ratio larger than a certain number (such as 2), the target will fall down. And this ratio can be changed to determine the conservative degree of the system. In other words, if the ratio is high, the system tends to be conservative because many hits will have no influence on the target. Moreover, if the external object affects a target with the height and width ratio less than one, the target itself will remain stable temporarily because the system will also evaluate its supporter to determine the final status of the target. After deciding the direct consequence of the hit, the system should be able to suggest further consequences of the status change of the direct target. Specifically, if the target is destroyed, only its supportees will be affected. If the target falls down, the case will be more complex because it may influence its supporters due to the friction, as well as its supportees and neighbours. If the target remains stable temporarily, it will also influence its supporters, and its supporters may again affect its behaviour. Algorithm 1 demonstrates the evaluation process of an object that has been hit by another object.

Algorithm 1 Direct hit estimation

```
1:  PROCEDURE Hit(Object o)
2:  o ∈ pendingList
3:  if o ∈ affectedList then
4:     pendingList.remove(o)
5:     Exit()
6:  end if
7:  if o.isRegular then
8:     if o.affectedMethod = "Hit" then
9:        if o.isDestroyed then
10:          affectedList.add(o)
11:          pendingList.add(supporteesOf(o))
12:       else if !o.isDestroyed then
13:          if o.height/o.width > threshold then
14:             affectedList.add(o)
15:             pendingList.add(supporteesOf(o))
16:             pendingList.add(supportersOf(o))
17:          else
18:             pendingList.add(supportersOf(o))
19:          end if
20:       end if
21:    end if
22: end if
23: pendingList.remove(o)
```

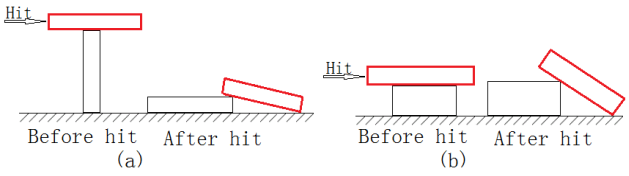


Figure 8: Two examples for case 2.

Case 2: The supportee of the target object topples down. The target object's stability is also determined by its height to width ratio, but the number should be larger (about 5) as the influence from a supportee is much weaker than from a direct hit. If the target is considered as instable, it will fall down and affect is neighbours and supporters; otherwise, it will only influence its supporters (see Fig. 8).

Case 3: The supporter of the target object topples down. Here a structure stability check process (applying Rules 1.1 - 1.4) is necessary because after a supporter falls,

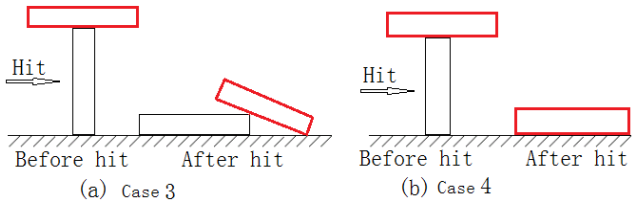


Figure 9: Examples for Cases 3&4

the target may have some other supporters and if the projection of its mass centre falls into the areas of its other supporters, it can stay stable. Then, if the target remains stable, it will again only affect its supporters due to the friction; otherwise, it may fall and affect its supporters, supportees and neighbours (see Fig. 9(a)).

Case 4: The supporter of the target is destroyed. This is more like a sub case of the previous one. If the target cannot remain stable after its supporter is destroyed, it may fall and affect its supporters, supportees and neighbours (see Fig. 9(b)).

Next we consider the four cases where the target objects are angular rectangles:

Case 5: The target object is directly hit. As analysed in Section *Inferring stability using ERA rules*, angular rectangles can be classified into 9 classes. If we only consider the direction of the objects, there only exists 3 classes, which are *"lean to left"*,*"lean to right"* and *"stand neutrally"*. Assume the hit is from left to right. Then if a *"lean-to-left"* object is hit and the force is strong enough, the object will rotate around the lowest pivot, i.e. the lowest corner of the object. However, in the Angry Birds game, before the force becomes sufficient to make the object rotate, the object will be destroyed first. Thus, the two possibilities here are either the target object affects its supporters or it is destroyed and affects its supportees. The case of *"lean-to-right"* object is the same as above. For a *"stand-neutrally"* object, it cannot stand by itself and at least one *point-to-surface* touched supporter on each side is necessary. If it is hit from left, there will be no friction applied to its left supporter, thus it will either be destroyed or affect its right supporter.

Case 6: The supportee of the target object topples down. Angular objects will not topple due to the fall of their supportees. Because their supporters restrict their spatial location. And therefore only the supporters will be affected. For *"stand-neutrally"* objects, if the friction given by their supportees is from left to right, the left supporters will not be affected, and vice-versa for the right supporter.

Case 7: The supporter of the target object topples down. First check the stability using Rules 1.5 - 1.8. If the object still has sufficient support, it only affects its supporters, otherwise it will rotate and affect its supportees, other supporters and neighbours.

Case 8: The supporter of the target object is destroyed. Again, a stability check is applied first. However if it remains stable, it will not affect its supporters because no friction applies to the target from the destroyed supporter. If it is not stable, than it will affect its supportees, other supporters and neighbours.

By applying these cases recursively for each potential target object, we obtain a list of objects that are affected when hitting a particular target object. Then, with all the affected objects in a list, the quality of the shot can be evaluated by calculating a total heuristic value depending on the affected objects. The scoring method is defined as follows and is based on experimenting with different heuristic values: if an affected object belongs to the support structure or the sheltering structure of a pig, 1 point will be added to this shot; and if the affected object is itself a pig, 10 points will be added to the shot. After assigning scores to shots at different possible target objects, the target with the highest score is expected to have the largest influence on the structures containing pigs when it is hit. The heuristic value roughly corresponds to the number of affected objects plus a higher value for destroyed pigs. Then, based on different strategies, the agent can choose either to hit the reachable object with highest heuristic value or generate a sequence of shots in order to hit the essential support object of the structure.

We first extract the ERA relations between all objects and then match the rules for all relevant combinations of objects. Thus, the process of evaluating the significance of the targets is straightforward and fast.

Planning in Angry Birds

The previous analysis only looks at the effects of one shot, while taking into account reachability of target objects. Reachability is calculated using the impact different bird types have on different block types. For example, a yellow bird can hit through 2 layers of wood, a blue bird can go through 2 layers of ice, etc. We can use this knowledge to reason about the reachability of a bird and roughly estimate the damage of the direct consequence of a shot. Ideally, we would like to plan a sequence of shots. Due to the inability to accurately predict the state after a shot (we can currently only predict affected objects), successful planning of shot sequences seems very hard.

However, in order to demonstrate how planning in Angry Birds could work, we have implemented one special case of planning, which can be used when the shot with highest score is not reachable due to some blocks in the path. We can identify which blocks are on the trajectory and block the target. Then a planner could be used to generate a shot to first clear the blocks in the path and then hit the primary target with the second shot. Specifically, the planner will first check if the highest-ranked shot is reachable. If not, it will retrieve the block objects in the calculated trajectory. Then it will search for a shot that can affect most of the objects in the block list. Finally, the planner will suggest using the first shot to clear the blocks if applicable. However, as we cannot exactly simulate the consequence of a shot, sometimes the first shot may lead to an even worse situation. Initial experi-

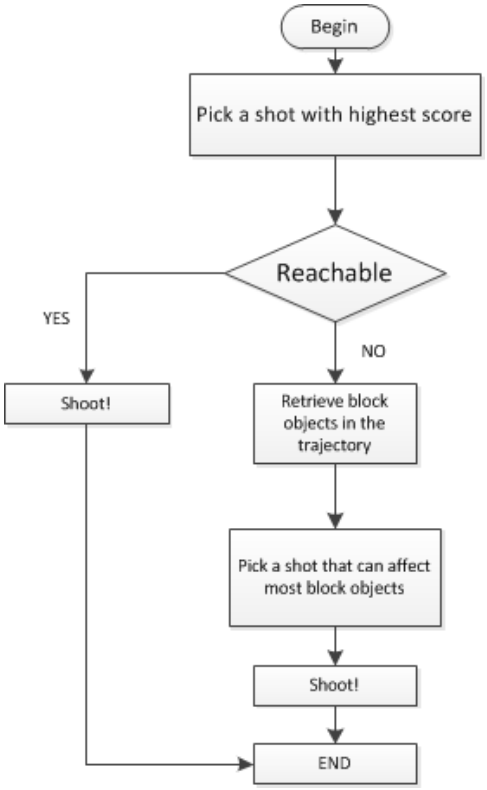


Figure 10: Process of the proposed two step planner

ments we did with this method showed good results though. Fig.10 shows the process of the planner.

Evaluation

We evaluate our approach by applying it to the first 21 Poached Eggs levels, which are freely available at chrome.angrybirds.com. These levels are also used to benchmark the participants of the annual Angry Birds AI competition. We take each level and calculate the heuristic value of all reachable objects and rank them according to the heuristic value. We then fire the bird at each of the reachable objects and record the actual game score resulting from each shot. We reload the level before each shot, so each bird always fires at the initial structure. We restricted our analysis to initial shots only, as we found that it is not possible to consistently obtain the same intermediate game state simply by repeating the same shot. However, our method works equally well for intermediate game states.

The results of our evaluation are shown in Table 1 where we listed for each of the 21 levels how many reachable objects there are, the resulting game scores of shooting at the four highest ranked reachable objects, as well as the rank of the object that received the highest game score. It turns out that in about half of the levels (10 out of 21), our method correctly identified the best object to hit. In a further 5 cases, the second highest ranked object received the highest score, in 2 cases the 3rd highest ranked object. On average there are about 7.8 reachable objects. The average score of the shooting at the highest ranked object over all 21 levels is about 21,700, shooting at the second ranked object gives an average of about 16,700 and the third ranked object an average of about 13,100. This clearly demonstrates the usefulness of our method in identifying target objects.

Table 1: Evaluation on first 21 levels. #RO is the number of reachable objects, S1, S2, and S3 are the scores obtained when shooting at the highest, second-highest and third highest ranked object. HS is the rank of the reachable object with the highest score.

Level	#RO	S1	S2	S3	Time(s)	HS
1	6	29030	30370	27330	1.34	2
2	7	16240	6040	11250	1.06	1
3	5	5410	41850	6790	0.77	4
4	2	36890	1850		1.45	1
5	6	22380	66240	16800	5.11	2
6	10	1610	6770	7710	1.83	4
7	10	46200	6230	6210	3.81	1
8	7	17310	17310	6010	1.08	4
9	9	14030	10140	13210	5.37	1
10	9	22920	3860	5920	3.88	1
11	10	57110	22640	9680	6.60	1
12	6	26570	15010	21800	3.47	1
13	8	12200	12650	14460	4.02	3
14	10	30330	28590	37670	2.43	7
15	9	16880	17550	4580	3.97	2
16	11	15090	22430	23860	4.24	3
17	12	48850	12240	19850	7.18	1
18	10	9460	14590	3240	3.02	2
19	2	5870	3290		5.83	1
20	8	7420	8170	6820	4.46	2
21	8	14270	4650	7170	12.70	1

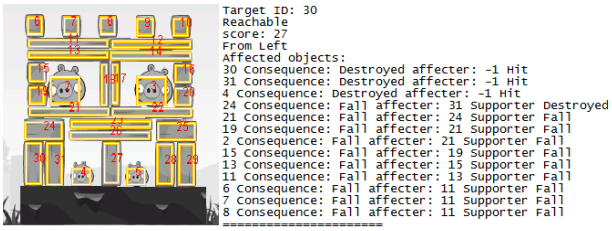


Figure 11: Level 1-16 in Chrome Angry Birds and agent output for this level

However, the resulting game score is not always a good measure, as shots can have unexpected side effects that lead to higher scores than expected. These side effects are typically unrelated to the stability of a structure and, hence, are not detected by our method. As an example, consider level 16 where the second and third ranked objects resulted in a higher score than the first ranked object. Figure 11 shows the level and output of our stability analysis. Our method predicts that shooting at object 30 will kill the two pigs on the left and result in the toppling of the left half structure.

Figure 12 shows the actual result after the recommended shot has been made. We can see that the prediction of the system is very accurate in this case. The left two pigs are

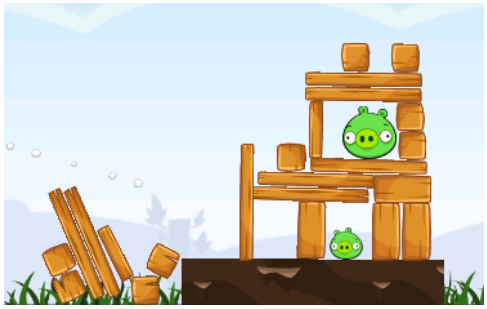


Figure 12: Actual result for the recommended shot

killed and the left part of the structure toppled. However, the shots at the second and third ranked objects each killed three pigs and therefore obtained higher scores. It turns out that two of the three pigs were killed when they bounced to the wall without actually destroying much of the structure (see Figure 13). Therefore, the next shot is much less likely to solve the level as more of the structure remains. So the shot at the highest ranked object first is likely still the best shot overall as the second shot can easily solve the level. This example nicely shows that Angry Birds requires even more sophisticated methods to correctly predict the outcome of shots.



Figure 13: Actual result for the 3rd ranked shots

Related Work

Qualitative spatial representation and reasoning has been applied to some physical systems to do common sense reasoning (Klenk et al. 2005). Physical reasoning has two important theories, namely kinematics and dynamics(Forbus 1988). Kinematics mainly concerns about the position of objects which may change continuously over time and the shape which is strictly rigid and static over time (Davis 2008). However it considers less about the forces between objects and the type of motions. These features make kinematics easy to be formulated, on the other hand models using kinematics are usually limited by the context and appear to be less expressive. The CLOCK Project (Forbus, Nielsen, and Faltings 1991) is a typical success which uses a variety of techniques but mainly kinematics to represent and reason about the inner motions of a mechanical clock. Under some given environments and constraints, this approach can successfully analyse the mechanisms; however, as this system requires restricted constraints such as the exact shape of the parts and the range of the motion, it may not be applicable in the cases with high degrees of freedom such as the world of Angry Birds.

In contrast, dynamics takes force analysis into account which allows reasoning about more complex motions and the transfer of motion. However, the limitation is that precisely predicting a consequence of a motion in an arbitrary environment is almost impossible due to uncertainty. However, it is possible to reason about some simpler physical features of a structure, such as stability. Fahlman (1973) implemented a stability verification system dealing with the stability of blocks with simple shapes (e.g. cuboid). This system can produce qualitative output to determine whether a structure is stable without external forces, but it requires some quantitative inputs (e.g. magnitude of forces and moments). The stability check in our work is partially inspired by this system, however without quantitative inputs about forces, we use a purely qualitative way to analyse the probability of the balance of forces and moments in a structure. Although our approach may produce a few false predictions in very complex cases, it is a good estimation of humans' reasoning approach. Our work also focuses on reasoning about the impacts of external forces (a hit from an external object) on a structure, which has not been discussed much in previous work.

Additionally, a method for mapping quantitative physical parameters (e.g. mass, velocity and force) to different qualitative magnitudes will be very useful for the situation where the input information is often incomplete. (Raiman 1991) developed a formalization to define and solve *order of magnitude equations* by introducing comparison operators such as $\approx$ (in the same magnitude) and $\ll$ (far less than). Raiman's method is of particular significance for our approach, for example, we will be able to infer that a hit from 1×1 block will not change the motion state of a 100×100 block with the same density.

Discussion and Future Work

In this paper we have introduced an extended rectangle algebra useful for representing and reasoning about stability under gravity and other properties of 2-dimensional structures. By splitting some basic interval relations into more detailed ones, we obtained 27 interval relations in each dimension that can express the physical relations between rectangular objects more precisely. Our algebra is very useful when representing a human-eye view of the physical world, rather than the birds-eye view which is surprisingly common in qualitative spatial representation and reasoning. A nice example of a two-dimensional human-eye view of the world is Angry Birds, where we can represent the world using the new algebra.

In addition to representing what we see using the extended interval algebra, we also used it for defining some useful structural rules regarding properties such as stability, reachability, support, and shelter that allow us to perform rule-based reasoning about these properties. We tested the usefulness of our rules by designing an agent that performs a struc-

tural analysis of Angry Birds levels. Based on these rules, we predict for each block the consequences if it gets hit and calculate a heuristic value that determines the usefulness to hit the block. We then shoot at the block with the highest value that is reachable with the current bird. Our evaluation shows that our method is very successful in determining good target objects and good shots.

Despite its success, some parts of our method offer significant room for improvement. For example, our current motion analysis can only vaguely predict which objects may be affected by a shot, i.e., we do not have accurate predictions about a possible successor state at this stage. Because of this, we currently cannot plan ahead and evaluate sequences of shots in advance. Also we only consider stability of an object with respect to its supporters, but not with respect to objects that are on top of it, which means we do not analyse the interactions between substructures. In the future, we will try to design a feedback loop between the dynamic analysis and ERA, specifically, so we can use the results of the dynamic analysis to restrict a range of possible subsequent positions of an object and use ERA rules to suggest several possible successor cases. Then we can use dynamic analysis again to check the stability of these cases and finally decide on desirable successor states. Moreover, we will find a method to determine useful substructures in order to increase the reliability of the dynamic analysis.

Acknowledgements

This research was supported under Australian Research Council's Future Fellowship and Discovery Projects funding schemes (project numbers FT0991917 and DP120104159).

References

AIBIRDS. Angry Birds AI competition. http://aibirds.org.

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26:832–843.

Balbiani, P.; Condotta, J.; and del Cerro, L. 1999. A new tractable subclass of the rectangle algebra. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 442–447.

Cohn, A. G., and Renz, J. 2008. Qualitative spatial representation and reasoning. In *Handbook of Knowledge Representation*. Oxford, England: Elsevier. 551–584.

Davis, E. 2008. Physical reasoning. In *Handbook of Knowledge Representation*. Oxford, England: Elsevier. 597–620.

Fahlman, S. E. 1973. A planning system for robot construction tasks. Technical report, Cambridge, MA, USA.

Forbus, K. D.; Nielsen, P.; and Faltings, B. 1991. Qualitative spatial reasoning: the clock project. *Artificial Intelligence* 51(1-3):417–471.

Forbus, K. D. 1988. Commonsense physics: a review. *Annual Review of Computer Science* 3(1):197–232.

Frank, A. U. 1992. Qualitative spatial reasoning about distances and directions in geographic space. *Journal of Visual Languages & Computing* 3(4):343–371.

Freksa, C. 1992. Using orientation information for qualitative spatial reasoning. In *Theories and Methods of SpatioTemporal Reasoning in Geographic Space*.

Funt, B. V. 1987. Problem-solving with diagrammatic representations. In *Readings in computer vision: issues, problems, principles, and paradigms*, 456–470. Morgan Kaufmann.

Ge, X., and Renz, J. 2013. Representation and reasoning about general solid rectangles. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 905–911.

Ge, X.; Gould, S.; Renz, J.; Abeyasinghe, S.; Keys, J.; Wang, A.; and Zhang, P. 2014. Angry Birds game playing software version 1.3: Basic game playing software. http://www.aibirds.org/basic-game-playing-software.html.

Klenk, M.; Forbus, K.; Tomai, E.; Kim, H.; and Kyckelhahn, B. 2005. Solving everyday physical reasoning problems by analogy using sketches. In *Proceedings of the 20th National Conference on Artificial Intelligence*, 209–215.

Ligozat, G. 1991. On generalized interval calculi. In *Proceedings of the 9th National Conference on Artificial Intelligence*, 234–240.

Liu, W.; Li, S.; and Renz, J. 2009. Combining RCC-8 with qualitative direction calculi: Algorithms and complexity. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 854–859.

Mukerjee, A., and Joe, G. 1990. A qualitative model for space. In *Proceedings of the 8th National Conference on Artificial Intelligence*, 721–727.

Pujari, A. K.; Kumari, G. V.; and Sattar, A. 2000. Indu: An interval duration network. In *Proceedings of the 16th Australian joint conference on AI*, 291–303.

Raiman, O. 1991. Order of magnitude reasoning. *Artificial Intelligence* 51(1):11–38.

Randell; Cui, Z.; and Cohn, A. G. 1992. An interval logic for space based on connection. In *Proceedings of the 10th European Conference on Artificial Intelligence*, 394–398.

Renz, J., and Mitra, D. 2004. Qualitative direction calculi with arbitrary granularity. In *Proceedings of the 8th Pacific Rim International Conference on Artificial Intelligence*, 65–74.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Diagnostic Problem Solving via Planning with Ontic and Epistemic Goals

Jorge A. Baier
Depto. de Ciencia de la Computación
Pontificia Universidad Católica de Chile
Santiago, Chile

Brent Mombourquette
Department of Computer Science
University of Toronto
Toronto, Canada

Sheila A. McIlraith
Department of Computer Science
University of Toronto
Toronto, Canada

Abstract

Diagnostic problem solving involves a myriad of reasoning tasks associated with the determination of diagnoses, the generation and execution of tests to discriminate diagnoses, and the determination and execution of actions to alleviate symptoms and/or their root causes. Fundamental to diagnostic problem solving is the need to reason about action and change. In this work we explore these myriad of reasoning tasks through the lens of artificial intelligence (AI) automated planning. We characterize a diversity of reasoning tasks associated with diagnostic problem solving, prove properties of these characterizations, and define correspondences with established automated planning tasks and existing state-of-the-art planning systems. In doing so, we characterize a class of planning tasks with epistemic and ontic goals which we show can be compiled into non-epistemic planning, allowing state-of-the-art planners to compute plans for such tasks. Furthermore, we explore the effectiveness of using the conditional planner Contingent-FF with a number of diagnostic planning tasks.

1 Introduction

Automated diagnosis seeks to determine what is wrong with a system, prompted by some observations of egregious behaviour. As our world becomes increasingly instrumented with sensors – our street corners, our homes, our cars, and even our bodies – and as the infrastructure that controls our power, communication, and transportation systems grows in complexity, we must rely on computers to monitor the state of these systems and to oversee their operation. Unfortunately, these systems can and do malfunction, resulting in diagnostic problems of such enormous complexity that they confound human reasoning.

Diagnostic Problem Solving (DPS) refers to the myriad of reasoning tasks associated with the diagnosis, testing, and repair of a system. In this work we advocate for a *purposeful view of diagnostic problem solving*. While a naïve approach to DPS suggests that we generate candidate diagnoses, identify a unique diagnosis through testing, and then treat or repair, we instead observe that identifying candidate diagnoses may be unnecessary or perhaps only necessary to the extent that it informs an appropriate course of action to be selected – a course of action that may result in the realization of further tests to discriminate diagnoses, or to alleviate the symptoms or potential root causes, potentially without actually identifying a unique root cause.

Example 1 *Consider a run-of-the-mill flashlight that is not emitting light. A common response is to turn the flashlight on and off a few times. If it's still malfunctioning, the most likely hypothesis is that the batteries are dead, but it could also be the case that the bulb is burned out, or that there is a loose connection somewhere. That's three candidate diagnoses, and there could be more. A typical course of action would be to open up the flashlight, take out the batteries, put in new ones, re-assemble the flashlight and turn it on again. If the flashlight emits light, you'll likely be happy, recycle the batteries and consider yourself "done." Your purpose was* not *to diagnose the flashlight, but rather to get it working again. A more careful examination of what went on shows that the course of action you took, together with the happy outcome that the flashlight is now "working," served to eliminate the hypothesis that the bulb was burned out, and it tightened the connection, effectively* repairing *the connection regardless of whether it was faulty or not. This changed the space of hypotheses under consideration. What is equally interesting is that this sequence of actions neither confirmed nor refuted the hypothesis that the batteries were dead. The cause of the faulty behaviour could have been the result of a loose connection which got fixed in the process of changing the batteries. Those batteries you recycled could still be OK! What's also noteworthy is that if after executing the procedure the flashlight had not emitted light, you would still be left with the hypotheses that the original batteries were dead or that the bulb was broken, but you would also have the further (granted, unlikely) hypothesis that the new batteries were also dead.*

The above seemingly simple DPS scenario illustrates the need for reasoning about action and change as well as reasoning about knowledge, and in particular what an agent comes to know about aspects of the world (symptoms and diagnoses in this case) based upon the execution of both world-altering and sensing actions. It is also suggestive of the somewhat subordinate role the actual candidate diagnoses may play in the resolution of a system failure scenario.

In this paper, we explore this purposeful view of diag-

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

nostic problem solving through the lens of AI automated planning. Our motivation for doing so is pragmatic. We are interested in characterizing the fundamental knowledge representation and reasoning tasks that underlie DPS in its many guises, but we wish to do so in a manner that supports establishing its correspondence with the state of the art in AI automated planning theory and practice. There have been tremendous advances in AI automated planning systems over the last decade and a number of highly optimized planning systems exists. Further there have been significant recent advances in non-classical planning such as conformant or contingent planning that support planning with incomplete information and/or with sensing. A major contribution of this paper is to show that many purposeful DPS tasks that have been heretofore unsolvable can be realized in varying measure through recent advances in state-of-the-art AI planning systems. Our characterization not only allows us to solve certain problems now, but it provides the insight and understanding that will support the realization of these and isomorphic tasks as computational machinery in planning improves in the coming years.

Characterizing diagnostic problem solving tasks in terms of planning builds upon a large body of research that includes research on topics as varied as reasoning about knowledge and action (e.g., Scherl and Levesque 2003; Petrick and Bacchus 2002), planning with epistemic goals (e.g., Herzig, Lang, and Marquis 2003; Aucher and Bolander 2013), and conformant (e.g., Palacios and Geffner 2009), contingent (e.g., To, Pontelli, and Son 2011), and classical (e.g., Helmert 2006) planning. We discuss this related research later in the paper.

In Section 2 we introduce the mathematical formalisms upon which our work rests. In Section 3 we introduce the notion of a diagnosis that we use here, and contrast it to other forms of diagnosis discussed in the literature. With a definition of diagnosis in hand, in Section 4 we present the notion of a diagnostic plan, establish its correspondence to known planning paradigms, and establish properties of various forms of diagnostic plans. In Section 5 we introduce a notion of epistemic diagnostic planning – planning to determine a particular diagnosis, or to discriminate diagnoses. We define compelling classes of epistemic goals and then show that it is possible to use state-of-the-art non-epistemic planners to plan for such epistemic goals, by providing a sound and complete translation of such epistemic tasks to conditional planning. In Section 6 we turn our attention to the realization of our newly established diagnostic planning tasks via existing planning systems. We conclude with some reflections on our work, its relationship to other work and prospects for future work.

2 Preliminaries

In this section we introduce a planning language that will allow us to define various kinds of planning tasks which we then will show can model interesting DPS tasks. In particular, we define a common language for deterministic (classical), conformant, and conditional planning.[1] The planning language we present below builds on the ADL planning language (Pednault 1989), and considers extensions for uncertainty about the initial state and conditional plans that have been presented in a similar way by other researchers (e.g., Palacios and Geffner 2006; To, Pontelli, and Son 2011).

2.1 Dynamical Systems

Dynamical systems can be formally described in many ways. In this paper we model them as transition systems, which we represent with a standard planning language. As such, transitions occur as the result of actions described in terms of preconditions and effects, and the domain is finite (i.e., there is a finite number of system configurations). Formally, a dynamical system is a tuple $\Sigma = (F, A, \Omega, I)$, where F is a finite set of fluent symbols, A is a set of deterministic actions, Ω is a set of sensing actions, and I is a set of clauses over F that defines a set of possible initial states. If $p \in F$, then p and $\neg p$ are *fluent literals*. If ℓ is a literal, we denote its complement by $\overline{\ell}$; thus, $\overline{p} = \neg p$ and $\overline{\neg p} = p$. Every action $a \in A$ is defined by a precondition $prec(a)$, which is a conjunction of fluent literals, and $\mathit{eff}(a)$, a set of conditional effects of the form $C \rightarrow L$, where C is a conjunction of fluent literals and L is a fluent literal. We sometimes write the unconditional effect $\rightarrow L$ as simply L, and use $true$ to denote an empty precondition. Each sensing action, on the other hand, is defined by its precondition $prec(a)$, which is a conjunction of fluent literals, and $obs(a)$, which is the fluent literal that is observed by the sensing action.

A *system state* s is a set of fluent symbols, which intuitively defines all that is true in a particular state of the dynamical system. For a system state s, we define $M_s : F \rightarrow \{true, false\}$ as the truth assignment that assigns the truth value $true$ to p if $p \in s$, and assigns $false$ to p otherwise. We say a state s is *consistent* with a set of clauses $\mathcal{C}$, if $M_s \models c$, for every $c \in \mathcal{C}$.

We denote by S_0 the set of planning states consistent with the clauses of the initial state I. We say a dynamical system has a *complete initial state* iff $|S_0| = 1$; i.e., I has only one model. Σ has an *incomplete initial state* iff $|S_0| > 1$. We formalize the notion of conditional plans through *action trees* as follows.

Definition 1 (Action Tree) *Given a system $\Sigma = (F, A, \Omega, I)$, an action tree T is:*

- *ϵ (the empty tree); or*
- *aT', where $a \in A$, and T' is an action tree; or*
- *$a(T', T'')$, where $a \in \Omega$ and T' and T'' are action trees.*

We denote by $\mathcal{T}_\Sigma$ the set of action trees in Σ. Furthermore, we say that an action a is *executable* in a state s if $M_s \models$

[1] A paradigm related to conditional planning is *contingent planning*. Although originally understood to define the same class of problems as conditional planning, current research in contingent planning frequently sees contingent planning as an incremental process in which planning is intertwined with execution (see e.g., Brafman and Shani 2012). We thus stick here to conditional planning to emphasize that we are looking for a conditional plan in an offline manner.

$prec(a)$. If $a \in A$ is executable in a state s, we define its successor state as $\delta(a, s) = (s \setminus Del) \cup Add$, where Add contains a fluent f iff $C \rightarrow f$ is an effect of a and $M_s \models C$. On the other hand Del contains a fluent f iff $C \rightarrow \neg f$ is an effect of a, and $M_s \models C$.

Now we define how to compute the set of states that result from executing an action tree. To that end, we define the relation $\vdash_\Sigma$: $\mathcal{T}_\Sigma \times 2^F$ such that $(T, S) \vdash_\Sigma (T', S')$ intuitively means that if the agent is in any of the states in S, then performing one step of T results in being in some state in S', with action tree T' remaining. Formally,

- $(aT, S) \vdash_\Sigma (T, S')$ if a is executable in every state in S, and $S' = \{\delta(a, s) \mid s \in S\}$.
- $(a(T', T), S) \vdash_\Sigma (T, S')$ if a is executable in every state in S and $S' = \{s \in S \mid M_s \models obs(a)\}$ or $S' = \{s \in S \mid M_s \not\models obs(a)\}$.

We use $\vdash$ instead of $\vdash_\Sigma$ when the transition system is obvious from the context. We denote by $\vdash^*$ the reflexive and transitive closure of $\vdash$. This allows us to define what states result from executing an action tree in the initial state of a transition system.

Definition 2 (Resulting State) *Given a transition system Σ, state s is a resulting state from executing action tree T in Σ iff $(T, S_0) \vdash^* (\epsilon, S')$ and $s \in S'$.*

2.2 Classical, Conformant, and Conditional Planning

Below we define classes of planning tasks that have been studied extensively by the planning community. Deterministic (classical) planning is the most standard form of planning in which there is a unique initial state, and hence knowledge about the state of the world is always complete. In conformant planning, on the other hand, there is uncertainty but no observability, and actions that change the state of the world are deterministic; hence solutions to these tasks are sequences of actions. Finally, in conditional planning,[2] there is uncertainty about the initial state and the agent may observe the world through sensing actions. The resulting plan in this case is typically an action tree.

Definition 3 (Classes of Planning Tasks) *Given a set of literals G, and a system $\Sigma = (F, A, \Omega, I)$ we define the following classes of planning tasks:*

- (Σ, G) *is a* deterministic *or* classical *planning task if I defines a complete initial state and $\Omega = \emptyset$.*
- (Σ, G) *is a* conformant *planning task if I does not define a complete initial state and $\Omega = \emptyset$.*
- (Σ, G) *is a* conditional *planning task if I does not define a complete initial state and $\Omega \neq \emptyset$.*

Definition 4 (Plan) *Action tree T is a plan for planning task (Σ, G) iff for every state s_f that may result from the execution of T in Σ it holds that $M_{s_f} \models G$.*

[2]Early literature in conditional planning (e.g. Pryor and Collins 1996) assumed complete observability of the world. However, the current standard in the planning community—under which state-of-the-art solvers are developed—is to consider sensing actions as the only mechanism to observe the world (e.g. To, Pontelli, and Son 2011).

When T is a plan for a deterministic task, then we say T is a *deterministic plan*. Analogously, we use the terms *classical plan*, *conformant plan*, and *conditional plan*.

3 Characterizing Diagnoses

Automated diagnosis has long been a problem of interest to the AI community. Well-publicized early work focused on expert systems approaches as exploited in the medical diagnosis expert systems MYCIN (Shortliffe and Buchanan 1975). In the mid-1980's AI researchers turned their attention to model-based diagnosis. Early work in this area included Geneserth's DART system (Genesereth 1984), as well as GDE, the General Diagnosis Engine by de Kleer and Williams (de Kleer and Williams 1987) among others. In 1987 Reiter published his seminal paper formalizing the notion of *consistency-based diagnosis* and *minimal diagnosis*, which were predicated on a so-called first principles model of the normative behaviour of a system (Reiter 1987). This characterization defined a diagnosis to be a minimal set of components that must be designated as *abnormal* in order for observations of system behaviour to be logically consistent with the model (axiomatization) of the system. The exploitation of fault models and other aspects of system behaviour and function necessitated a more stringent characterization of diagnosis in terms of *abduction* (e.g., (Console and Torasso 2006), (Poole 1994)) in which faults or other explanations were posited in order to *entail* or otherwise explain observations. Such abductive diagnoses were originally conceived to work with fault models, rather than normative models, in order to entail faulty behaviour. de Kleer, Mackworth, and Reiter published a follow-on to Reiter's 1987 paper in 1992 that combined aspects of abductive and consistency based diagnosis into kernel diagnoses (de Kleer, Mackworth, and Reiter 1992). All of these characterizations of diagnosis related to static systems, described using a triple $(SD, COMPS, OBS)$ – the system description, a finite set of components to be diagnosed, and an observation. There was no notion of system dynamics – just a single state.

The systems we are concerned with in this paper include not only these static descriptions but also a rich theory of action that supports diagnostic planning in its many guises. We begin with a definition of such a diagnostic system.

- SD, the system description, a set of propositional sentences;
- $COMPS$, the components, a finite set of constants;
- OBS, the observation, a conjunction of ground literals:
- $\Sigma = (F, A, \Omega, I)$, a dynamical system that describes actions relevant to diagnostic problem solving tasks associated with the system described by SD.

We illustrate these with a simple example.

Example. Consider a flashlight, comprised of a battery and a switch. If the switch is on and both the battery and switch are operating normally, then light will be emitted. This fact can be described by the following logical formula, which is included in the system's description SD:

$$on \wedge \neg AB(battery) \wedge \neg AB(switch) \supset light.$$

a	$prec(a)$	Effect/Observation
$turn\text{-}on$	$\neg on$	on
$change\text{-}battery$	$true$	$\neg AB(battery)$
$fix\text{-}switch$	$true$	$\neg AB(switch)$
$sense\text{-}light$	$true$	$light$

Table 1: Preconditions and direct effects of the actions in our flashlight example.

The set of system components is simply defined by $COMPS = \{battery, switch\}$.

Now we assume the flashlight can be operated by a human user. The available actions are: turn on the switch, change the battery, and fix the switch. In addition, we have the ability to observe whether or not there is light in the room. The dynamics of these actions is described using a transition system Σ, where $F = \{on, AB(switch), AB(battery), light\}$, $A = \{turn\text{-}on, change\text{-}battery, fix\text{-}switch\}$, and $\Omega = \{sense\text{-}light\}$. The effects of the actions are described using our planning language as shown by Table 1. §

In general, the action theory described by Σ supports reasoning about actions to test, repair, or eradicate egregious behaviour in a system. On the other hand, the system description describes complex interactions between the components of the system.

Our notion of *diagnostic system*, which we define formally below, intuitively integrates the system description SD with the dynamics of the world described by Σ. In doing so, we need to address the so-called *ramification problem* – the problem of characterizing the indirect effects of actions. To see why this is, in our example, observe that performing the action $turn\text{-}on$ may have the indirect effect of $light$, under the condition $\neg AB(battery) \wedge \neg AB(switch)$.

The ramification problem is a well-studied problem in the KR community (e.g., Lin 1995; McCain and Turner 1995; Thielscher 1995; Sandewall 1996; Pinto 1999; Strass and Thielscher 2013) and has also been previously studied in the specific context of DPS (e.g., McIlraith 1997; McIlraith 2000; McIlraith and Scherl 2000). For the purposes of this paper, we adopt an existing solution to the ramification problem that compiles the indirect effects of actions into additional direct effects of actions proposed both by Pinto (1999) and in variation most recently by Strass and Thielscher (2013). The solution is predicated on augmentation of the constraints, SD, that additionally captures the causal relationship between fluents. These causal relationships have historically been captured via an explicit causal relation in the constraints (e.g., Lin 1995; McCain and Turner 1995) or by the augmentation of SD with a causal graph structure that ranges over the literals in SD (e.g., McIlraith 2000).

Example (continued). In our flashlight example, we consider the following ramification constraint, which is extracted directly from SD.

$$on \wedge \neg AB(battery) \wedge \neg AB(switch) \textbf{ causes } light.$$

a	New Effect
$turn\text{-}on$	$\neg AB(battery) \wedge \neg AB(switch) \rightarrow light$
$change\text{-}battery$	$on \wedge \neg AB(switch) \rightarrow light$
$fix\text{-}switch$	$on \wedge \neg AB(battery) \rightarrow light$

Table 2: Additional effects of actions in our example theory.

From there, we use Pinto's algorithm (1999) to compute additional effects for the actions in our theory. Some of the resulting effects are shown in Table 2. §

Now we provide a formal definition for a diagnostic system.

Definition 5 (Diagnostic System) *Given SD, $COMPS$, OBS, and $\Sigma = (F, A, \Omega, I)$, as defined above, a diagnostic system Σ_{SD} is a tuple (F', A', Ω, I') where:*

- *F' contains the elements in F and in $Vars(SD)$, the propositional variables in SD, and ground fluents of the form $AB(c)$ for every $c \in COMPS$;*
- *A' contain the same actions in A but augmented, following Pinto (1999), with conditional effects to address the ramification problem that emerges from the integration of Σ and SD;*
- *$I' = I \cup SD \cup OBS$.*

Note that the initial state I' contains both the system's description and the observation. In our example, we could have that $I = \{on\}$ and that $OBS = \{\neg light\}$. By including SD in I' we enforce that states consistent with I' have at least one abnormal component. Note that an alternative means of characterizing a diagnostic system is to treat ramifications as further actions that are triggered by the direct effects of actions.

We now formally define the notion of diagnosis, which we borrow from de Kleer, Mackworth, and Reiter (1992)'s, in which we posit a minimal subset of components that must be behaving abnormally in order to account for the observation in the initial state of our dynamical system. To facilitate explication, we appeal to a characterization of diagnosis in terms of abnormal components; however the work in this paper is applicable to a diversity of definitions of diagnoses or hypotheses and need not rely on the use of distinguished components. The generalization is straightforward. Recall that I', of Σ_{SD}, includes our observation, OBS of (potentially egregious) behaviour.

Definition 6 (Diagnosis) *Given a diagnostic system Σ_{SD}, $\Delta \subseteq COMPS$ is a diagnosis iff*

$$I' \cup \bigcup_{c \in \Delta} AB(c) \cup \bigcup_{c' \in COMPS \setminus \Delta} \neg AB(c')$$

is satisfiable.

Definition 7 (Minimal Diagnosis) *Δ is a minimal diagnosis of Σ_{SD} if Δ is a diagnosis and no other proper subset Δ' of Δ is a diagnosis.*

In previous work, we examined diagnosis of dynamical systems with respect to a narrative of observations that

evolved over a period of time. We characterized the notion of an *explanatory diagnosis* in the situation calculus (Sohrabi, Baier, and McIlraith 2010; McIlraith 1998) and relatedly the notion of an *explanation* in a planning-inspired propositional language (Sohrabi, Baier, and McIlraith 2011). These definitions of diagnosis and explanation conjecture a set of zero or more assumptions together with a sequence of actions to account for the observations. These definitions of diagnosis and explanation are not at odds with what is proposed here. Indeed, a diagnosis, Δ would hold in the final state of an explanatory diagnosis, with OBS as the final observation of the narrative of observations used to construct the explanatory diagnosis. The generation of explanatory diagnosis looks back in time to conjecture what happened based on past observations. Observations going forward are integrated with the diagnostic plans.

4 Diagnostic Planning

While automated diagnosis remains a well-studied area of research, we argue for a purposeful view of diagnosis. In particular, rather than generating candidate diagnoses and performing tests to identify a unique diagnosis, after which a course of action (amelioration/treatment) is determined, we argue that the determination of a unique diagnosis is generally not an end in itself and that a pragmatic view of DPS should focus on acting.

One argument in support of this viewpoint is that in many settings there are a limited number of courses of action that can achieve a particular goal. These actions/plans induce an equivalence class of diagnoses for which a particular course of action is relevant. For example, many families of bacterial infection require treatment with the same antibiotic and we need not completely discriminate the nature and location of the infection before treating. Similarly, if a photocopier is malfunctioning, despite the ability to perform an in-depth diagnosis, the first course of action is often to turn the machine off and on, resolving a large family of faults simultaneously without the need for time-consuming, and time-wasting in this case, differential diagnosis.

All of the diagnostic planning tasks we examine take the same general form. The dynamical model of the system, Σ_{SD}, is augmented with additional information about the initial state, the goal state and some constraints that need to be enforced throughout execution of the plan. In this section, we provide a general formulation of a diagnostic plan. We discuss the diversity of diagnostic tasks that can be achieved via the specification of different initial conditions, goals, and constraints, and the complexity of plan existence in some of these different settings. While the types of plans we are interested in here change the state of the world, we are also interested in plans that are designed to change our state of knowledge without necessarily changing (much of) the world. In the section that follows, we look at epistemic goals – planning to know whether or not a diagnosis is true, to refute a diagnosis, or to discriminate a collection of candidate diagnoses.

Definition 8 (Diagnostic Plan) *Given a diagnostic planning task* $(\Sigma_{SD}, Init, \Phi, G)$ *where*

- $\Sigma_{SD} = (F, A, \Omega, I)$*, is the diagnostic system;*
- $Init$*, is a set of logical formulae that provide additional information about the initial state;*
- Φ*, is a logical formula representing additional state constraints that must be enforced throughout the plan; and*
- G*, is a set of literals that prescribe the diagnostic planning goal,*

and where $Init \cup I \cup \Phi$ *is satisfiable.*

Action tree T *is a diagnostic plan for the diagnostic planning task* $((F, A, \Omega, I \cup Init), G)$ *under the constraint of* Φ *iff* T *is a plan for the planning task* $((F, A, \Omega, I \cup Init), G)$*, and for every* S *such that* $(T, S_0) \vdash^* (T', S)$ *it holds that* $M_s \models \Phi$*, for every* $s \in S$*.*

4.1 Properties of Diagnostic Plans

As evident from the definition presented, our characterization of a diagnostic plan is coupled to our previously defined classes of planning tasks. These classes differ with respect to the completeness of their initial states and whether they exploit sensing of any form. Such characteristics have implications with respect to the complexity of planning.

The following are the complexity classes of the decision problem for the different planning tasks.

Theorem 1 (Diagnostic Planning w/ Complete Info.) *Given a diagnostic planning problem* $(\Sigma_{SD}, Init, \Phi, G)$*, if* $I \cup Init \cup \Phi$ *defines a complete initial state and if* $\Omega = \emptyset$ *then this is a* classical *planning task and deciding whether there exists such a plan is PSPACE-complete.*

This result follows from Bylander's result on the complexity of deterministic planning (Bylander 1994).

Theorem 2 (Diagnostic Planning without Sensing) *Given a diagnostic planning problem* $(\Sigma_{SD}, Init, \Phi, G)$*, if* $\Omega = \emptyset$ *but* $I \cup Init \cup \Phi$ *does not define a complete initial state then this is a* conformant *planning problem and deciding whether there exists such a plan is EXPSPACE-complete.*

This follows from Haslum's result on the complexity of conformant planning (Haslum and Jonsson 1999). Finally,

Theorem 3 (Diagnostic Planning with Sensing) *Given a diagnostic planning problem* $(\Sigma_{SD}, Init, \Phi, G)$*, if* $I \cup Init \cup \Phi$ *does not define a complete initial state and* $\Omega \neq \emptyset$ *then this is a* conditional *planning problem and deciding whether there exists such a plan is 2-EXPTIME-complete.*

This follows from Rintanen's results on the complexity of conditional planning (Rintanen 2004).

4.2 The Many Guises of Diagnostic Planning

Our characterization of a diagnostic plan encompasses a diversity of DPS scenarios. Here we informally explore some of these varied scenarios in order to illustrate the broad utility of our characterization. Each scenario is realized by varying the $Init$ and G and can be performed with or without sensing as the scenario necessitates, and with commensurate implications regarding the type of planner.

Eradicate egregious behaviour By setting $Init = \emptyset$ and $G = \neg OBS$ we can plan to eradicate behaviour without

first explicitly computing a diagnosis. If sensing is permitted, sufficient information will be garnered to select an appropriate course of action. Such information may or may not entail a unique diagnosis as illustrated in some of the previous examples. Without sensing, a conformant plan will be generated (where possible) that will work regardless of what is wrong with the system.

Fix the system, given a diagnosis Given a diagnosis Δ, $Init = \{AB(c) \mid c \in \Delta\} \cup \{\neg AB(c) \mid c \in COMPS \setminus \Delta\}$ and $G = \bigwedge_{c \in \Delta} \neg AB(c)$. When a unique diagnosis has been determined, this can be added to the initial state and a plan generated accordingly. In such a case I may transform or be transformable into a classical planning problem, greatly diminishing the computational effort involved in generating a plan. $Init$ may also be used to capture the set of candidate diagnoses. (E.g., *The car won't start because it's either out of gas or the battery is dead.*) Even in a scenario such as this one, the best plan may require no sensing since this candidate diagnosis set may dictate the same "fix" – call for road-side assistance.

Assume a diagnosis and fix it or eradicate behaviour We can also use the planner to do what-if planning. Whereas the previous scenario used $Init$ to add facts to the initial state, it is often compelling to *assume* a particular likely diagnosis and generate a plan that will work predicated on the assumption being correct. (E.g., *Assume the battery is dead and fix it.*) A subset of candidate diagnoses may similarly be assumed.

Discriminate between different diagnoses Given two diagnoses Δ_1 and Δ_2, return a plan to determine which one of Δ_1 or Δ_2 may be causing the egregious behaviour. We require a notion of *epistemic goals*, which we define below, to formalize this as a planning problem.

Our definition of diagnostic plan also supports the enforcement of (safety) constraints. There are a number of compelling uses for such constraints, particularly in cases where a diagnosis is being assumed. In such a scenario, the user may wish to prescribe avoidance of things that would be fatal should the assumptions be flawed (e.g., giving penicillin to a patient with allergies). Constraints of this form can often be compiled away so as not to add to the difficulty of planning. Finally, the user can exercise flexibility in eliminating sensing actions to generate conformant rather than contingent plans, or to enforce Φ by eliminating the actions that would result in its violation.

5 Epistemic Diagnostic Planning

In the previous section we examined tasks to achieve some state of the world. Here we wish to generate plans to achieve epistemic goals. For example, we may wish to generate a plan to *know* a particular diagnosis (*E.g., I know Ralph has meningitis.*), to discriminate between diagnoses (*E.g., I know Ralph has one and only one of meningitis, strep-A, or influenza.*), or to eliminate a diagnosis (*E.g, I've eliminated the possibility that Ralph has strep-A.*).

The notion of planning and reasoning to achieve a state of knowledge dates back to work by Moore (1985). Scherl and Levesque (2003) later integrated Moore's approach with Reiter's solution to the frame problem (1991). More recently, epistemic planning has been discussed in the context of dynamic epistemic logic (e.g, (Herzig, Lang, and Marquis 2003; Andersen, Bolander, and Jensen 2012).

Currently, there are no competitive planning systems that implement the possible world semantics. There are a few systems that however implement the idea of *knowledge-level planning* (Demolombe and Pozos Parra 2000; Petrick and Bacchus 2002), in which the knowledge of the agent is explicitly represented as propositions of the language (e.g., using a fluent KF to represent that the agent knows F). These planners are capable of carrying out a limited, but still reasonably expressive form of reasoning about knowledge. A few systems like PKS (Petrick and Bacchus 2002) and the web-service composition planner by Pistore et al. (2005) adopt the approach of *planning at the knowledge level* to implement these ideas. Planning at the knowledge level may achieve good performance in many planning benchmarks however, their ability to reason about knowledge is limited. Along the same lines, Palacios and Geffner (2009) proposed a compilation technique that maps contingent planning into knowledge-level deterministic tasks. As expected, their translation is compact, sound, and complete for only a certain class of problems (Palacios and Geffner 2009).

While many simple epistemic diagnostic planning tasks can indeed be mapped into knowledge-level planning domains we propose an alternative compilation into conditional planning with sensing actions. Our motivation is practical since this enables computing diagnoses with a variety of existing planning systems.

Our translation still takes a somewhat impoverished view of the world, choosing not to appeal to rich modal theories of knowledge and belief in favour of adopting the stance of planning at the so-called *belief level*. In this view, the state of the system captures the agent's beliefs about the world. Specifically, in the initial state, we assume that the agent knows any formula ϕ that is such that for every s consistent with I it holds that $M_s \models \phi$. Similarly, when the agent performs actions, the agent knows all formulae that hold in all states that it could reach. Formally, given a set of states S, and a formula ϕ, we say that:

$$K(\phi, S) \text{ iff } M_s \models \phi \text{ for each } s \in S,$$

where the intuitive meaning of $K(\phi, S)$ is that the agent knows ϕ when the set of states it is possibly in is S.

With this definition in hand we are ready to define our notion of planning with epistemic goals.

Definition 9 (Epistemic Plan) *Let $\Sigma = (F, A, \Omega, I)$ be a transition system and S_0 be the set of all plan states consistent with I. Then,*

- *T is a plan for $\mathsf{Know}(\phi)$ iff for every S such that $(T, S_0) \vdash^* (\epsilon, S)$ it holds that $K(\phi, S)$,*
- *T is a plan for $\mathsf{KnowWhether}(\phi)$ iff for every S such that $(T, S_0) \vdash^* (\epsilon, S)$ it holds either $K(\phi, S)$ or $K(\neg\phi, S)$, and*
- *T is a plan for $\mathsf{Discriminate}(\phi, \psi)$ iff for every S such that $(T, S_0) \vdash^* (\epsilon, S)$ it holds either $K(\phi \wedge \neg\psi, S)$ or $K(\neg\phi \wedge \psi, S)$.*

Under this definition it is simple to prove that T is a plan for ϕ if and only if T it is an epistemic plan for $\mathsf{Know}(\phi)$. This is compatible with our notion of planning at the belief level: the knowledge of the agent is captured by the set of states the agent is at. This also means that if one wants a plan for $\mathsf{Know}(\phi)$, then one can obtain such a plan by querying a regular conditional planner for the goal ϕ. However, it is not immediately straightforward how to obtain plans for the other types of epistemic goals using a conditional planner. For example, using the non-epistemic goal $\phi \vee \neg\phi$ instead of $\mathsf{KnowWhether}(\phi)$ would not work since $\phi \vee \neg\phi$ is tautological and therefore achieved by every action tree.

Interestingly, it *is* possible to treat all types of epistemic goals as ontic planning. Below we propose a simple compilation which maps our notion of planning with epistemic goals to ontic conditional planning.

5.1 From Epistemic Planning to Conditional Planning

Given an epistemic goal of the form $\mathsf{KnowWhether}(\phi)$, the main idea underlying this compilation is to add an additional fluent $kw\text{-}\phi$ that can be used to replace the epistemic goal $\mathsf{KnowWhether}(\phi)$.

To simplify the presentation we will assume, for now, that the goal is $\mathsf{KnowWhether}(L)$, where L is a literal, and that as usual $\Sigma = (F, A, \Omega, I)$ is the transition system. The compiled planning problem is $\Sigma' = (F', A', \Omega, I)$, and F' and A' are generated by performing the following steps:

1. Let F' be $F \cup \{kw\text{-}L\}$.
2. Add to A' all actions in A plus actions $kw\text{-}act\text{-}pos\text{-}L$ and $kw\text{-}act\text{-}neg\text{-}L$, whose preconditions are, respectively, L and $\overline{L}$. Both actions have a single effect, $kw\text{-}L$.
3. For each action a in A' that contains either $C \rightarrow L$ or $C \rightarrow \overline{L}$ as a conditional effect, for some C different from $\{true\}$, we add the unconditional effect $\neg kw\text{-}L$ to a.

Note that Step 2 generates actions that add the fact $kw\text{-}L$. These actions can only be performed if the set of states the agent believes it is in, say, S, is such that $K(L, S)$ or $K(\neg L, S)$ (i.e., the agent knows whether L is true). Step 3 handles the case in which there may be "loss of knowledge" due to a conditional effect of an action. To see this, imagine a system with a single action A which is always executable and which has conditional effect $p \rightarrow L$. Assume I is such that the set of states consistent with I is $\{\{p\}, \{\}\}$. Even though the agent knows whether L in the states of I, it is not the case anymore after performing A, since the set of resulting states is $\{\{p, L\}, \{\}\}$.

We now prove that our proposed translation is sound and complete, in the senses defined below.

Theorem 4 (Completeness) *Let $\Sigma = (F, A, \Omega, I)$ be a transition system, and let Σ' be defined as above. If T is a plan for $(\Sigma, \mathsf{KnowWhether}(L))$ then there exists a plan T' for $(\Sigma', kw\text{-}L)$ which differs from T only in that it contains actions of the form $kw\text{-}act\text{-}pos\text{-}L$ or $kw\text{-}act\text{-}neg\text{-}L$ at the end of each of T''s branches.*

Proof. We note that if there exists an action tree T that is a plan for $(\Sigma, \mathsf{KnowWhether}(L))$ then we can construct a plan T' for $(\Sigma', kw\text{-}L)$ by simply adding an additional $kw\text{-}act\text{-}pos\text{-}L$ or $kw\text{-}act\text{-}neg\text{-}L$ as the final action in each branch of the tree. Such actions are executable at that point since it holds that $K(L, S)$ or $K(\neg L, s)$, where S is the set of states reached by that branch. ■

Theorem 5 (Soundness) *Let $\Sigma = (F, A, \Omega, I)$ be a transition system, and let Σ' be defined as above. If T is a plan for $(\Sigma', kw\text{-}L)$ then, by removing all actions of the form $kw\text{-}act\text{-}pos\text{-}L$ or $kw\text{-}act\text{-}neg\text{-}L$ from T we obtain an action tree that is a plan for $(\Sigma, \mathsf{KnowWhether}(L))$.*

Proof. Let T be a plan for $kw\text{-}L$. Take any S_f that results from the execution of a branch of T; i.e., such that $(T, S_0) \vdash^* (\epsilon, S_f)$. Observe that $kw\text{-}L \in S_f$. Now let (kT', S) be the configuration visited by the execution of the branch in which kwL is added for the last time (here k is either $kw\text{-}act\text{-}pos\text{-}L$ or $kw\text{-}act\text{-}neg\text{-}L$). In other words, let S be such that $(T, S_0) \vdash^* (kT', S) \vdash (T_1, S_1) \vdash (T_2, S_2) \vdash^* (T_n, S_n)$, with $T_n = \epsilon$, $S_n = S_f$, and such that for all $i \in \{1, \ldots, n\}$, $kw\text{-}L \in S_i$. Because k is executable in S, either $K(L, S)$ or $K(\neg L, S)$ holds. Furthermore, because S and S_1 differ in at most $kw\text{-}L$, it also holds that either $K(L, S_1)$ or $K(\neg L, S_1)$ holds. Now assume that it holds that $K(L, S_i)$ or $K(\neg L, S_i)$ for some $i \geq 1$. Since $kwL \in S_{i+1}$ the action that was performed in S_i to yield S_{i+1} was either an action that does not change the truth value of L or changes it unconditionally (in other words, it is not an action modified by Step 3 of the compilation). In either case it holds either $K(L, S_{i+1})$ or $K(\neg L, S_{i+1})$. We conclude that $K(L, S_n)$ or $K(\neg L, S_n)$, which means that T achieves $\mathsf{KnowWhether}(L)$. We observe now that if we remove all occurrences of $kw\text{-}act\text{-}pos\text{-}L$ or $kw\text{-}act\text{-}neg\text{-}L$ we obtain a plan that also achieves $\mathsf{KnowWhether}(L)$. ■

Now consider the epistemic goal $\mathsf{Discriminate}(L_1, L_2)$, where L_1 and L_2 are literals. The compilation follows the same intuitions from above. We generate a new transition system $\Sigma' = (F', A', \Omega, I)$, where and F' and A' are computed by performing the following steps:

1. Let F' be $F \cup \{disc\text{-}L_1\text{-}L_2\}$.
2. Add to A' all actions in A plus actions $disc\text{-}act\text{-}1\text{-}L_1\text{-}L_2$ and $disc\text{-}act\text{-}2\text{-}L_1\text{-}L_2$, whose preconditions are $\{L_1, \overline{L_2}\}$ and $\{\overline{L_1}, L_2\}$, respectively. Both actions have a single effect, $disc\text{-}L_1\text{-}L_2$.
3. For each action a in A' that contains $C \rightarrow L_1$, $C \rightarrow \overline{L_1}$, $C \rightarrow L_2$, or $C \rightarrow \overline{L_2}$ as a conditional effect, for some C, we add the unconditional effect $\neg disc\text{-}L_1\text{-}L_2$ to a.

This translation is also sound and complete. The proofs are similar to the ones presented above.

Theorem 6 (Completeness) *Let $\Sigma = (F, A, \Omega, I)$ be a transition system, and let Σ' be defined as above. If T is a plan for $(\Sigma, \mathsf{Discriminate}(L_1, L_2)))$ then there exists a plan T' for $(\Sigma', disc\text{-}L_1\text{-}L_2)$ which differs from T only in that it contains actions of the form $disc\text{-}act\text{-}1\text{-}L_1\text{-}L_2$ or $disc\text{-}act\text{-}2\text{-}L_1\text{-}L_2$ at the end of each of T''s branches.*

Theorem 7 (Soundness) *Let $\Sigma = (F, A, \Omega, I)$ be a transition system, and let Σ' be defined as above. If T is a plan for $(\Sigma', \mathsf{Discriminate}(L_1, L_2))$ then, by removing all*

actions of the form $disc\text{-}act\text{-}1\text{-}L_1\text{-}L_2$ *or* $disc\text{-}act\text{-}2\text{-}L_1\text{-}L_2$ *from* T *we obtain an action tree that is a plan for* $(\Sigma, \mathsf{Discriminate}(L_1, L_2))$.

Extending the Compilation to Formulae We have discussed how to compile goals of the form $\mathsf{KnowWhether}(L)$ and $\mathsf{Discriminate}(L_1, L_2)$, for the case of literals. To extend our compilation to general formulae we can use a pre-compilation step in which, for each formula ϕ involved in the epistemic goal, we generate the ramification constraints ϕ **causes** L_ϕ and $\neg\phi$ **causes** $\neg L_\phi$. Then we apply the compilation we described above. Soundness and completeness now follows from the theorems above together with soundness of the compilation of ramifications into effect axioms.

6 Computing Diagnostic Plans

The purpose of this paper was largely to define the theoretical underpinnings that support the exploitation of state-of-the-art AI planning systems for diagnostic problem solving. As such, our translation can be exploited by any planning system capable of handling conditional effects and sensing actions; in particular it can be used along with conditional planners such as Contingent-FF (Hoffmann and Brafman 2005), POND (Bryce, Kambhampati, and Smith 2006), and the CNF- and DNF-based conditional planners by To, Pontelli, and Son (2011).

Our translation could also be used with the latest contingent online planning systems CLG (Albore, Palacios, and Geffner 2009), CLG+ (Albore and Geffner 2009), and the SDR planner by (Brafman and Shani 2012), even though some of these systems may not return a complete conditional plan. However, for our experiments we wished to generate offline conditional plans and we opted to use Contingent-FF.

We developed 8 diagnostic planning problem scenarios in the following domains: an extension of the light-switch-battery domain given in the example presented in Section 3, an agent moving between rooms fixing light bulbs to complete a circuit, and embedded computer chips. We varied the complexity of the problems and ran them with Contingent-FF using a PC with an Intel Xeon 2.66 GHz processor with 4GB RAM running Linux. The simple problems generated reasonable plans, requiring from under 0.01 seconds to 0.03 seconds to complete. The more complex problems resulted in poor quality plans that favoured conformant rather than contingent solutions. We attribute much of this poor quality to the general effectiveness of the planner and how it's underlying heuristics interact with our approach to handling ramification constraints. We also experimented with encoding ramifications as additional actions rather than compiling them into effects. In future work, we will explore different representations for our ramification constraints across different planners.

An interesting result came from the problems created for the embedded computer chips domain, which consists of a collection of computer chips which themselves contain chips, and so forth, where all chips must be working normally for the output to be displayed. When allowing the planner to sense the status of and replace any chip, the resulting plan was always to replace the top level chips. This supports the notion of taking a purposeful view of diagnosis – it can be much faster to simply repair the issue than to determine the unique diagnosis. More generally, selecting the best course of purposeful actions can be informed by many factor including the cost of actions (sensing and world altering), time-criticality of a response, and issues of specificity or generality with respect to how a grouping of related faults should be addressed (e.g., fix them individually vs. doing a more general fix). The tradeoff between reasoning and acting, and the value of information has been addressed by a variety of researchers, notably Horvitz and Seiver (1997).

7 Related Work and Concluding Remarks

In addition to the literature on model-based diagnosis cited in Section 3, there is a body of previous work that relates diagnosis to theories of action in some guise. The relationship between actions and diagnoses was observed sometime ago by (e.g., Cordier and Thiébaux 1994; McIlraith 1994), while Sampath et al. (1995) were the first to present comprehensive results diagnosing discrete event systems via finite state automata. Thielscher (1997), McIlraith (1998), and subsequently Iwan (2001) and Baral, McIlraith, and Son (2000) cast the diagnosis problems in terms of an AI theory of action and change. More recently Grastien et al. (2007), Sohrabi, Baier, and McIlraith (2010), Lamperti and Zanella (2011) and Yu, Wen, and Liu (2013) have all addressed aspects of dynamical diagnosis.

Specifically in the area of diagnostic and repair planning, the most notable related work is that of Baral, McIlraith, and Son (2000) who introduced the notion of diagnostic and repair plans as conditional plans, and who have a treatment of causality and sensing. That work shares a number of intuitions with the work presented here, but without the focus on planning for epistemic goals, and the correspondence to conformant and contingent planning. Also of note is the work of Kuhn et al. (2008) who introduce the notion of *pervasive diagnosis* which produces diagnostic production plans that achieve production goals while coincidentally uncovering additional information about system health. The notion of combining diagnosis with repair has been addressed in varying fashion by Sun and Weld (1993) and by Friedrich, Gottlob, and Nejdl (1992) among others. Thiébaux et al. (1996), in this and later work, discuss the challenges of diagnosis and repair in the context of a power supply restoration problem, identifying the task as problem of planning with uncertainty. These works share intuitions with the approach advocated here, but do not reflect the advances in our collective understanding of the representational and computational issues associated with planning and sensing. Finally, outside the area of diagnostic problem solving there has been a variety of work looking at planning with some form of sensing. Of particular note is the work of Brenner and Nebel (2009) on MAPL, a continual planning system that interleaves planning with acting and sensing. This paradigm of planning and sensing is also one that is very amenable to diagnostic problem solving.

In this paper we have argued for and explored a purposeful view of diagnostic problem solving, examining the problem through the lens of AI automated planning. We have

characterized diagnostic planning with both ontic and epistemic goals, and established properties of these diagnostic planning tasks, including both complexity results and an understanding of their relationship to classical, conformant, and conditional planning systems. Of particular note was the characterization of diagnostic epistemic goals such as Discriminate and KnowWhether and their translation into planning problems with ontic goals. The correspondence with existing planning paradigms enables diagnostic planning to leverage ongoing advances in the development of non-classical planners. We discuss the exploitation of such planners, outlining our experience addressing diagnostic problem solving with Contingent-FF. Results to date are guardedly encouraging but expose the need for further investigation of the nuances of these planners as a complement to the results of this paper.

Beyond diagnostic problem solving, the work presented is relevant to a diversity of problems that involve generating hypotheses to conjecture system state, and sensing and acting in the world to discriminate those hypotheses or to purposefully effect change in response to observed behaviour. Some such problems include active vision applications, activity recognition, and goal recognition.

Acknowledgements: We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Albore, A., and Geffner, H. 2009. Acting in partially observable environments when achievement of the goal cannot be guaranteed. In *Proc. of ICAPS Workshop on Planning and Plan Execution for Real-World Systems*.

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *Proc. of the 21st Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 1623–1628.

Andersen, M. B.; Bolander, T.; and Jensen, M. H. 2012. Conditional epistemic planning. In *Proc. of the 13th European Conf. Logics in Artificial Intelligence (JELIA)*, volume 7519 of *Lecture notes in Computer Science*. Springer. 94–106.

Aucher, G., and Bolander, T. 2013. Undecidability in epistemic planning. In *Proc. of the 23rd Int'l Joint Conference on Artificial Intelligence (IJCAI)*.

Baral, C.; McIlraith, S.; and Son, T. 2000. Formulating diagnostic problem solving using an action language with narratives and sensing. In *Proc. of the 7th Int'l Conference on Knowledge Representation and Reasoning (KR)*, 311–322.

Brafman, R. I., and Shani, G. 2012. Replanning in domains with partial information and sensing actions. *Journal of Artificial Intelligence Research* 45:565–600.

Brenner, M., and Nebel, B. 2009. Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems* 19(3):297–331.

Bryce, D.; Kambhampati, S.; and Smith, D. E. 2006. Planning graph heuristics for belief space search. *J. Artif. Intell. Res. (JAIR)* 26:35–99.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1-2):165–204.

Console, L., and Torasso, P. 2006. Automated diagnosis. *Intelligenza Artificiale* 3(1-2):42–48.

Cordier, M.-O., and Thiébaux, S. 1994. Event-based diagnosis of evolutive systems. In *Proc. of the 5th Int'l Workshop on Principles of Diagnosis (DX)*, 64–69.

de Kleer, J., and Williams, B. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32:97–130.

de Kleer, J.; Mackworth, A.; and Reiter, R. 1992. Characterizing diagnoses and systems. *Artificial Intelligence* 56(2–3):197–222.

Demolombe, R., and Pozos Parra, M. d. P. 2000. A simple and tractable extension of situation calculus to epistemic logic. In *Proc. of the 12th International Symposium on Foundations of Intelligent Systems*, volume 1932 of *Lecture notes in Computer Science*. Springer. 515–524.

Friedrich, G.; Gottlob, G.; and Nejdl, W. 1992. Formalizing the repair process. In *Proc. of the 10th European Conference on Artificial Intelligence (ECAI)*, 709–713.

Genesereth, M. 1984. The use of design descriptions in automated diagnosis. *Artificial Intelligence* 24:411–436.

Grastien, A.; Anbulagan; Rintanen, J.; and Kelareva, E. 2007. Diagnosis of discrete-event systems using satisfiability algorithms. In *Proc. of the 22nd AAAI Conference on Artificial Intelligence (AAAI)*, 305–310.

Haslum, P., and Jonsson, P. 1999. Some results on the complexity of planning with incomplete information. In *Proc. of the 5th European Conference on Planning (ECP)*, 308–318.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Herzig, A.; Lang, J.; and Marquis, P. 2003. Action representation and partially observable planning using epistemic logic. In *Proc. of the 18th Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 1067–1072.

Hoffmann, J., and Brafman, R. I. 2005. Contingent planning via heuristic forward search witn implicit belief states. In *Proc. of the 15th Int'l Conference on Automated Planning and Scheduling (ICAPS)*, 71–80.

Horvitz, E., and Seiver, A. 1997. Time-critical reasoning: Representations and application. In *Proceedings of the Thirteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, 250–257. San Francisco, CA: Morgan Kaufmann.

Iwan, G. 2001. History-based diagnosis templates in the framework of the situation calculus. In *Proc. of the Joint German/Austrian Conference on Artificial Intelligence (KR/ÖGAI)*, 244–259.

Kuhn, L. D.; Price, B.; de Kleer, J.; Do, M. B.; and Zhou, R. 2008. Pervasive diagnosis: The integration of diagnostic goals into production plans. In *Proc. of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, 1306–1312.

Lamperti, G., and Zanella, M. 2011. Context-sensitive diagnosis of discrete-event systems. In *Proc. of the 22nd Int'l*

Joint Conference on Artificial Intelligence (IJCAI), 969–975.

Lin, F. 1995. Embracing causality in specifying the indirect effects of actions. In *Proc. of the 14th Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 1985–1991.

McCain, N., and Turner, H. 1995. A causal theory of ramifications and qualifications. In *Proc. of the 14th Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 1978–1984.

McIlraith, S. A., and Scherl, R. B. 2000. What sensing tells us: Towards a formal theory of testing for dynamical systems. In *Proc. of the 17th National Conference on Artificial Intelligence (AAAI)*, 483–490.

McIlraith, S. 1994. Towards a theory of diagnosis, testing and repair. In *Proc. of the 5th Int'l Workshop on Principles of Diagnosis (DX)*, 185–192.

McIlraith, S. 1997. Representing actions and state constraints in model-based diagnosis. In *Proc. of the 14th National Conference on Artificial Intelligence (AAAI)*, 43–49.

McIlraith, S. 1998. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proc. of the 6th Int'l Conference of Knowledge Representation and Reasoning (KR)*, 167–179.

McIlraith, S. 2000. Integrating actions and state constraints: A closed-form solution to the ramification problem (sometimes). *Artificial Intelligence* 116(1-2):87–121.

Moore, R. C. 1985. A Formal Theory of Knowledge and Action. In Hobbs, J. B., and Moore, R. C., eds., *Formal Theories of the Commonsense World*. Norwood, New Jersey: Ablex Publishing Corp. chapter 9, 319–358.

Palacios, H., and Geffner, H. 2006. Compiling uncertainty away: Solving conformant planning problems using a classical planner (sometimes). In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI)*.

Palacios, H., and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research* 35:623–675.

Pednault, E. P. D. 1989. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proc. of the 1st Int'l Conference of Knowledge Representation and Reasoning (KR)*, 324–332.

Petrick, R. P. A., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. of the 6th Int'l Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, 212–222.

Pinto, J. 1999. Compiling ramification constraints into effect axioms. *Computational Intelligence* 15:280–307.

Pistore, M.; Marconi, A.; Bertoli, P.; and Traverso, P. 2005. Automated composition of web services by planning at the knowledge level. In *Proc. of the 19th Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 1252–1259.

Poole, D. 1994. Representing diagnosis knowledge. *Ann. Math. Artif. Intell.* 11(1-4):33–50.

Pryor, L., and Collins, G. 1996. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research* 4:287–339.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32(1):57–95.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*. San Diego, CA: Academic Press. 359–380.

Rintanen, J. 2004. Complexity of planning with partial observability. In *Proc. of the 14th Int'l Conference on Automated Planning and Scheduling (ICAPS)*, 345–354.

Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; and Teneketzis, D. 1995. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control* 40(9):1555–1575.

Sandewall, E. 1996. Assessments of ramification methods that use static domain constraints. In *Proc. of the 5st Int'l Conference of Knowledge Representation and Reasoning (KR)*, 99–110.

Scherl, R., and Levesque, H. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence* 144(1–2):1–39.

Shortliffe, E. H., and Buchanan, B. G. 1975. A model of inexact reasoning in medicine. *Mathematical Biosciences* 23(34):351 – 379.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. 2010. Diagnosis as planning revisited. In *Proc. of the 12th Int'l Conference on the Principles of Knowledge Representation and Reasoning (KR)*, 26–36.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. 2011. Preferred explanations: Theory and generation via planning. In *Proc. of the 25th AAAI Conference on Artificial Intelligence (AAAI)*.

Strass, H., and Thielscher, M. 2013. A general first-order solution to the ramification problem with cycles. *Journal of Applied Logic* 11(3):289–308.

Sun, Y., and Weld, D. S. 1993. A framework for model-based repair. In *Proc. of the 11th National Conference on Artificial Intelligence (AAAI)*, 182–187.

Thiébaux, S.; Cordier, M.-O.; Jehl, O.; and Krivine, J.-P. 1996. Supply restoration in power distribution systems: A case study in integrating model-based diagnosis and repair planning. In *Proc. of the 12th Conference in Uncertainty in Artificial Intelligence (UAI)*, 525–532.

Thielscher, M. 1995. Computing ramifications by postprocessing. In *Proc. of the 14th Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 1994–2000.

Thielscher, M. 1997. A theory of dynamic diagnosis. *Electronic Transactions on Artificial Intelligence* 1:73–104.

To, S. T.; Pontelli, E.; and Son, T. C. 2011. On the effectiveness of CNF and DNF representations in contingent planning. In *Proc. of the 22nd Int'l Joint Conference on Artificial Intelligence (IJCAI)*, 2033–2038.

Yu, Q.; Wen, X.; and Liu, Y. 2013. Multi-agent epistemic explanatory diagnosis via reasoning about actions. In *Proc. of the 23rd Int'l Joint Conference on Artificial Intelligence (IJCAI)*.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Satisfiability of Alternating-Time Temporal Epistemic Logic Through Tableaux

Francesco Belardinelli
Laboratoire IBISC – Université d'Evry

Abstract

In this paper we present a tableau-based method to decide the satisfiability of formulas in ATEL, an extension of the alternating-time temporal logic ATL including epistemic modalities for individual knowledge. Specifically, we analyse satisfiability of ATEL formulas under a number of conditions. We evaluate the assumptions of synchronicity and of a unique initial state, which have been proposed in the context of Interpreted Systems. Also, we consider satisfiability at an initial state as opposed to any state in the system. We introduce a tableau-based decision procedure for each of these combinations. Moreover, we adopt an agent-based approach to satisfiability, namely, the decision procedure returns a set of agents inducing a concurrent game structure that satisfies the relevant specification.

1 Introduction

Formal languages for knowledge reasoning and representation find increasingly application to the analysis of concurrent and reactive systems (van Harmelen, Lifschitz, and Porter 2007). In particular, multi-modal epistemic logics (Fagin et al. 1995; Meyer and Hoek 1995) have proved to be an invaluable tool for the specification and verification of systems of autonomous agents, especially in combination with temporal operators (Gammie and van der Meyden 2004; Lomuscio, Qu, and Raimondi 2009; Kacprzak et al. 2008). Among these multi-agent logics, the alternating-time temporal epistemic logic ATEL has been proposed as a formal language to represent and reason about individual and group strategies, as well as the knowledge thereof (van der Hoek and Wooldridge 2003; Jamroga and van der Hoek 2004). The theoretical properties of ATEL have been thoroughly investigated, also w.r.t. the assumptions of imperfect information and commitment in strategies, thus producing a wealth of results (Agotnes, Goranko, and Jamroga 2007; Bulling, Dix, and Jamroga 2010).

This paper aims to make a novel contribution to the formal analysis of logics of knowledge and strategies by introducing a tableau-based decision procedure for ATEL formulas. Tableaux are a well-established technique to decide satisfiability, actively researched both in the realm of logic and in theoretical computer science (D'Agostino et al. 1999). Recently, a stream of papers has appeared on tableau methods for various flavours of temporal logics as well as multi-agent epistemic logics (Goranko and Shkatov 2009a; 2009b; 2009c; Ajspur, Goranko, and Shkatov 2013). While we explicitly acknowledge the influence of these works and make use of part of their formal machinery, we substantially extend the object of investigation. Specifically, our work differs from the contributions above in three ways. Firstly, we adopt an agent-based perspective and consider agents as the basic components of our epistemic concurrent game models. This means that the decision procedure returns not just a model satisfying the formula if successful, instead a system of agents is provided. Secondly, we follow the paradigm of Interpreted Systems (Fagin et al. 1995) and consider forms of interaction between the temporal and epistemic dimensions. In particular, we introduce epistemic concurrent game models which are synchronous or have a unique initial state. Thirdly, we analyse two different notions of satisfiability, namely satisfiability in some initial state, as opposed to satisfiability in any state. We maintain that the former notion is typical in the modelling and verification of concurrent systems (Baier and Katoen 2008), while the latter has traditionally been studied in mathematical logic (Blackburn, de Rijke, and Venema 2001). We will see that all these choices do have an impact on tableaux construction.

The motivation for the present work comes also from the fact that, besides the theoretical interest of algorithmic decision techniques, tableaux for ATEL can in principle be used to synthesize agent systems capable of enforcing behaviours specified as ATEL formulas. Thus, the investigations carried out hereafter can be seen as a preliminary contribution to bridge the gap between knowledge representation and model synthesis.

Related Work. This contribution builds on a series of papers on tableaux for multi-agent modal logics. Specifically, (Goranko and Shkatov 2009a) puts forward incremental tableaux for (non-epistemic) ATL; while in (Ajspur, Goranko, and Shkatov 2013) an epistemic logic with group knowledge is considered. In (Goranko and Shkatov 2009b; 2009c) the linear- and branching-time temporal epistemic logics LTLK and CTLK are given tableau-based decision procedures. However, we extend the object of investigation as detailed above. In (Walther 2005) tableaux for ATEL are

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

presented, but their construction is top-down, while we follow the references above and provide incremental tableaux. This point is crucial because, even though in the worst case the computational complexity remains the same, the incremental procedure needs not to build all formula types. Hence, for the purpose of implementation its performance is, on average, better. On a related subject, we remark that in (Goranko and Jamroga 2004) a translation from ATEL to ATL is provided, but at the cost of an exponential blow-up.

Scheme of the paper. In Section 2 we present the epistemic concurrent game models and the logic ATEL; while Section 3 introduces the Hintikka structures for ATEL. In Section 4 we define the tableaux for satisfiability in any state of the model; while Section 5 is dedicated to satisfiability at initial states. We conclude in Section 6 and point to future work. For reasons of space we provide sketches of proofs only in selected cases. An extended version with some more details is available for the reader's perusal (Belardinelli 2014).

2 Epistemic Concurrent Game Models

In this section we present the formal machinery that will be used in the rest of the paper. We first introduce the epistemic concurrent game models, inspired to (van der Hoek and Wooldridge 2003; Jamroga and van der Hoek 2004), by adopting an agent-based perspective. Then, we consider the alternating-time epistemic logic ATEL. We start with the notion of *agent*.

Definition 1 (Agent) *An* agent *is a tuple* $i = \langle L_i, Act_i, Pr_i \rangle$ *where*

- L_i *is a set of* local states $l_i, l_i', \ldots$;
- Act_i *is the finite set of* actions $\sigma_i, \sigma_i', \ldots$;
- $Pr_i : L_i \mapsto 2^{Act_i}$ *is the* protocol function.

Intuitively, at each moment the agent i is situated in some local state $l_i \in L_i$ representing all the information she has. In this respect we follow the typical approach to Multi-agent Systems (Fagin et al. 1995; Wooldridge 2000). Also, we assume that agents perform the actions in Act_i according to the protocol function Pr_i.

As we are interested in agent interactions, we introduce their composition. Given a set $Ag = \{i_0, \ldots, i_n\}$ of agents, we define the set L of *global states* $s, s', \ldots$ (resp. the set Act of *joint actions* $\sigma, \sigma', \ldots$) as the cartesian product $L_0 \times \ldots \times L_n$ (resp. $Act_0 \times \ldots \times Act_n$). In the rest of the paper we fix a set AP of atomic propositions and represent the j-th element of a tuple $t = \langle t_1, \ldots, t_n \rangle$ as t_j or $t(j)$.

Definition 2 (ECGM) *Given a set* $Ag = \{i_0, \ldots, i_n\}$ *of agents* $i = \langle L_i, Act_i, Pr_i \rangle$*, an* epistemic concurrent game model *is a tuple* $\mathcal{P} = \langle Ag, I, \tau, \pi \rangle$ *where*

- $I \subseteq L$ *is the set of* initial global states;
- $\tau : L \times Act \mapsto L$ *is the* global transition function*, where* $\tau(s, \sigma)$ *is defined iff* $\sigma_i \in Pr_i(l_i)$ *for every* $i \in Ag$;
- $\pi : AP \to 2^L$ *is an* interpretation *of atomic propositions.*

Intuitively, an ECGM evolves from an initial state in I as specified by the global transition function τ, which returns a successor state for each enabled joint action σ. In line with the tradition of Interpreted Systems (Fagin et al. 1995), the agent i_0 will be reserved to model the environment the agents act in. That is, i_0 will encode all the relevant information about the system that does not appear in the private local state of any agent. In this capacity we denote i_0 as Env. Further, we observe that standard notions of concurrent game models can be defined in the present framework. Specifically, given a global state $s = \langle l_0, \ldots, l_n \rangle$ and an agent i, we introduce the functions $D_i(s) = Pr_i(l_i)$ for the set of enabled actions, $d_i(s) = |Pr_i(l_i)|$ for the number of enabled actions, and $D(s) = \{\sigma \in Act \mid \text{for every } i \in Ag, \sigma_i \in Pr_i(l_i)\} = D_0(s) \times \ldots \times D_n(s)$ for the set of joint enabled actions (Alur, Henzinger, and Kupferman 2002; Goranko and Shkatov 2009a).

We now fix some more notation. The *transition relation* $\to$ on global states is defined so that $s \to s'$ if there exists $\sigma \in Act$ and $s \xrightarrow{\sigma} s'$, i.e., $\tau(s, \sigma) = s'$. We will also consider the transitive closure $\to^*$. A *run* λ from a state s, or s-run, is an infinite sequence $s^0, s^1, \ldots$ s.t. $s_i \to s_{i+1}$ and $s^0 = s$. For $n, m \in \mathbb{N}$, with $n \leq m$, we define $\lambda(n) = s^n$ and $\lambda[n, m] = s^n, s^{n+1}, \ldots, s^m$. A state s' is *reachable from* s iff $s \to^* s'$. Now, $\mathcal{R}$ is the set of all s_0-runs, for some initial state $s_0 \in I$, and $S = \{\lambda(n) \mid \lambda \in \mathcal{R}, n \in \mathbb{N}\}$ is the set of states reachable from any initial state. Further, let $\sharp$ be a placeholder for arbitrary actions. Given a group $A \subseteq Ag$ of agents, an *A-action* σ_A is an $|Ag|$-tuple s.t. *(i)* $\sigma_A(i) \in Act_i$ for $i \in A$, and *(ii)* $\sigma_A(j) = \sharp$ for $j \notin A$. We denote by Act_A the set of all such A-actions and $D_A(s) = \{\sigma_A \in Act_A \mid \text{for every } i \in A, \sigma_A(i) \in Pr_i(l_i)\}$. A joint action σ *extends* an A-action σ_A, or $\sigma_A \sqsubseteq \sigma$, iff $\sigma_A(i) = \sigma(i)$ for all $i \in A$. The *outcome of an A-action σ_A at a state s*, denoted by $out(s, \sigma^A)$, is the set of all states s' for which there exists a joint action $\sigma \in Act$ s.t. $\sigma_A \sqsubseteq \sigma$ and $\tau(s, \sigma) = s'$. As for the epistemic component of ECGM, two global states $s = \langle l_0, \ldots, l_n \rangle$ and $s' = \langle l_0', \ldots, l_n' \rangle$ are *indistinguishable* for agent i, written $s \sim_i s'$, iff $l_i = l_i'$ (Fagin et al. 1995).

In this paper we consider two conditions on ECGM.

Definition 3 *An ECGM* $\mathcal{P}$

- *is* synchronous *iff for every* $\lambda, \lambda' \in \mathcal{R}$*,* $i \in Ag$*,* $n \in \mathbb{N}$*, if* $\lambda(n) \sim_i \lambda'(n')$ *then* $n = n'$;
- *has a* unique initial state *iff* $|I| = 1$.

Synchronicity and uniqueness of initial state are typically considered in the context of Interpreted Systems, where their impact on axiomatisations and the satisfiability problem has been thoroughly assessed (Halpern and Vardi 1986; 1989; Halpern, van der Meyden, and Vardi 2004). The meaning of a unique initial state is intuitively clear; while in synchronous ECGM the agents are assumed to have access to a global clock. Hereafter, we denote the class of all ECGM satisfying either uniqueness of initial state or synchronicity with the superscripts uis and $sync$. Hence, for instance, $ECGM^{uis,sync}$ is the class of all synchronous ECGM with a unique initial state.

We now present the alternating-time epistemic logic ATEL as a specification language for ECGM.

Definition 4 (ATEL) *The ATEL formulas* φ *are defined in BNF as follows:*

$\varphi ::= p \mid \neg\varphi \mid \varphi \to \varphi \mid \langle\langle A\rangle\rangle X\varphi \mid \langle\langle A\rangle\rangle G\varphi \mid \langle\langle A\rangle\rangle \varphi U \varphi \mid K_i\varphi$

where $A \subseteq Ag$, $i \neq Env$, *and* $Env \notin A$.

The language ATEL is an extension of the alternating-time logic ATL enriched with an epistemic operator K_i for each agent different from the environment (van der Hoek and Wooldridge 2003). Also, the environment Env does not appear in coalition modalities. This is once more in line with the semantics of Interpreted Systems, where the environment is seen as adversarial w.r.t. the other agents (Fagin et al. 1995). The ATEL formulas *(i)* $\langle\langle A\rangle\rangle X\varphi$, *(ii)* $\langle\langle A\rangle\rangle G\varphi$ and *(iii)* $\langle\langle A\rangle\rangle \varphi U \varphi'$ are read as "the agents in A have a strategy to..." *(i)* "...enforce φ at the next state", *(ii)* "...always enforce φ", and *(iii)* "...enforce φ until φ'"; while $\langle\langle A\rangle\rangle F\varphi$ is defined as $\langle\langle A\rangle\rangle \top U \varphi$. The epistemic formula $K_i\varphi$ intuitively means that "agent i knows φ". For ease of presentation we do not consider operators for group knowledge, even if we envisage such an extension. In what follows we define $(\langle\langle A\rangle\rangle X)^k\phi$ by induction on k as *(i)* $(\langle\langle A\rangle\rangle X)^0\phi = \phi$, and *(ii)* $(\langle\langle A\rangle\rangle X)^{k+1}\phi = (\langle\langle A\rangle X)^k\langle\langle A\rangle\rangle X\phi$. Also, the positive (resp. negative) X-formulas have the form $\langle\langle A\rangle\rangle X\phi$ (resp. $\neg\langle\langle A\rangle\rangle X\phi$).

To intepret ATEL formulas on ECGM we need to introduce the notion of a *strategy* for a set A of agents. Let γ be an ordinal with $1 \leq \gamma \leq \omega$, a *$\gamma$-recall A-strategy* is a mapping $F_A[\gamma] : \bigcup_{1\leq n<1+\gamma} S^n \mapsto \bigcup_{s\in S} D_A(s)$ s.t. $F_A[\gamma](\kappa) \in D_A(lst(\kappa))$ for every $\kappa \in \bigcup_{1\leq n<1+\gamma} S^n$, where $lst(\kappa)$ is the last element of κ and $1 + \gamma = \gamma$ for $\gamma = \omega$. Intuitively, a strategy returns an enabled A-action for every finite sequence of states. We remark that, according to standard terminology in concurrent game models (Bulling, Dix, and Jamroga 2010), the agents in ECGM have *perfect information*, that is, their strategies are determined by all information available at each global state. Further, the *outcome* of strategy $F_A[\gamma]$ at state s, or $out(s, F_A[\gamma])$, is the set of all s-runs λ s.t. $\lambda(i+1) \in out(\lambda(i), F_A[\gamma](\lambda[j,i]))$ for all $i \geq 0$ and $j = max(i - \gamma + 1, 0)$. Depending on γ, we can define positional strategies, perfect-recall strategies, etc. Since these distinctions are not relevant for the present discussion, as they all generate the same class of models, hereafter we assume that γ is fixed and omit it.

Definition 5 (Semantics of ATEL) *We define whether an ECGM $\mathcal{P}$* satisfies *a formula φ at state s, or $(\mathcal{P}, s) \models \varphi$, as follows:*

$(\mathcal{P}, s) \models p$ *iff* $s \in \pi(p)$
$(\mathcal{P}, s) \models \neg\varphi$ *iff* $(\mathcal{P}, s) \not\models \varphi$
$(\mathcal{P}, s) \models \varphi \to \varphi'$ *iff* $(\mathcal{P}, s) \not\models \varphi$ *or* $(\mathcal{P}, s) \models \varphi'$
$(\mathcal{P}, s) \models \langle\langle A\rangle\rangle X\varphi$ *iff there is an A-strategy F_A s.t. for all $\lambda \in out(s, F_A)$, $(\mathcal{P}, \lambda(1)) \models \varphi$*
$(\mathcal{P}, s) \models \langle\langle A\rangle\rangle G\varphi$ *iff there is an A-strategy F_A s.t. for all $\lambda \in out(s, F_A)$, $i \geq 0$, $(\mathcal{P}, \lambda(i)) \models \varphi$*
$(\mathcal{P}, s) \models \langle\langle A\rangle\rangle \varphi U \varphi'$ *iff there is an A-strategy F_A s.t. for all $\lambda \in out(s, F_A)$ there is $k \geq 0$ s.t. $(\mathcal{P}, \lambda(k)) \models \varphi'$, and for all j, $0 \leq j < k$ implies $(\mathcal{P}, \lambda(j)) \models \varphi$*
$(\mathcal{P}, s) \models K_i\varphi$ *iff for all $s' \in S$, $s \sim_i s'$ implies $(\mathcal{P}, s') \models \varphi$*

As to the notion of satisfaction, multiple choices are possible. In mathematical logic satisfaction typically takes into account all the states in a system (Blackburn, de Rijke, and Venema 2001); while in the modelling of concurrent systems satisfaction w.r.t. initial states only is also considered (Baier and Katoen 2008). This remark motivates the following definition.

Definition 6 (Satisfaction) *An ATEL-formula θ is* S-satisfied *(resp.* I-satisfied*) in a ECGM $\mathcal{P}$ iff there exists $s \in S$ (resp. $s \in I$) s.t. $(\mathcal{P}, s) \models \theta$.*

While for the class of all ECGM the two notions are equivalent, under the assumptions of a unique initial state or synchronicity this is not longer the case. For instance, the ATEL-formula $\langle\langle\emptyset\rangle\rangle G\phi \wedge \langle\langle A\rangle\rangle F\neg K_i\phi$ is not I-satisfiable in $ECGM^{uis}$, but it is S-satisfiable in the same class. Similarly, the ATEL-formula $(\langle\langle\emptyset\rangle\rangle X)^k\phi \wedge (\langle\langle A\rangle\rangle X)^k\neg K_i\phi$ is not I-satisfiable in $ECGM^{uis,sync}$, but it is S-satisfiable. Indeed, for ECGM with a unique initial state, the operator $\langle\langle\emptyset\rangle\rangle G$ acts as a universal modality, traversing the whole state space. We maintain that both notions of satisfaction are of interest and will analyse both. In the following lemma we report the equivalences between classes of formulas, where, for instance, $ATEL_I^{uis}$ is the set of ATEL-formulas I-satisfiable in the class $ECGM^{uis}$.

Lemma 1 *The classes of ATEL formulas in different boxes are different.*

$ATEL_S = ATEL_S^{uis} = ATEL_S^{sync} = ATEL_S^{sync,uis}$ $= ATEL_I = ATEL_I^{sync}$
$ATEL_I^{uis}$
$ATEL_I^{sync,uis}$

By Lemma 1 we have to consider three distinct cases: $ATEL_S$, $ATEL_I^{uis}$, and $ATEL_I^{sync,uis}$, which will be the main focus of the paper. Hereafter we simply refer to *satisfaction* to indicate any of the two notions above. Moreover, satisfiability can be also considered w.r.t. different sets of agents (Walther et al. 2006). Let Ag_θ be the set of agents mentioned in an ATEL-formula θ, while $Ag_\theta^+ = Ag_\theta \cup \{Env\}$ (remember that Env does appear in θ.)

Definition 7 *An ATEL-formula θ is*

1. Ag-satisfiable*, for some $Ag \supseteq Ag_\theta^+$, iff θ is satisfiable in an ECGM $\mathcal{P} = \langle Ag, I, \tau, \pi\rangle$;*
2. generally satisfiable *iff θ is satisfiable in an ECGM $\mathcal{P} = \langle Ag, I, \tau, \pi\rangle$ for some $Ag \supseteq Ag_\theta^+$.*

By "absorbing" the agents in $Ag \setminus Ag_\theta$ in the environment and by adding dummy agents, we can now prove the following result that extends Corollary 2.25 in (Goranko and Shkatov 2009a) to ATEL.

Theorem 2 *Every ATEL-formula θ is generally satisfiable iff it is Ag_θ^+-satisfiable.*

As a result, it is sufficient to consider Ag_θ^+-satisfiability when checking for models of ATEL formulas.

3 Hintikka Structures for ECGM

Tableau-based methods do not provide directly the formula in question with a relevant model (ECGM in the present case), but they first go through the construction of *Hintikka*

α	α_1	α_2
$\neg\neg\phi$	ϕ	ϕ
$\neg(\phi \to \psi)$	ϕ	$\neg\psi$
$\langle\langle A\rangle\rangle G\phi$	ϕ	$\langle\langle A\rangle\rangle X\langle\langle A\rangle\rangle G\phi$
$K_i\phi$	ϕ	$K_i\phi$

β	β_1	β_2
$\phi \to \psi$	$\neg\phi$	ψ
$\langle\langle A\rangle\rangle(\phi U\psi)$	ψ	$\phi \wedge \langle\langle A\rangle\rangle X\langle\langle A\rangle\rangle(\phi U\psi)$
$\neg\langle\langle A\rangle\rangle(\phi U\psi)$	$\neg\phi \wedge \neg\psi$	$\neg\psi \wedge \neg\langle\langle A\rangle\rangle X\langle\langle A\rangle\rangle(\phi U\psi)$
$\neg\langle\langle A\rangle\rangle G\phi$	$\neg\phi$	$\neg\langle\langle A\rangle\rangle X\langle\langle A\rangle\rangle G\phi$

Figure 1: α- and β-formulas in ATEL.

structures. The main difference between ECGM and Hintikka structures is that, while the former define truth conditions for every ATEL formula on the set AP of atomic propositions, the latter is only concerned with formulas 'related' to the satisfaction of the formula to be checked. To specify this relation between ATEL formulas, we introduce the distinction between α- and β-formulas of ATEL in Fig. 1. Intuitively, the states in a Hintikka structures are sets of formulas that contain α_1 *and* α_2 (resp. β_1 *or* β_2) whenever they contain α (resp. β), according to the semantics of the main logical symbol. This idea is made precise by the following definition.

Definition 8 *A set Δ of ATEL formulas is*

- consistent *if it does not contain both ψ and $\neg\psi$.*
- downward saturated *if*
  1. *$\alpha \in \Delta$ implies $\alpha_1 \in \Delta$ and $\alpha_2 \in \Delta$;*
  2. *$\beta \in \Delta$ implies $\beta_1 \in \Delta$ or $\beta_2 \in \Delta$.*

The states of a Hintikka structure will be consistent and downward saturated sets of ATEL formulas. To introduce the epistemic concurrent game Hintikka structures, we need to present the notions of *co-action* and *co-strategy* from (Goranko and van Drimmelen 2006). First, a *co-A-move* at state s is a function $\sigma^c_A : D_A(s) \to D(s)$ s.t. $\sigma_A \sqsubseteq \sigma^c_A(\sigma_A)$; $D^c_A(s)$ denotes the set of all co-A-moves at s. Further, $out(s, \sigma^c_A) = \{\tau(s, \sigma^c_A(\sigma_A)) \mid \sigma_A \in D_A(s)\}$ is the *outcome* of σ^c_A at s. Then, a *γ-recall co-A-strategy* is a mapping $F^c_A[\gamma] : \bigcup_{1\le n<1+\gamma} S^n \mapsto \bigcup_{s\in S} D^c_A(s)$ s.t. $F^c_A[\gamma](\kappa) \in D^c_A(lst(\kappa))$ for every $\kappa \in \bigcup_{1\le n<1+\gamma} S^n$. Finally, the *outcome* of $F^c_A[\gamma]$ at s is the set $out(s, F^c_A[\gamma])$ of all s-runs λ s.t. $\lambda(i+1) \in out(\lambda(i), F^c_A[\gamma](\lambda[j,i]))$ for all $i \ge 0$ and $j = max(i - \gamma + 1, 0)$. Co-strategies are used to define the labelling function of Hintikka structures in the cases of ATEL-formulas $\neg\langle\langle A\rangle\rangle X\varphi$ and $\neg\langle\langle A\rangle\rangle G\varphi$. In what follows we assume γ fixed and omit it.

Definition 9 (ECGHS) *Let $\mathcal{A}$ be a finite set of agent indexes $i_0, \ldots, i_n$. An epistemic concurrent game Hintikka structure, or ECGHS, is a tuple $\mathcal{H} = \langle \S, I, \{Act_i\}_{i\in\mathcal{A}}, \{\sim_i\}_{i\in\mathcal{A}\setminus\{Env\}}, \tau, H\rangle$ where*

- *$\S$ is a set of states;*
- *$I \subseteq \S$ is a set of initial states;*
- *for each $i \in \mathcal{A}$, Act_i is a finite set of actions;*
- *$\tau : \S \times Act \to \S$ is the transitions function;*
- *for each $i \in \mathcal{A} \setminus \{Env\}$, $\sim_i$ is a symmetric, reflexive and transitive relation on $\S$;*
- *H is a labelling function from $\S$ to sets of ATEL formulas s.t.*

H1 if $\neg\phi \in H(s)$ then $\phi \notin H(s)$;

H2 every $H(s)$ is downward saturated;

H3 if $\langle\langle A\rangle\rangle X\varphi \in H(s)$ then there is an A-strategy F_A s.t. for all $\lambda \in out(s, F_A)$, $\varphi \in H(\lambda(1))$;

H4 if $\neg\langle\langle A\rangle\rangle X\varphi \in H(s)$ then there is a co-A-strategy F^c_A s.t. for all $\lambda \in out(s, F^c_A)$, $\neg\varphi \in H(\lambda(1))$;

H5 if $\neg\langle\langle A\rangle\rangle G\varphi \in H(s)$ there is a co-A-strategy F^c_A s.t. for all $\lambda \in out(s, F^c_A)$ there is $i \ge 0$ s.t. $\neg\varphi \in H(\lambda(i))$;

H6 if $\langle\langle A\rangle\rangle\varphi U\varphi' \in H(s)$ there is an A-strategy F_A s.t. for all $\lambda \in out(s, F_A)$ there is $k \ge 0$ s.t. $\varphi' \in H(\lambda(k))$, and for all j, $0 \le j < k$ implies $\varphi \in H(\lambda(j))$;

H7 if $s \sim_i s'$ then $K_i\varphi \in H(s)$ iff $K_i\varphi \in H(s')$;

H8 if $\neg K_i\varphi \in H(s)$ then there is $s' \in \S$ s.t. $s \sim_i s'$ and $\neg\varphi \in H(s')$.

Notice that conditions H3-8 mimic the satisfaction clauses for ATEL formulas in Def. 5. Also, ECGHS are syntactic structures, where each state is labelled by a finite set of ATEL formulas. In addition, the main difference w.r.t. ECGM consists in that global states are primitive. This implies that the indistinguishability relations $\sim_i$ are also primitive. Further, as for ECGM, we can define a notion of run, as well as the set $\mathcal{R}$ of all runs. Hence, we can introduce the following classes of ECGHS.

Definition 10 *An ECGHS $\mathcal{H}$*

- *is* synchronous *iff for every $\lambda, \lambda' \in \mathcal{R}$, $i \in \mathcal{A}$, if $\lambda(n) \sim_i \lambda'(n')$ then $n = n'$;*
- *has a* unique initial state *iff $|I| = 1$.*

Further, similarly to the case for ECGM, we introduce two notions of satisfaction.

Definition 11 (Satisfaction) *An ECGHS $\mathcal{H}$ $\S$-satisfies (resp. I-satisfies) an ATEL-formula θ iff $\theta \in H(s)$ for some $s \in \S$ (resp. $s \in I$).*

The tableau procedure that we present in the following sections ultimately generates a Hintikka structure. From this ECGHS we are always able to extract an ECGM, as stated in the following result.

Theorem 3 *Given a finite set $\mathcal{A}$ of agent indexes and an ECGHS $\mathcal{H}$ on $\mathcal{A}$ that $\S$-satisfies (resp. I-satisfies) an ATEL-formula θ, we can derive a finite set $\mathcal{A}$ of agents and an ECGM $\mathcal{P}$ on $\mathcal{A}$ that also S-satisfies (resp. I-satisfies) θ. Further, this can be done so as to preserve synchronicity and uniqueness of initial state.*

As a consequence of Theorem 3, satisfiability w.r.t. Hintikka structures suffices to obtain satisfiability w.r.t. ECGM. Also, we underline that Theorem 3 is constructive as it provides a set $\mathcal{A}$ of agents generating the ECGM $\mathcal{P}$. In this respect our approach is agent-based. In the following sections we present the tableau-based procedures for deciding satisfiability of ATEL formulas.

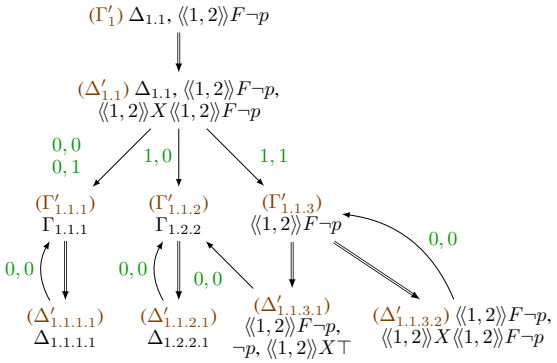


Figure 2: fragment of the pre-tableau for $\Gamma'_1 = \{\theta_1, \langle\langle 1,2\rangle\rangle F(\neg p \wedge \neg K_2 p)\}$.

4 Tableaux for S-Satisfiability

In this section we introduce a tableau-based decision procedure for S-satisfiability of ATEL formulas. Specifically, we build on the methods developed for (non-epistemic) ATL and multi-agent epistemic CMAEL(CD) in (Goranko and Shkatov 2009a; Ajspur, Goranko, and Shkatov 2013). As stated in Lemma 1, for S-satisfiability the assumptions of a unique initial state or synchronicity do not affect satisfiability of ATEL formulas. Nonetheless, we deem the tableaux for S-satisfiability interesting in themselves, as well as useful for a comparison with the case of I-satisfiability. Indeed in this section we introduce notions and present results that will be used also in Section 5.

Similarly to (Goranko and Shkatov 2009a; Ajspur, Goranko, and Shkatov 2013), the tableau procedure for ATEL consists of two phases: a *construction phase*, where a *pre-tableau* is populated with states and prestates (i.e., sets of formulas), and an *elimination phase*, where the pre-tableau is pruned into a proper *tableau* by removing pre-states together with "unsatisfiable" states. The aim of the whole construction is to obtain a tableau from which it is possible to extract a Hintikka structure iff the relevant formula is satisfiable. We start with the construction phase.

4.1 Construction Phase

The construction of the pre-tableau pre-$\mathcal{T}^\theta$ for an ATEL-formula θ proceeds in stages. At each stage rules are applied to create either states from pre-states or pre-states from states. The process starts with the pre-state $\Gamma = \{\theta\}$ containing the ATEL formula to be checked. Then at each stage we apply one of the rules $\mathbf{SR}_S$, $\mathbf{K}_S$ and $\mathbf{Next}_S$ that adapt the corresponding rules in (Goranko and Shkatov 2009a; 2009c). In what follows we say that a set Δ of formulas is a *minimal downward saturated* extension of a set Γ if *(i)* Δ is downward saturated, *(ii)* $\Gamma \subseteq \Delta$, and *(iii)* there is no downward saturated Δ' s.t. $\Gamma \subseteq \Delta' \subset \Delta$. The first rule we present allows us to build states from pre-states.

Definition 12 (Saturation Rule $\mathbf{SR}_S$) *Given a pre-state Γ s.t. $\boldsymbol{SR}_S$ has not been applied to Γ, for every consistent and minimal downward saturated extension Δ of Γ,*

- *if the pre-tableau contains a state Δ' that coincides with Δ, set $\Gamma \Rightarrow \Delta'$;*
- *else,*
  - *add Δ as a new state to the pre-tableau;*
  - *if Δ contains no positive nor negative X-formula, then add $\langle\langle Ag_\theta\rangle\rangle X\top$ to Δ;*
  - *set $\Gamma \Rightarrow \Delta$.*

The (finite) set of states $\{\Delta \mid \Gamma \Rightarrow \Delta\}$ is denoted by **states**(Γ). To introduce the next rule we need some more notation. Hereafter, $\Delta_{|K_i}$ denotes the set $\{K_i\psi \mid K_i\psi \in \Delta\}$. Also, we do not distinguish between a finite set of formulas and their conjunction, the context will disambiguate. The next rule is designed to ensure the satisfaction of epistemic possibilities of the form $\neg K_i\phi$.

Definition 13 (Rule $\mathbf{K}_S$) *Given a state Δ s.t. $\boldsymbol{K}_S$ has not been applied to Δ, for every $\neg K_i\phi \in \Delta$ and consistent $\Gamma = \{\neg\phi\} \cup \Delta_{|K_i}$,*

- *if the pre-tableau contains a pre-state $\Gamma' = \Gamma$, set $\Delta \xrightarrow{\neg K_i\phi} \Gamma'$;*
- *else, add Γ as a new pre-state and set $\Delta \xrightarrow{\neg K_i\phi} \Gamma$.*

Finally, we consider the rule $\mathbf{Next}_S$, which builds the next level of pre-states.

Definition 14 (Rule $\mathbf{Next}_S$) *Given a state Δ s.t. $\boldsymbol{Next}_S$ has not been applied to Δ,*

1. *order linearly all positive X-formulas of Δ followed by all negative ones:*

$$L = \langle\langle A_0\rangle\rangle X\phi_0, \ldots, \langle\langle A_{m-1}\rangle\rangle X\phi_{m-1}, \neg\langle\langle B_0\rangle\rangle X\psi_0, \ldots, \neg\langle\langle B_{l-1}\rangle\rangle X\psi_{l-1}$$

(because of $\boldsymbol{SR}_S$, L is always non-empty.) Let $r_\Delta = m+l$ and $D(\Delta) = \{0, \ldots, r_\Delta - 1\}^{|Ag^+_\theta|}$. Further, for every $\sigma \in D(\Delta)$, we set $N(\sigma) = \{i \mid \sigma_i \geq m\}$ and $neg(\sigma) = \sum_{i \in N(\sigma)}(\sigma_i - m) \mod l$.

2. *For each $\sigma \in D(\Delta)$ consider every consistent*

$$\Gamma_\sigma = \{\neg\psi_q \mid \neg\langle\langle B_q\rangle\rangle X\psi_q \in \Delta, neg(\sigma) = q, Ag^+_\theta \setminus B_q \subseteq N(\sigma)\} \cup \{\phi_p \mid \langle\langle A_p\rangle\rangle X\phi_p \in \Delta \text{ and } \sigma_i = p \text{ for all } i \in A_p\} \quad (1)$$

where $\Gamma_\sigma = \{\top\}$ if the set on the RHS of (1) is empty.

- *If the pre-tableau contains a pre-state $\Gamma' = \Gamma_\sigma$, set $\Delta \xrightarrow{\sigma} \Gamma'$;*
- *else, add Γ_σ as a new pre-state and set $\Delta \xrightarrow{\sigma} \Gamma_\sigma$.*

We remark that by definition of $\mathbf{SR}_S$, $\mathbf{K}_S$ and $\mathbf{Next}_S$, no state nor pre-state in pre-$\mathcal{T}^\theta$ is inconsistent, i.e., it does not contain both ϕ and $\neg\phi$. Also, we denote the (finite) set of pre-states $\{\Gamma \mid \Delta \xrightarrow{\sigma} \Gamma \text{ for some } \sigma\}$ as **pre-states**(Δ). We now briefly describe the construction of the pre-tableau for an ATEL-formula θ. We begin at stage $k = 0$ by creating a pre-state $\Gamma = \{\theta\}$ for the input formula θ. Then, at each stage $k > 0$, we alternatingly apply the rule $\mathbf{SR}_S$ to obtain the states in **states**(Γ), followed by $\mathbf{Next}_S$ and $\mathbf{K}_S$ to create the next level of pre-states and the epistemically related pre-states. The construction phase terminates, after a finite number of stages, when $\mathbf{K}_S$ and $\mathbf{Next}_S$ do not produce any new pre-state, nor $\mathbf{SR}_S$ new states. We conclude this section with an example.

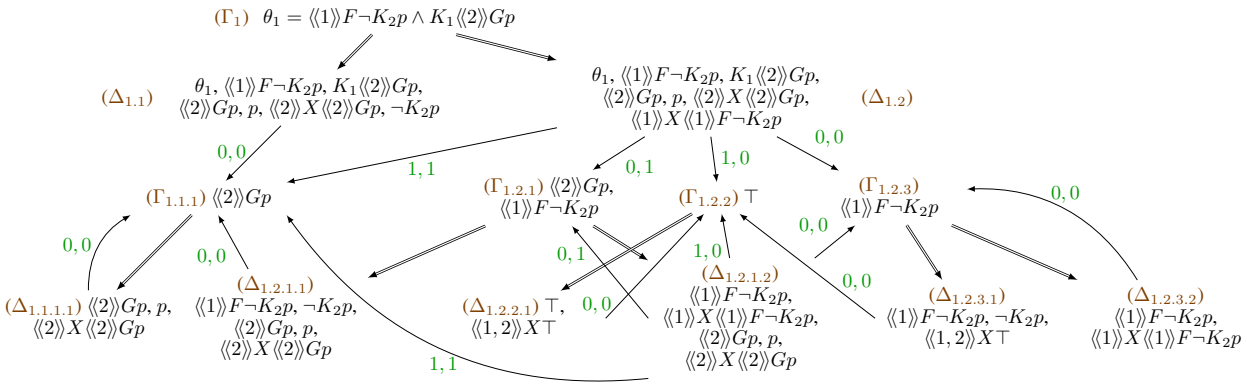


Figure 3: fragment of the pre-tableau for $\theta_1 = \langle\langle 1\rangle\rangle F\neg K_2 p \wedge K_1\langle\langle 2\rangle\rangle Gp$ obtained by applying $\mathbf{SR}_S$ and $\mathbf{Next}_S$ only.

Example 1 Fig. 3 shows a fragment of the pre-tableau for the ATEL-formula $\theta_1 = \langle\langle 1\rangle\rangle F\neg K_2 p \wedge K_1\langle\langle 2\rangle\rangle Gp$ obtained by applying rules $\mathbf{SR}_S$ and $\mathbf{Next}_S$ only. We present this restricted structure as it will be used later on. Also, the environment's actions are omitted. Indeed, since no negative X-formula appears in the pre-tableau, it suffices to consider each action σ_1, σ_2 in Fig. 3 as a shorthand for the set $\{\sigma_{Env}, \sigma_1, \sigma_2 \mid 0 \leq \sigma_{Env} < r_\Delta\}$. ■

4.2 Elimination phase

After completion of the pre-tableau pre-$\mathcal{T}^\theta$ we step into the elimination phase, where we prune pre-$\mathcal{T}^\theta$ first by removing all pre-states, and then by deleting "unsatisfiable" states. We have already noticed that by construction no state in the pre-tableau is inconsistent. Moreover, if a state either *(i)* has no successor for an enabled joint action, or *(ii)* it does not fulfil an epistemic possibility, or *(iii)* it does not fulfil a temporal eventuality, then it is removed. We make this intuition precise in the following elimination rules, adapted from (Goranko and Shkatov 2009a).

Definition 15 (Rule PR) *For every state Δ and pre-state Γ in pre-$\mathcal{T}^\theta$, and $\Delta' \in$ **states**(Γ),*

- *if $\Delta \xrightarrow{C} \Gamma$ then $\Delta \xrightarrow{C} \Delta'$, where C can be either an action σ or an epistemic possibility $\neg K_i\phi$;*
- *remove Γ from pre-$\mathcal{T}^\theta$.*

We now describe the process of state elimination also in stages. We start with the tableau $\mathcal{T}^\theta_0$, obtained by applying **PR** to pre-$\mathcal{T}^\theta$. At each stage n we apply one of the rules **E1**-**3** below to obtain the tableau $\mathcal{T}^\theta_{n+1}$.

E1 If for some $\sigma \in D(\Delta)$ there is no $\Delta' \in \mathcal{T}^\theta_n$ s.t. $\Delta \xrightarrow{\sigma} \Delta'$, then $\mathcal{T}^\theta_{n+1} = \mathcal{T}^\theta_n \setminus \{\Delta\}$.

E2$_S$ If for some $\neg K_i\phi \in \Delta$ there is no $\Delta' \in \mathcal{T}^\theta_n$ s.t. $\Delta \xrightarrow{\neg K_i\phi} \Delta'$, then $\mathcal{T}^\theta_{n+1} = \mathcal{T}^\theta_n \setminus \{\Delta\}$.

Rule **E1** removes states with no successor for an enabled action; while **E2**$_S$ deletes states with unrealized epistemic possibilities. Rule **E3** to be introduced deals with the realisation of temporal eventualities, namely the satisfaction of formulas of the form $\langle\langle A\rangle\rangle\phi U\psi$ and $\neg\langle\langle A\rangle\rangle G\phi$. To present **E3** we consider the following notation. Let Δ be a state in $\mathcal{T}^\theta_0$ and $\langle\langle A\rangle\rangle X\phi$ (resp. $\neg\langle\langle B\rangle\rangle X\psi$) the p-th (resp. q-th) formula in the linear ordering L of X-formulas of Δ in the application of $\mathbf{Next}_S$. Then,

$$D(\Delta, \langle\langle A\rangle\rangle X\phi) = \{\sigma \in D(\Delta) \mid \sigma_i = p \text{ for every } i \in A\}$$
$$D(\Delta, \neg\langle\langle B\rangle\rangle X\psi) = \{\sigma \in D(\Delta) \mid neg(\sigma) = q \text{ and } Ag^+_\theta \setminus B \subseteq N(\sigma)\}$$

We can now define when a temporal eventuality is realized.

Definition 16 (Realization of temporal eventualities) *A temporal eventuality $\beta = \langle\langle A\rangle\rangle\phi U\psi$ (resp. $\neg\langle\langle A\rangle\rangle G\phi$) in $\Delta \in \mathcal{T}^\theta_n$ is* realized *at Δ iff*

1. *$\beta_1 \in \Delta$, or*
2. *$\beta_2 \in \Delta$ and for every $\sigma \in D(\Delta, \langle\langle A\rangle\rangle X\langle\langle A\rangle\rangle\phi U\psi)$ (resp. $D(\Delta, \neg\langle\langle A\rangle\rangle X\langle\langle A\rangle\rangle G\phi)$) there exists $\Delta' \in \mathcal{T}^\theta_n$ s.t.* (i) *$\Delta \xrightarrow{\sigma} \Delta'$; and* (ii) *$\beta$ is realized at $\Delta' \in \mathcal{T}^\theta_n$.*

We can now state the final elimination rule.

E3 If $\Delta \in \mathcal{T}^\theta_n$ contains an unrealized temporal eventuality, then $\mathcal{T}^\theta_{n+1} = \mathcal{T}^\theta_n \setminus \{\Delta\}$.

The elimination phase terminates when none of the rules **E1**-**3** is any longer applicable (obviously, this happens after a finite number of stages.) The tableau thus obtained is denoted as $\mathcal{T}^\theta$. In particular, we say that $\mathcal{T}_\theta$ is *open* if $\theta \in \Delta$ for some $\Delta \in \mathcal{T}_\theta$; otherwise, it is *closed*. In the next sections we show the soundness and completeness of the tableau procedure, that is, the tableau $\mathcal{T}^\theta$ is open iff θ is S-satisfiable.

4.3 Soundness and Completeness

We first outline the soundness proof for the tableau procedure presented above, which amounts to the following result:

Theorem 4 (Soundness) *If θ is S-satisfiable, then $\mathcal{T}^\theta$ is open.*

The proof of Theorem 4 consists of two parts. First, we use the following lemma to show that if θ is S-satisfiable, then at least one of the states in **states**$(\{\theta\})$ is S-satisfiable.

Lemma 5 *Let Γ be an S-satisfiable pre-state in pre-$\mathcal{T}^\theta$ Then, at least one $\Delta \in$ **states**(Γ) is S-satisfiable.*

Then, we show that no satisfiable state is removed during the elimination phase. The soundness result then follows, as θ belongs to $\Delta \in \mathcal{T}^\theta$, for some $\Delta \in$ **states**$(\{\theta\})$. To show that no S-satisfiable state is removed, we prove by induction on the number n of stages that, for every $\Delta \in \mathcal{T}_0^\theta$, if Δ is S-satisfiable, then Δ is not removed at stage n. The base case for $n = 0$ is trivial as no elimination rule has been applied. As regards the inductive step, we consider the following lemma:

Lemma 6 *If $\Delta \in \mathcal{T}_n^\theta$ is S-satisfiable, then*

1. *for every $\sigma \in D(\Delta)$ there is $\Delta' \in \mathcal{T}_n^\theta$ s.t. $\Delta \xrightarrow{\sigma} \Delta'$;*
2. *for every $\neg K_i\phi \in \Delta$ there is $\Delta' \in \mathcal{T}_n^\theta$ s.t. $\Delta \xrightarrow{\neg K_i\phi} \Delta'$;*
3. *every temporal eventuality is realized at Δ.*

We conclude that if Δ is S-satisfiable, then it will not be eliminated by **E1**-**3**. Hence, some $\Delta \supseteq \{\theta\}$ belongs to $\mathcal{T}^\theta$, i.e., $\mathcal{T}^\theta$ is open.

The completeness proof consists in showing that if the tableau $\mathcal{T}^\theta$ is open, then the ATEL-formula θ is S-satisfiable.

Theorem 7 (Completeness) *If $\mathcal{T}^\theta$ is open, then θ is S-satisfiable.*

The proof amounts to building an ECGHS $\mathcal{H}_\theta$ starting from the open tableau $\mathcal{T}^\theta$. By Theorem 3 the desired result immediately follows. Specifically, when building the Hintikka structure for θ we take care of satisfying conditions H5, H6 and H8 for temporal eventualities and epistemic possibilities. In this task the key concept is that of a *temporal component* $F(\xi, \Delta)$ for an eventuality ξ and state Δ (Goranko and Shkatov 2009a; 2009c). If $\xi \in \Delta$ then $F(\xi, \Delta)$ is a tree rooted at Δ that realizes ξ. If $\xi \notin \Delta$ then $F(\xi, \Delta)$ is a tree with all the successor of Δ. We now outline the construction of the ECGHS $\mathcal{H}_\theta$ satisfying an ATEL-formula θ.

The construction proceeds in stages. First, we list all states in $\mathcal{T}^\theta$ as $\Delta_0, \ldots, \Delta_{n-1}$ and all temporal eventualities as $\xi_0, \ldots, \xi_{m-1}$. Further, we assume that the temporal components are arranged in an $m \times n$ matrix, whose rows (resp. columns) are marked with the corresponding temporal eventualities (resp. states). Hence, $F_{p,q} = F(\xi_p, \Delta_q)$. Now, if θ is the temporal eventuality ξ_p, for $p < m$, we set $\mathfrak{S}_0^0 = F_{p,q}$ for some q s.t. $\theta \in \Delta_q$. Otherwise, we set $\mathfrak{S}_0^0 = F_{0,q}$, where q is as above. For the induction step, suppose that we have defined $\mathfrak{S}_k^i$. Then, we construct $\mathfrak{S}_k^{i+1}$ by realizing all temporal eventualities $\xi \in \mathfrak{S}_k^i$. That is, we keep track of the last realized eventuality, say ξ_i, and replace every deadlocked state w such that $l(w) = \Delta_j$ with the temporal component $F_{(i+1) \mod m,j}$. We repeat the procedure until all temporal eventualities have been realized. Further, given $\mathfrak{S}_k^i$ we construct $\mathfrak{S}_{k+1}^i$ by adding for every $\Delta \in \mathfrak{S}_k^i$ and every epistemic possibility $\neg K_i\phi \in \Delta$ a state $\Delta' \in \mathcal{T}^\theta$ s.t. $\neg\phi \in \Delta'$. Notice that such Δ' always exists as $\Delta \in \mathcal{T}^\theta$. We proceed in this way to realize all epistemic possibilities contained in $\mathfrak{S}_k^i$. To guarantee that the ECGHS we are building remains finite, whenever the component we are about to add, say $F_{p,q}$, is already contained in $\mathfrak{S}_k^i$, instead of replacing the state w (s.t. $l(w) = \Delta_q$) with $F_{p,q}$, we connect every "predecessor" v of w to the root of $F_{p,q}$. The procedure is repeated until no more components are added, thus obtaining an ECGHS $\mathcal{H}_\theta$.

Lemma 8 *If $\mathcal{T}^\theta$ is open, then there exists an ECGHS $\mathcal{H}_\theta$ that $\S$-satisfies θ.*

Complexity. We conclude this section with a discussion of the complexity of the decisione procedure outlined above. It can be shown that the construction and elimination phases can be performed in time exponential in the size of the formula. Hence, the procedure here provided meets the lower-bound of the satisfiability problem for ATEL (Walther 2005). In this respect, it is optimal. Further, as discussed in (Goranko and Shkatov 2009a), since the tableau construction is incremental, in many cases it allows to decide satisfaction in less time than top-down tableaux, which need to build all formula types.

5 Tableaux for I-Satisfiability

In this section we introduce the tableau-based decision procedure for I-satisfiability. Specifically, in Lemma 1 we remarked that the notions of I- and S-satisfiability coincide over the class of all (resp., synchronous) ECGM. On the other hand, for ECGM with a unique initial state we need to consider a different construction for I-satisfiability.

5.1 Construction Phase

We start with the rules to build the pre-tableau pre-$\mathcal{T}^\theta$, but first we need to introduce some new terminology. In what follows we index each state and pre-state with a finite set $\{\vec{k}\}$ of time-stamps (we assume the set is enumerated). Intuitively, these represent the times at which a state (resp. pre-state) occurs in a fragment of a run. Operations on a time-stamp $\{\vec{k}\}$ (such as addition) are defined on each member of $\{\vec{k}\}$. Further, in the construction phase we define a sequence $\Gamma_0^{\{0\}}, \ldots, \Gamma_n^{\{0\}}$ of "initial" pre-states, starting from $\Gamma_0^{\{0\}} = \{\theta\}$. Also, we introduce a mark mk that intuitively keeps track of the initial state a state (resp. pre-state) stems from. We set $mk(\Gamma_0^{\{0\}}) = undef$ at the beginning. Finally, let $\mapsto$ be the transitive closure of $\Rightarrow \cup \rightarrow$. We now provide the saturation rule $\mathbf{SR}_I^{uis,sync}$ for I-satisfiability w.r.t. synchronous ECGM with a unique initial state.

Definition 17 (Saturation Rule $\mathbf{SR}_I^{uis,sync}$) *Given a pre-state $\Gamma^{\{\vec{k}\}}$ s.t. $\boldsymbol{SR}_I^{uis,sync}$ has not been applied to $\Gamma^{\{\vec{k}\}}$, consider every consistent and minimal downward saturated extensions $\Delta^{\{\vec{k}\}}$ of $\Gamma^{\{\vec{k}\}}$,*

- *if the pre-tableau contains a state $\Delta'^{\{\vec{m}\}} = \Delta^{\{\vec{k}\}}$ and $mk(\Delta'^{\{m\}}) = mk(\Gamma^{\{\vec{k}\}})$, then*
  - *rename $\Delta'^{\{\vec{m}\}}$ and all pre-states and states $\Theta_i^{\{\vec{m}+i\}}$ s.t. $\Delta'^{\{\vec{m}\}} \mapsto \Theta_i^{\{\vec{m}+i\}}$ as $\Delta'^{\{\vec{m}\}\cup\{\vec{k}\}}$ and $\Theta_i^{\{\vec{m}+i\}\cup\{\vec{k}+i\}}$. Then, set $\Gamma^{\{\vec{k}\}} \Rightarrow \Delta'^{\{\vec{m}\}\cup\{\vec{k}\}}$.*
- *else,*
  - *add $\Delta^{\{\vec{k}\}}$ as a new state to the pre-tableau;*

– *if* $\Delta^{\{\vec{k}\}}$ *contains no positive nor negative X-formulas, then add* $\langle\langle Ag_\theta\rangle\rangle X\top$ *to* $\Delta^{\{\vec{k}\}}$*;*

– *set* $\Gamma^{\{\vec{k}\}} \Rightarrow \Delta^{\{\vec{k}\}}$ *and* $mk(\Delta^{\{\vec{k}\}}) = mk(\Gamma^{\{\vec{k}\}})$ *unless* $\{\vec{k}\} = \{0\}$ *and* $mk(\Gamma^{\{0\}}) = undef$*. In this case, set* $mk(\Delta^{\{0\}}) = \Delta^{\{0\}}$*.*

Hence, $\mathbf{SR}_I^{uis,sync}$ modifies $\mathbf{SR}_S$ by keeping track of time-stamps as well as of mark mk: the former are needed for synchronicity, while the latter registers the initial state. In particular, for all $\Delta, \Delta' \in$ **states**(Γ), $mk(\Delta) = mk(\Delta')$, and for every state Δ', $\Delta^{\{0\}} \mapsto \Delta'$ iff $mk(\Delta') = \Delta^{[0]}$. The rule $\mathbf{SR}_I^{uis}$ for I-satisfiability w.r.t. $ECGM^{uis}$ can be obtained from $\mathbf{SR}_I^{uis,sync}$ by omitting time-stamps. Further, we need to modify the rule $\mathbf{Next}_S$ as follows.

Definition 18 (Rule $\mathbf{Next}_I^{uis,sync}$) *Given a state* $\Delta^{\{\vec{k}\}}$ *s.t.* $\boldsymbol{Next}_I^{uis,sync}$ *has not been applied to* $\Delta^{\{\vec{k}\}}$*,*

1. *(as in rule* $\boldsymbol{Next}_S$*)*
2. *For each* $\sigma \in D(\Delta^{\{\vec{k}\}})$ *consider every consistent*

$$\Gamma_\sigma^{\{\vec{k}+1\}} = \{\phi_p \mid \langle\langle A_p\rangle\rangle X\phi_p \in \Delta^{\{\vec{k}\}}, \sigma_i = p \text{ for all } i \in A_p\} \cup \{\neg\psi_q \mid \neg\langle\langle B_q\rangle\rangle X\psi_q \in \Delta^{\{\vec{k}\}}, neg(\sigma) = q, Ag_\theta^+ \setminus B_q \subseteq N(\sigma)\} \quad (2)$$

where $\Gamma_\sigma^{\{\vec{k}+1\}} = \{\top\}$ *if the set on the RHS of (2) is empty.*

- *If the pre-tableau contains a pre-state* $\Gamma'^{\{\vec{m}\}} = \Gamma_\sigma^{\{\vec{k}+1\}}$ *s.t.* $mk(\Gamma'^{\{\vec{m}\}}) = mk(\Gamma_\sigma^{\{\vec{k}+1\}})$*, then*
  – *rename* $\Gamma'^{\{\vec{m}\}}$ *and all pre-states and states* $\Theta_i^{\{\vec{m}+i\}}$ *s.t.* $\Gamma'^{\{\vec{m}\}} \mapsto \Theta_i^{\{\vec{m}+i\}}$ *as* $\Gamma'^{\{\vec{m}\}\cup\{\vec{k}+1\}}$ *and* $\Theta_i^{\{\vec{m}+i\}\cup\{\vec{k}+i+1\}}$*. Then, set* $\Delta^{\{\vec{k}\}} \xrightarrow{\sigma} \Gamma'^{\{\vec{m}\}\cup\{\vec{k}+1\}}$*;*
- *else,*
  – *add* $\Gamma_\sigma^{\{\vec{k}+1\}}$ *as a new pre-state to the pre-tableau;*
  – *set* $\Delta^{\{\vec{k}\}} \xrightarrow{\sigma} \Gamma_\sigma^{\{\vec{k}+1\}}$ *and* $mk(\Gamma_\sigma^{\{\vec{k}+1\}}) = mk(\Delta^{\{\vec{k}\}})$*.*

Also in this case, the rule $\mathbf{Next}_I^{uis}$ is obtained by omitting indexes from states and pre-states in $\mathbf{Next}_I^{uis,sync}$. Now we consider the epistemic operator. Similarly to the above, we have to introduce a distinct case for each of $ECGM^{uis}$ and $ECGM^{uis,sync}$. Moreover, for each case we have to introduce two rules that apply at different stages of the construction phase.

Definition 19 (Rule $\mathbf{K}_1^{uis}$) *Given a state* Δ *s.t.* $\boldsymbol{K}_1^{uis}$ *has not been applied to* Δ*, for every* $\neg K_i\phi \in \Delta$*,*

- *if* $\langle\langle Ag_\theta\rangle\rangle F(\neg\phi \wedge \Delta|_{K_i}) \notin mk(\Delta)$*, then add* $\Gamma_{i+1} = mk(\Delta) \cup \{\langle\langle Ag_\theta\rangle\rangle F(\neg\phi \wedge \Delta|_{K_i})\}$ *if it is consistent, where* i *is the greatest index s.t.* Γ_i *appears in the pre-tableau;*
- *set* $mk(\Gamma_{i+1}) = undef$*.*

Intuitively, when Δ contains an epistemic possibility $\neg K_i\phi$ and the root of Δ does not anticipate to fulfil this possibility (i.e., $\langle\langle Ag_\theta\rangle\rangle F(\neg\phi \wedge \Delta|_{K_i}) \notin mk(\Delta)$), then by $\mathbf{K}_1^{uis}$ we attempt to construct a branch of the pre-tableau that satisfies $\neg K_i\phi$ by adding a pre-state Γ_{i+1} as above. To define the corresponding rule for $ECGM^{uis,sync}$ we introduce the notion of a *loop* on a state Δ, that is, a sequence $\Delta_0, \Gamma_1, \Delta_1 \ldots, \Gamma_n, \Delta_n$ s.t. *(i)* $\Delta = \Delta_0 = \Delta_n$; *(ii)* $\Gamma_{i+1} \in$ **pre-states**(Δ_i); *(iii)* $\Delta_i \in$ **states**(Γ_i); and *(iv)* $\Delta_i \neq \Delta$ for $0 < i < n$. Let n be the *length* of the loop. We denote the set of lengths of all loops for a state Δ as $loop(\Delta)$. Also, for $\neg K_i\phi \in \Delta^{\{\vec{k}\}}$, we use ζ as a shorthand for $\neg\phi \wedge \Delta^{\{\vec{k}\}}|_{K_i}$.

Definition 20 (Rule $\mathbf{K}_1^{uis,sync}$) *Given a state* $\Delta^{\{\vec{k}\}}$ *s.t.* $\boldsymbol{K}_1^{uis,sync}$ *has not been applied to* $\Delta^{[\vec{k}]}$*, for every* $\neg K_i\phi \in \Delta^{\{\vec{k}\}}$*,* $Y \subseteq \{\vec{k}\}$*, and* $W \subseteq loop(\Delta^{\{\vec{k}\}})$*,*

- *if* $\eta = \bigwedge_{y\in Y}(\langle\langle Ag_\theta\rangle\rangle X)^y(\zeta \wedge \langle\langle\emptyset\rangle\rangle G(\zeta \rightarrow \bigwedge_{w\in W}(\langle\langle Ag_\theta\rangle\rangle X)^w\zeta)) \notin \Delta^{\{\vec{k}\}}$*, then add* $\Gamma_{i+1}^{\{0\}} = mk(\Delta^{\{\vec{k}\}}) \cup \{\eta\}$ *if it is consistent, where* i *is the smallest index s.t.* $\Gamma_i^{\{0\}}$ *appears in the pre-tableau;*
- *set* $mk(\Gamma_{i+1}^{\{0\}}) = undef$*.*

The *rationale* for $\mathbf{K}_1^{uis,sync}$ comes from the fact that satisfiability w.r.t. $ECGM^{sync,uis}$ needs stronger conditions. Indeed, we have to ensure that for every $\neg K_i\phi$ in $\Delta^{\{\vec{k}\}}$ there exists a $\Delta'^{\{\vec{k}\}}$ s.t. $\neg\phi \in \Delta^{\{\vec{k}\}}$. In particular, $\Delta^{\{\vec{k}\}}$ and $\Delta'^{\{\vec{k}\}}$ have to share the same time-stamp $\{\vec{k}\}$.

The construction phase for satisfiability w.r.t. $ECGM^{uis}$ (resp. $ECGM^{uis,sync}$) goes then as follows. As customary, we start with the initial pre-state $\Gamma_0^{\{0\}} = \{\theta\}$, which contains only the input formula. Then, we alternatingly apply $\mathbf{SR}_I$ and $\mathbf{Next}_I$ (we omit superscripts to refer to both cases) until these rules are no longer applicable. This procedure builds a purely temporal pre-tableau, i.e., a pre-tableau whose $\rightarrow$-edges are marked by joint actions only. At this point, we apply $\mathbf{K}_1^{uis}$ (resp. $\mathbf{K}_1^{uis,sync}$). If a new initial pre-states $\Gamma_i^{\{0\}}$ is created, then we repeat the procedure above. We proceed in this way until none of the rules is any longer applicable. Finally, we apply the rules $\mathbf{K}_2^{uis}$ (resp. $\mathbf{K}_2^{uis,sync}$).

Definition 21 (Rule $\mathbf{K}_2^{uis,sync}$) *Given a state* $\Delta^{\{\vec{k}\}}$ *s.t.* $\boldsymbol{K}_2^{uis,sync}$ *has not been applied to* $\Delta^{[\vec{k}]}$*, for every* $\neg K_i\phi \in \Delta^{\{\vec{k}\}}$ *and state* $\Delta'^{[\vec{k}]}$*,*

- *if* $mk(\Delta'^{\{\vec{k}\}}) = mk(\Delta^{\{\vec{k}\}})$ *and* $\{\neg\phi\} \cup \Delta^{\{\vec{k}\}}|_{K_i} \subseteq \Delta'^{\{\vec{k}\}}$*, then set* $\Delta^{\{\vec{k}\}} \xrightarrow{\neg K_i\phi} \Delta'^{\{\vec{k}\}}$*.*

The rule $\mathbf{K}_2^{uis}$ is obtained from $\mathbf{K}_2^{uis,sync}$ by omitting time-stamps, so we can relate states with different time-stamps. Notice that neither $\mathbf{K}_2^{uis}$ nor $\mathbf{K}_2^{uis,sync}$ creates new pre-states. Instead, they link a state containing an epistemic possiblity with a state that fulfils that possibility. Also in the present case we can prove that the construction phase terminates after a finite number of steps. We conclude this section with an example.

Example 2 In Fig. 3 we reported a fragment of the pre-tableau for the ATEL-formula $\theta_1 = \langle\langle 1\rangle\rangle F\neg K_2 p \wedge K_1\langle\langle 2\rangle\rangle Gp$. We now observe that it also corresponds to the fragment of the pre-tableau obtainable by applying rules $\mathbf{SR}_I^{uis}$ and $\mathbf{Next}_I^{uis}$ only, modulo the mark mk. Since state

$\Delta_{1.1}$ in Fig. 3 contains an epistemic possibility, we apply rule $\mathbf{K}_1^{uis}$ and create a new initial pre-state $\Gamma'_1 = \{\theta_1, \langle\langle 1,2\rangle\rangle F(\neg p \wedge \neg K_2 p)\}$. The fragment of the pre-tableau for Γ'_1 is shown in Fig. 2. ■

5.2 Elimination Phase

For I-satisfiability the elimination phase makes use of most of the rules for S-satisfiability, with some exceptions. First, by **PR** we remove all pre-states and re-direct incoming and outcoming $\rightarrow$-edges. Further, we proceed in stages and at each stage we apply one of the elimination rules **E1**, **E3**, and the following new rule:

E2$_I^{uis,sync}$ If for some $\neg K_i\phi \in \Delta^{\vec{k}}$ there is no $\Delta'^{\vec{k}} \in \mathcal{T}_n^\theta$ s.t. $mk(\Delta'^{\vec{k}}) = mk(\Delta^{\vec{k}})$, $mk(\Delta'^{\vec{k}}) \Rightarrow^* \Delta'^{\vec{k}}$, and $\Delta^{\vec{k}} \xrightarrow{\neg K_i\phi} \Delta'^{\vec{k}}$, then $\mathcal{T}_{n+1}^\theta = \mathcal{T}_n^\theta \setminus \{\Delta^{\vec{k}}\}$.

This means that epistemic possibilities have to be satisfied by states that have the same time-stamp and are reachable from an initial state. The corresponding rule **E2**$_I^{uis}$ can be obtained by omitting time-stamps. Finally, for synchronous ECGM only, we have also to consider the following rule to manipulate time-stamps:

Ets$_I$ If $\mathcal{T}_{n+1}^\theta = \mathcal{T}_n^\theta \setminus \{\Delta^{\vec{k}}\}$ and in $\mathcal{T}_n^\theta$ it was the case that $\Delta^{\vec{k}} \Rightarrow^* \Delta'^{\vec{m}}$, then in $\mathcal{T}_{n+1}^\theta$ rename $\Delta'^{\vec{m}}$ as $\Delta'^{\vec{m}\setminus\vec{k}}$ (remove $\Delta'^{\vec{m}}$ if $\vec{m} \setminus \vec{k} = \emptyset$.)

The elimination phase terminates when it is no longer possible to apply any of the elimination rules above. As in the case of S-satisfiability, the structure obtained at the end of the elimination phase is the tableau $\mathcal{T}^\theta$, which is said to be open if $\theta \in \Delta$, for some $\Delta \in \mathcal{T}^\theta$ s.t. $\Delta \in$ **states**(Γ_i), for some initial Γ_i. Hereafter we outline the soundess and completeness proof of the tableau procedure for I-satisfiability.

5.3 Soundness and Completeness

The soundness proof for the tableau-based decision procedure amounts to the following result, analogous to Theorem 4.

Theorem 9 *If θ is I-satisfiable, then $\mathcal{T}^\theta$ is open.*

The structure of the proof is similar to Theorem 4, i.e., we show that the elimination rules do not remove satisfiable state. However, we need to consider different elimination rules. Hereafter we briefly consider the case of **E2**$_I^{uis}$. Suppose that $\Delta' \in \mathcal{T}_n^\theta$ is S-satisfiable and reachable from Δ, also $\neg K_i\phi \in \Delta'$, and there exists a sequence $\Theta_0 \rightarrow \ldots \rightarrow \Theta_n$ of states s.t. $\Theta_0 = \Delta$, $\Theta_n = \Delta'$, and each Θ_i is S-satisfiable (the existence of such a sequence follows from Lemma 5.2 in (Goranko and Shkatov 2009a)). In particular, we can deduce that there exists an ECGM $\mathcal{P}$ with a unique initial state, and $s_0 \in \mathcal{P}$ s.t. $(\mathcal{P}, s_0) \models \Theta_0 = \Delta$, $s_i \rightarrow s_{i+1}$ and $(\mathcal{P}, s_i) \models \Theta_i$. Since $\neg K_i\phi \in \Delta' = \Theta_n$, there exists $s' \in \mathcal{P}$ s.t. $s' \sim_i s$ and $(\mathcal{P}, s') \models \neg\phi \wedge \Delta|_{K_i}$. By the uniqueness of the initial state, we have that s' is reachable from s_0. Hence $(\mathcal{P}, s_0) \models \langle\langle Ag_\theta\rangle\rangle F(\neg\phi \wedge \Delta|_{K_i})$. As a result we obtain that $\Gamma_{i+1} = \Delta \cup \{\langle\langle Ag_\theta\rangle\rangle F(\neg\phi \wedge \Delta|_{K_i})\}$ is also I-satisfiable, and by Lemma 5 some Δ_{i+1} saturating Γ_{i+1} is also I-satisfiable. Now, we remark that if Θ is satisfiable and $\Delta \rightarrow^* \Theta$ then $\Delta_{i+1} \rightarrow^* \Theta$. In particular, Δ' is reachable from Δ_{i+1}. Moreover, since $\langle\langle Ag_\theta\rangle\rangle F(\neg\phi \wedge \Delta|_{K_i})\} \in \Delta_{i+1}$ and Δ_{i+1} is I-satisfiable, the eventuality $\langle\langle Ag_\theta\rangle\rangle F(\neg\phi \wedge \Delta|_{K_i})$ is realized in some Δ'' reachable from Δ_{i+1}. As a consequence, when we apply rule $\mathbf{K}_2^{uis}$ we obtain that $\Delta' \xrightarrow{\neg K_i\phi} \Delta''$ and Δ' cannot be eliminated by an application of **E2**$_I^{uis}$. As a final result, we have that the I-satisfiable set Δ containing θ is never removed from $\mathcal{T}^\theta$, thus it remains open.

To prove the completeness of the tableaux for I-satisfiability, we have to prove the following result, corresponding to Theorem 7.

Theorem 10 (Completeness) *If $\mathcal{T}^\theta$ is open, then θ is I-satisfiable.*

Also in this case, the proof amounts to build an ECGHS $\mathcal{H}_\theta$ starting from the open tableau $\mathcal{T}^\theta$. Moreover, $\mathcal{H}_\theta$ has to have an unique initial state, or being synchronous. The key remark here is that we can restrict the construction of temporal components to states having the same mark mk. Thus, we state the following result, from which Theorem 10 immediately follows.

Lemma 11 *If $\mathcal{T}^\theta$ is open, then there exists an ECGHS $\mathcal{H}_\theta$ that I-satisfies θ.*

Complexity. We briefly discuss the complexity of the procedure above and notice that it is non-elementary, as in the worst case we have to reinitiate the tableau construction for each epistemic possibility to be fulfilled. We are not aware of any contribution to the complexity of I-satisfiability for ATEL formulas in either $ECGM^{sync}$ or $ECGM^{uis,sync}$. So, we here provide an upper bound for this problem.

6 Conclusions

In this paper we presented a tableau-based method for deciding the satisfiability of ATEL formulas in a number of settings. We considered satisfiability in some initial state as opposed to any state of the model. We argued that the former problem is of interest for verification purposes, while the latter notion is typical of logical investigations. Further, we considered the assumptions of synchronicity and of a unique initial state. These are standard conditions in the literature on Interpreted Systems and their impact on the satisfiability problem has been assessed in relation with various temporal epistemic logics, based on both linear and branching time (Halpern and Vardi 1986; 1989). This paper aimed at extending this type of investigations to alternating time epistemic logic.

There is a number of possible extensions for the present results. In one direction, other assumptions from Interpreted Systems, such as perfect knowldge and no learning, can be imposed on ECGM and corresponding decision procedures developed. It is well-known that the decision problem under these conditions becomes particularly hard (Halpern and Vardi 1986; 1989). So, incremental tableau techniques can contribute to alleviate the decision task for such cases, at least on average, since we do not have to build all formula types as in top-down tableaux. On another direction, notions such as imperfect information and commitment in strategies

have been considered in relation to ATL. It would be of interest to check their impact also w.r.t. satisfiability of ATEL formulas. In yet another direction, modalities for group knowledge can be added to ATEL. This extension has not been considered here, so as not to make the formal language too cumbersome. We leave all these issues for future research.

References

Agotnes, T.; Goranko, V.; and Jamroga, W. 2007. Alternating-time temporal logics with irrevocable strategies. In *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '07, 15–24. New York, NY, USA: ACM.

Ajspur, M.; Goranko, V.; and Shkatov, D. 2013. Tableau-based decision procedure for the multiagent epistemic logic with all coalitional operators for common and distributed knowledge. *Logic Journal of the IGPL* 21(3):407–437.

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *J. ACM* 49(5):672–713.

Baier, C., and Katoen, J.-P. 2008. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press.

Belardinelli, F. 2014. Satisfiability of alternating-time temporal epistemic logic through tableaux. In *Proceedings of the 2nd International Workshop on Strategic Reasoning*.

Blackburn, P.; de Rijke, M.; and Venema, Y. 2001. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Bulling, N.; Dix, J.; and Jamroga, W. 2010. Model checking logics of strategic ability: Complexity*. In Dastani, M.; Hindriks, K. V.; and Meyer, J.-J. C., eds., *Specification and Verification of Multi-agent Systems*. Springer US. 125–159.

D'Agostino, M.; Gabbay, D. M.; Hhnle, R.; and Posegga, J., eds. 1999. *Handbook of Tableau Methods*. Springer.

Fagin, R.; Halpern, J.; Moses, Y.; and Vardi, M. 1995. *Reasoning About Knowledge*. The MIT Press.

Gammie, P., and van der Meyden, R. 2004. MCK: Model checking the logic of knowledge. In *Proceedings of 16th International Conference on Computer Aided Verification (CAV'04)*, volume 3114 of *LNCS*, 479–483. Springer-Verlag.

Goranko, V., and Jamroga, W. 2004. Comparing semantics of logics for multi-agent systems. *Synthese* 139(2):241–280.

Goranko, V., and Shkatov, D. 2009a. Tableau-based decision procedures for logics of strategic ability in multiagent systems. *ACM Trans. Comput. Log.* 11(1).

Goranko, V., and Shkatov, D. 2009b. Tableau-based decision procedure for full coalitional multiagent temporal-epistemic logic of linear time. In Sierra, C.; Castelfranchi, C.; Decker, K. S.; and Sichman, J. S., eds., *AAMAS (2)*, 969–976. IFAAMAS.

Goranko, V., and Shkatov, D. 2009c. Tableau-based decision procedure for the full coalitional multiagent logic of branching time. In Baldoni, M.; Baroglio, C.; Bentahar, J.; Boella, G.; Cossentino, M.; Dastani, M.; Dunin-Keplicz, B.; Fortino, G.; Gleizes, M. P.; Leite, J.; Mascardi, V.; Padget, J. A.; Pavón, J.; Polleres, A.; Fallah-Seghrouchni, A. E.; Torroni, P.; and Verbrugge, R., eds., *MALLOW*, volume 494 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Goranko, V., and van Drimmelen, G. 2006. Complete axiomatization and decidability of alternating-time temporal logic. *Theor. Comput. Sci.* 353(1-3):93–117.

Halpern, J., and Vardi, M. 1986. The complexity of reasoning about knowledge and time. In *ACM Symposium on Theory of Computing (STOC '86)*, 304–315. Baltimore, USA: ACM Press.

Halpern, J., and Vardi, M. 1989. The complexity of reasoning about knowledge and time 1: lower bounds. *Journal of Computer and System Sciences* 38(1):195–237.

Halpern, J.; van der Meyden, R.; and Vardi, M. 2004. Complete axiomatizations for reasoning about knowledge and time. *SIAM Journal on Computing* 33(3):674–703.

Jamroga, W., and van der Hoek, W. 2004. Agents that know how to play. *Fundam. Inform.* 63(2-3):185–219.

Kacprzak, M.; Nabialek, W.; Niewiadomski, A.; Penczek, W.; Pólrola, A.; Szreter, M.; Wozna, B.; and Zbrzezny, A. 2008. Verics 2007 - a model checker for knowledge and real-time. *Fundamenta Informaticae* 85(1-4):313–328.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2009. MCMAS: A model checker for the verification of multi-agent systems. In Bouajjani, A., and Maler, O., eds., *CAV*, volume 5643 of *Lecture Notes in Computer Science*, 682–688. Springer.

Meyer, J.-J. C., and Hoek, W. 1995. *Epistemic Logic for AI and Computer Science*, volume 41 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

van der Hoek, W., and Wooldridge, M. 2003. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica* 75(1):125–157.

van Harmelen, F.; Lifschitz, V.; and Porter, B. 2007. *Handbook of Knowledge Representation*. San Diego, USA: Elsevier Science.

Walther, D.; Lutz, C.; Wolter, F.; and Wooldridge, M. 2006. Atl satisfiability is indeed exptime-complete. *J. Log. Comput.* 16(6):765–787.

Walther, D. 2005. Atel with common and distributed knowledge is exptime-complete. In *Proceedings of Methods for Modalities 4*.

Wooldridge, M. 2000. Computationally Grounded Theories of Agency. In *Proc. of ICMAS*. IEEE Press. 13–22.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Reasoning about Equilibria in Game-Like Concurrent Systems

Julian Gutierrez and **Paul Harrenstein** and **Michael Wooldridge**
Department of Computer Science
University of Oxford

Abstract

Our aim is to develop techniques for reasoning about game-like concurrent systems, where the components of the system act rationally and strategically in pursuit of logically-specified goals. We first present a computational model for such systems, and investigate its properties. We then define and investigate a branching-time logic for reasoning about the equilibrium properties of such systems. The key operator in this logic is a path quantifier $[\mathbf{NE}]\varphi$, which asserts that φ holds on all Nash equilibrium computations of the system.

1 Introduction

Our goal is to develop a theory and techniques for reasoning about *game-like concurrent systems*: concurrent systems in which system components (agents) act strategically in pursuit of their own interests. Game theory is the mathematical theory of strategic interaction, and as such is an obvious candidate to provide the analytical tools for this purpose (Osborne and Rubinstein 1994). However, since the systems we are interested in modelling and reasoning about are interacting computer programs, it seems appropriate to consider how existing techniques for the analysis of computer systems might be combined with game theoretic concepts. Temporal logics (Emerson 1990) and model checking (Clarke, Grumberg, and Peled 2000) form the most important class of techniques for reasoning about computer programs, and in this paper we are concerned with extending such formalisms and techniques to the strategic analysis of systems.

The AI/computer science literature is of course replete with logics intended for reasoning about game-like systems: Parikh's Game Logic was an early example (Parikh 1985), and more recently ATL (Alur, Henzinger, and Kupferman 2002) and Strategy Logic (Chatterjee, Henzinger, and Piterman 2010) have received much attention. However, these formalisms are primarily intended for reasoning about the strategies/choices of players and their effects, rather than the preferences of players and the strategic choices they will make arising from them. It is, of course, possible to use a temporal logic like ATL or Strategy Logic (or indeed LTL, CTL, ...) to define the goals of agents, and hence their preferences; but such languages don't provide any direct mechanism for reasoning about the behaviour of such agents under the assumption that they act rationally and strategically in pursuit of their goals. In this paper, we present a branching time logic that is explicitly intended for this purpose. Specifically, we provide a logic for reasoning about the *equilibrium* properties of game-like concurrent systems.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Equilibrium concepts are the best-known and most widely applied analytical tools in the game theory literature, and of these Nash equilibrium is the best-known (Osborne and Rubinstein 1994). A Nash equilibrium is an outcome that obtains because no player has a rational incentive to deviate from it. If we consider Nash equilibrium in the context of game-like concurrent systems, then it is natural to ask *which computations (runs, histories, ...) will be generated in equilibrium*? In (Gutierrez, Harrenstein, and Wooldridge 2013), this question was investigated using the Iterated Boolean Games (iBG) model. In this model, each player is assumed to control a set of Boolean variables, and the game is played over an infinite sequence of rounds, where at each round every player chooses values for its variables. Each player has a goal, expressed as an LTL formula, and acts strategically in pursuit of this goal. Given this, some computations of a game can be identified as being the result of Nash equilibrium strategies, and (Gutierrez, Harrenstein, and Wooldridge 2013) suggested that the key questions in the strategic analysis of the system are whether a given LTL formula holds in some or all equilibrium computations.

While the iBG model of (Gutierrez, Harrenstein, and Wooldridge 2013) is useful for the purposes of exposition, it is not a realistic model of concurrent programs. Moreover, (Gutierrez, Harrenstein, and Wooldridge 2013) provides no language for reasoning *about* the equilibria of systems: such reasoning must be carried out at the meta-level. This paper fills those gaps. First, we present a computational model that is more appropriate for modelling concurrent systems than the iBG model. In this model, the goals (and thus preferences) of players are given as temporal logic formulae that the respective player aspires to satisfy. After exploring some properties of this model, we introduce *Equilibrium Logic* (EL) as a formalism for reasoning about the equilibria of such systems. EL is a branching time logic that provides a new path quantifier $[\mathbf{NE}]\varphi$, which asserts that φ holds *on all Nash equilibrium computations of the system*. Thus, EL supports reasoning about equilibria directly in the object language. We then investigate some properties of this logic.

In particular in this paper we show that via a logical characterisation of equilibria one can check useful properties of strategy profiles. We consider four logics for players' goals: LTL (Pnueli 1977), CTL (Clarke and Emerson 1981), the linear-time μ-calculus (Vardi 1988), and the branching-time μ-calculus (Kozen 1983). Based on our logical characterisation of equilibria in infinite games, three problems are studied: STRATEGY-CHECKING, NE-CHECKING, and EQUIVALENCE-CHECKING, all of which are shown to be in PSPACE or in EXPTIME depending on the particular problem and temporal logic at hand. We also study the computational complexity of checking equilibrium properties, which can be expressed in EL. We show that the problem is 2EXPTIME-hard, even for LTL or CTL goals. This result shows, in turn, that checking equilibrium properties is equally hard in the linear-time and in the branching-time spectra. A summary of key results is given at the end of the paper. Note that most proofs are omitted due to lack of space.

2 Models

Before giving formal definitions, let us start by describing a situation that can naturally be modelled as a game-like concurrent and multi-agent system.

Example 1. Consider a situation in which two agents can request a resource from a dispatch centre infinitely often; assume both agents will always eventually need the resource. The centre's behaviour is as follows:

1. if only one of the agents requests the resource, it gets it;
2. if both agents request the resource, neither agent gets it;
3. if one agent requested the resource twice in a row while the other agent did not do so, the latter agent gets the resource for ever after (thus, punishing greedy behaviour).

Because of 2 and 3 it may be the case that an agent (or both) fails to achieve its goal of being granted the resource infinitely often. But, of course, we can see a simple solution: if both agents request the resource alternately, then both agents get their goals achieved. Indeed, a game-theoretic analysis reveals that *in all equilibria of this system both agents get the resource infinitely often*.

In order to model this kind of situation, we will define a model for concurrent strategic interactions. Using this model, in Section 5 we present a formal model of the above example, and we then formally analyse its equilibrium properties (Example 14).

The basic formal model for capturing strategic interactions is a graph-based structure that is used for various distinct purposes. It will be used for both the *model* of game-like concurrent systems we are interested in modelling and the *strategies* of players in these systems. Depending on how we use and instantiate our model to each of the above applications they may acquire a particular name and have a refined structure. As will be clear from their definition (below), our basic model generalises both Kripke frames and transition systems, among others, thus allowing them to be used in very many different contexts. More importantly, compositions of these models naturally represent the behaviour of synchronous, multi-agent, and concurrent systems with interleaving semantics as well as of asynchronous systems (Nielsen and Winskel 1995). Formally, let

$$M = (V, E, v^0, \Omega, \Lambda, \omega, \lambda)$$

be a Λ-labelled Ω-model M, where V is the set of vertices[1] of M, $v^0 \in V$ is the initial vertex, $E \in V \times V$ is the set of edges[2], and $\omega : V \to 2^\Omega$ and $\lambda : E \to 2^\Lambda$ are two functions, the former indicating the set of 'properties' of a vertex and the latter the ways to go/move from one vertex to another. Based on M, some sets can be defined. The set of *transitions*:

$$T_M = \{(v, a, v') \in V \times \Lambda \times V \mid (v, v') \in E \wedge a \in \lambda(v, v')\};$$

and the sets of sequences[3] of adjacent vertices and transitions starting at v^0, which we denote by V^*_M and T^*_M.

A model is total if for every $v \in V$ there is $v' \in V$ and $a \in \Lambda$ such that $(v, a, v') \in T_M$; it is, moreover, Λ-total if for every $v \in V$ and every $a \in \Lambda$ there is $v' \in V$ such that $(v, a, v') \in T_M$. Observe that if M is total the set T^*_M contains only infinite sequences. The sets T^*_M and V^*_M induce two additional sets of sequences, one over the elements in Λ (the *action* names labelling the transitions) and another one over the elements in 2^Ω (the *properties* that hold, or can be observed, in the vertices of the model); namely the sets

$$\mathcal{A}^*_M = \{a, a', \ldots \mid (v^0, a, v'), (v', a', v'') \ldots \in T^*_M\}, \text{and}$$

$$\mathcal{P}^*_M = \{\omega(v^0), \omega(v'), \omega(v''), \ldots \mid v^0, v', v'', \ldots \in V^*_M\}.$$

Hereafter, for all sets and in all cases, we may omit their subscripts whenever clear from the context. Given a sequence ρ (of any kind), we write $\rho[0], \rho[1], \ldots$ for the first, second, ... element in the sequence; if ρ is finite, we write $last(\rho)$ to refer to its last element. We also write $|\rho|$ for the size of a sequence. The empty sequence is denoted by $\rho[\,] = \epsilon$ and has size 0. Restrictions to parts of a sequence and operations on them are useful. Given $k, k' \in \mathbb{N}$, with $k \leq k'$, we write $\rho[0 \ldots k]$ for the sequence $\rho[0], \rho[1], \ldots, \rho[k]$ (an initial segment of ρ), $\rho[k \ldots k']$ for the sequence $\rho[k], \ldots, \rho[k']$, and $\rho[k \ldots \infty]$ for the infinite sequence $\rho[k], \rho[k+1], \ldots$. We also write, *e.g.*, $\rho[k \ldots k')$ if the element $\rho[k']$ of the sequence is not included. Given a finite sequence ϱ, we write $\varrho \in \rho$ if $\varrho[k] = \rho[k]$ for all $0 \leq k \leq |\varrho|$. We also write $\varrho; \rho$ for the binary operation on sequences/words—and resulting run—of concatenating a finite run ϱ with a run ρ. When working with sequences, we assume the standard laws on them, in particular, w.r.t. concatenation ";" with the empty sequence ϵ we have $\epsilon; \rho = \rho$, for any ρ, and $\rho; \epsilon = \rho$, for any finite ρ.

We find it useful to associate *input and output languages* with models. The input language $\mathcal{L}_i(M)$ of M is defined to be $\mathcal{A}^*_M$ and the output language $\mathcal{L}_o(M)$ of M is defined to be $\mathcal{P}^*_M$. Given a set of models $\vec{M} = \{M_1, \ldots, M_n\}$, the input language of $\vec{M}$ is defined to be

$$\mathcal{L}_i(\vec{M}) = \bigcap_{1 \leq j \leq n} \mathcal{L}_i(M_j).$$

[1] We may write 'nodes' or 'states' when talking about vertices.
[2] We also call them 'events' or 'actions'.
[3] We also say 'words' or 'strings' when talking about sequences.

The set $\mathcal{L}_i(\vec{M})$ determines synchronised runs $V^*_{\vec{M}}$ for $\vec{M}$, *i.e.*, sequences $(v^0_1,\ldots,v^0_n),(v'_1,\ldots,v'_n),\ldots \in (V_1\times\ldots\times V_n)^*$ such that

$$\bigwedge_{1\leq j\leq n}(v^0_j,a,v'_j),(v'_j,a',v''_j),\ldots \in T^*_{M_j},$$

for some $a,a',\ldots\in\mathcal{L}_i(\vec{M})$. The set $V^*_{\vec{M}}$, in turn, determines the output language $\mathcal{L}_o(\vec{M})$ of $\vec{M}$, defined to be all sequences

$$\bigcup_{1\leq j\leq n}\omega_j(\rho[0]),\ \bigcup_{1\leq j\leq n}\omega_j(\rho[1]),\ldots\in(2^{\Omega_1\cup\cdots\cup\Omega_n})^*$$

where $\rho\in V^*_{\vec{M}}$ and $\omega_j(\rho[k])$, with $0\leq k<|\rho|$, is the application of the ω_j of M_j to the jth component of each $\rho[k]$.

Let $\mathcal{L}$ function equally for an input or output language of a model M or compound system $\vec{M}$; moreover, let $\mathcal{L}[\varrho]$ be the (sub)language of (sub)words $\{\rho[|\varrho|\ldots\infty]\mid \varrho\in\rho\in\mathcal{L}\}$.

There is an induced tree language $\mathit{TreeL}(\mathcal{L})$ for every (word) language $\mathcal{L}$, defined as:

$$\mathit{TreeL}(\mathcal{L})=\{\mathcal{T}\text{ is a tree}\mid \rho\in\mathcal{L},\text{ for each path }\rho\text{ in }\mathcal{T}\},$$

that is, the *tree language* of a *word language* $\mathcal{L}$ is the set of all trees all of whose paths are in $\mathcal{L}$.

Remark 2. Note that any non-empty word language induces a tree language comprising infinitely many trees, if such trees are allowed to be non-deterministic. For instance, the (singleton) word language $\mathcal{L}=\{a\}$ induces the tree language $\mathit{TreeL}(\mathcal{L})$ containing the following trees: the empty tree, the tree with one a-labelled branch, the tree with two a-labelled branches, ..., and the infinite tree with infinitely many a-labelled branches. However, if non-determinism is not allowed (or at least restricted in some way) some finite word languages may always induce finite tree languages. For the sake of *generality* we impose no restrictions at this point.

Our models support two useful operations: *restriction* and *projection*. The former selects a subset of the output language; a restriction with respect to a subset of the input language. We denote by $\mathcal{L}_o(M)|_{\mathcal{L}}$, where $\mathcal{L}\subseteq\mathcal{L}_i(M)$, such a subset of the output language. Projection, on the other hand, takes the sequences in the output language and forgets the elements in some subset of Ω. We write $\mathcal{L}_o(M)|_{\Omega'}$ for such an operation and resulting set, which is formally given by:

$$\{\rho[0]\cap\Omega',\rho[1]\cap\Omega',\ldots\in(2^{\Omega'})^*\mid\rho\in\mathcal{L}_o(M)\}.$$

3 Games and Strategies

Games. Using the model given in Section 2, we will define reactive games, a class of multi-player nonzero-sum games. In a reactive game a finite set of players interact with each other by assigning values to variables they have control over. The game has a designated initial state and the values given to the variables at each round determine the next state of the game. The game is played for infinitely many rounds. Players in a reactive game have goals they wish to satisfy. Such goals are expressed as temporal logic formulae. Formally, a reactive game is defined as follows. A *reactive* game (sometimes just called a "game") is a structure:

$$G=(N,C,(C_i)_{i\in N},(\gamma_i)_{i\in N},X,A)$$

where $N=\{1,\ldots,n\}$ is a set of *agents* (the players of the game), $C=\{p,q,r,\ldots\}$ is a set of *controlled variables*, $C_i\subseteq C$ is the set of variables under the *unique control* of player i, and γ_i is a formula (of some logical system[4]) over a set $X=\{x,y,\ldots\}$ of propositions; formula γ_i describes the *goal* that player i wants to achieve. There is a requirement on C: the sets of variables $C_1,\ldots,C_n$ form a partition of C, that is, $C_i\cap C_j=\emptyset$ for all $i\neq j\in N$, and $C=C_1\cup\cdots\cup C_n$. A *choice* c_i for agent $i\in N$ is an assignment of values for the variables under its control. Let Ch_i be the set of choices for agent i. A *choice vector* $\vec{c}=(c_1,\ldots,c_n)$ is a collection of choices, one for each player. Let Ch be the set of all choice vectors. And A—the *"arena" or "board"* where the game is played—is a Λ-total Ω-model such that $\Lambda=\mathit{Ch}$ and $\Omega=X$.

Note that Λ-totality ensures in a simple manner that a reactive game is played for an infinite number of rounds without imposing further consistency or validity conditions on strategies. Moreover, it does not limit our modelling power.

Remark 3. Reactive games can be considered as a metamodel of *infinite* games of *control* since their definition does not specify the logic each γ_i belongs to, the kinds of strategies used in the game, the types of variables the players have control over, what the outcome of a game would be given a set of players' strategies, or when the outcome of a game makes a player's goal satisfied. As we will see, different kinds of games and results will arise from different choices with respect to these properties. All we know for now is that (1) the games are played for *infinitely* many rounds, (2) the players have *unique control* over some given set of variables, and that (3) they have goals given in a *logical form*.

Strategies. Since in a reactive game a play is infinite, it is natural to think of a strategy for a player i as a function $f_i:E^*\to\mathit{Ch}_i$ or as $f'_i:V^*\to\mathit{Ch}_i$, that is, as a function from what has been played so far (or at least what a player *knows so far*) to a choice c_i for player i. To formalise this, we use a strategy model that is *finite*, *simple*, and *expressive* enough for most computational purposes. Our definition of strategies is based on the model in Section 2. Similar representations have been used to study, *e.g.*, 'repeated games' in game theory (Osborne and Rubinstein 1994, pp. 140-143).

Formally, we define a *strategy* σ_i for player i in a reactive game $G=(N,C,(C_i)_{i\in N},(\gamma_i)_{i\in N},X,A)$ to be a structure

$$\sigma_i=(Q_i,q^0_i,\delta_i,\tau_i)$$

modelled as a structure $M_i=(V,E,v^0,\Omega,\Lambda,\omega,\lambda)$ in which $Q_i=V$ is a finite and non-empty set of *states*, $q^0_i=v^0$ is the *initial* state, $\delta_i:Q_i\times\Lambda\to 2^{Q_i}$, with $\Lambda=2^{X_A}$, is the *transition function* given by T_{M_i}, and $\tau_i=\omega:Q_i\to\mathit{Ch}_i$ is a *choice function*. As one requires that a strategy for player i is able to *react* to any possible valid behaviour/strategy of the others, we only consider as *valid* the strategies that are based on structures M_i where δ_i is total.

[4] We will consider several logical temporal languages for γ_i, *e.g.*, LTL (Pnueli 1977), CTL (Clarke and Emerson 1981), or fixpoint linear-time and branching-time modal logics (Vardi 1988; Kozen 1983). At this point all definitions can be made leaving this choice open, which will make our framework more general.

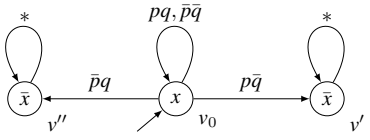


Figure 1: An arena, where $* = \{pq, \bar{p}q, p\bar{q}, \bar{p}\bar{q}\}$.

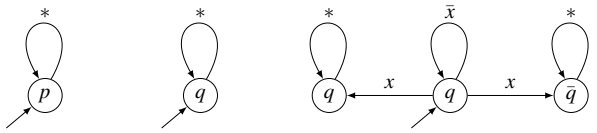


Figure 2: Strategy σ_1 for player 1 (left) and the strategies σ_2 (middle) and σ_2' (right) for player 2. Here $* = \{x, \bar{x}\}$.

Henceforth, given a game G with n players in N and a set of strategies $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$, we call $\vec{\sigma}$ or any subset of it a strategy profile. We write $\vec{\sigma}_{-S}$, with $S \subseteq N$, for $\vec{\sigma}$ without the strategies σ_i such that $i \in S$; we omit brackets if S is a singleton set. Also, we write $(\vec{\sigma}_{-i}, \sigma_i')$, with $1 \leq i \leq n$, for the strategy profile $\vec{\sigma}$ where σ_i is replaced with σ_i'.

Example 4. Consider a game with $N = \{1, 2\}$, $C_1 = \{p\}$, and $C_2 = \{q\}$ and arena as in Figure 1. There, we have $\omega_A(v^0) = x$, $\omega_A(v') = \omega_A(v'') = \bar{x}$. Moreover, the symbol $\bar{p}$ means $p := \bot$, p means $p := \top$ (and likewise for q and x). A possible strategy for player 1 would be to always play p and is depicted in Figure 2 (left). The exact outcome(s) of the game—to be defined next—can be determined only once the strategy for player 2 is given.

Our strategy model is simple but powerful; in particular, it can generate any word or tree ω-regular language. Formally:

Lemma 5. *Let $\mathcal{T}$ be an ω-regular tree—i.e., the unfolding of a finite, total graph. There is σ such that $\mathcal{T} \in TreeL(\mathcal{L}_o(\sigma))$.*

It is important to point out that the strategy σ may be non-deterministic. However, with respect to word languages, only deterministic strategies are needed. Specifically, because ω-regular words are ω-regular trees that do not branch, the following is an easy corollary of Lemma 5.

Corollary 6. *Let w be an ω-regular word. There is σ such that $w = \mathcal{L}_o(\sigma)$.*

Lemma 5 and Corollary 6 will be used to ensure the existence of strategies in a reactive game with ω-regular goals.

Outcomes and composition of strategies. Given a set of strategies $(\sigma_i)_{i \in N}$, which hereafter we will denote by $\vec{\sigma}$ whenever the set N is clear from the context, the *histories* (of choices) when playing such a set of strategies in a game $G = (N, C, (C_i)_{i \in N}, (\gamma_i)_{i \in N}, X, A)$ are the sequences in the input language of A, denoted by $\mathcal{L}_i^{\vec{\sigma}}(A)$, given by

$$\mathcal{L}_i^{\vec{\sigma}}(A) = \mathcal{L}_o(\vec{\sigma})|_{\mathcal{L}_i(\vec{\sigma}) \cap \mathcal{L}_o(A)[\vec{q}^0]},$$

where $\vec{q}^0 = (\tau_1(q_1^0), \ldots, \tau_n(q_n^0))$.

Informally, since $\mathcal{L}_o(\vec{\sigma})$, and hence $\mathcal{L}_i^{\vec{\sigma}}(A)$, is restricted to $\mathcal{L}_i(\vec{\sigma}) \cap \mathcal{L}_o(A)[\vec{q}^0]$, we know that when playing a strategy profile there is an alternation in the interaction between strategies and the arena, with the strategies making transitions only after a transition in the arena has been made.

The histories of a game with respect to a strategy profile $\vec{\sigma}$ record the choices that the players make based on such a given strategy profile $\vec{\sigma}$. These choices, in turn, determine the *outcomes* of the game, denoted by $\mathcal{L}_o^{\vec{\sigma}}(A)$ and defined as

$$\mathcal{L}_o^{\vec{\sigma}}(A) = \mathcal{L}_o(A)|_{\mathcal{L}_i^{\vec{\sigma}}(A)}.$$

Then, whereas *histories* are sequences in the *input* language of A, *outcomes* are sequences in its *output* language.

As defined, the outcomes of a game form a set of words or infinite sequences over $(2^{\Omega_A})^*$. However, they naturally define a set of trees with respect to the tree-unfolding of A. We write $unf(A, v)$ for the usual tree-unfolding of A—when A is seen as a graph—with respect to a given vertex $v \in V$; we simply write $unf(A)$ whenever $v = v^0$. Define the set

$$Tree(A) = \{T \mid T \text{ is a subtree of } unf(A)\}.$$

Thus, given a strategy profile $\vec{\sigma}$ in a reactive game G, the tree/branching outcomes of G are the trees in the set

$$TreeL(\mathcal{L}_o^{\vec{\sigma}}(A)) \cap Tree(A).$$

Regardless of whether we are talking about word outcomes or tree outcomes, we will uniformly denote by $Out(G, \vec{\sigma})$ the outcomes of a game G when playing the set of strategies $\vec{\sigma}$. Similarly we will denote by Out_G the set of all outcomes of the game G, that is, with respect to all valid sets of strategies, and omit the subscript G whenever which game G we are referring to is either clear or irrelevant. It is worth noting that Out is $\mathcal{L}_o(A)$ in case of word outcomes and, therefore, is $TreeL(\mathcal{L}_o(A)) \cap Tree(A)$ in case of tree outcomes. Also, note that because we allow *non-determinism*, the set $Out(G, \vec{\sigma})$ is not necessarily a singleton, as illustrated next.

Example 7. Consider again the game in Example 4 and the two strategies for player 2 depicted in Figure 2. The strategy profile $\vec{\sigma} = (\sigma_1, \sigma_2)$ induces the unique (word) outcome $x^\omega = x, x, \ldots$ in Out; the strategy profile $\vec{\sigma}' = (\sigma_1, \sigma_2')$, on the other hand, induces infinitely many outcomes, namely those sequences given by the ω-regular expression $x^\omega \cup x.\bar{x}^\omega$. The reason why $\vec{\sigma}'$ induces more than one outcome is because σ_2' is non-deterministic; thus, multiple outcomes are possible even when A is deterministic.

Also, given a set of *deterministic strategies* one can have a reactive game where multiple outcomes are possible if A is a *non-deterministic arena*, since the same players' choice can lead to different successor vertices in A. In this case the next state of the game is selected non-deterministically in A, *i.e.*, it is not under the control of any of the players.

Remark 8. Observe that the reactive games model strictly generalises the iBG model (Gutierrez, Harrenstein, and Wooldridge 2013), which can be represented as a reactive game where the arena is an *implicitly* defined clique whose nodes are the valuations of the variables the players control, goals are LTL formulae, and strategies are deterministic.

4 Equilibria in Logical Form

Because players have goals, which they wish to satisfy, and their satisfaction depends on the outcomes—whether word or tree outcomes—of the game, the players may prefer some sets of outcomes over others. To formalise this situation we define, for each player i, a *preference* relation $\leq_i$ over 2^{Out}. Even though $\leq_i$ can be any binary relation over 2^{Out}, it is natural to assume that it is a preorder, that is, a reflexive and transitive relation. We write $<_i$ whenever $\leq_i$ is strict—or asymmetric, *i.e.*, $X \leq_i X'$ implies that $X' \leq_i X$ does not hold. Because strategy profiles induce sets of outcomes, we abuse notation by writing $\vec{\sigma} \leq_i \vec{\sigma}'$ to mean $Out(G,\vec{\sigma}) \leq_i Out(G,\vec{\sigma}')$, that is, that player i does not prefer the set of outcomes $Out(G,\vec{\sigma})$ over the set of outcomes $Out(G,\vec{\sigma}')$.

Based on players' preferences, a notion of *equilibrium* can be defined. We provide the definition of the, arguably, main concept of equilibrium—sometimes called solution concept—in game theory, namely, *Nash equilibrium*. However, many solution concepts can be found in the literature, *e.g.*, dominant strategy, subgame perfect Nash, correlated, amongst others. We say that a strategy profile $\vec{\sigma}$ is a Nash equilibrium if for every player i and strategy σ_i' we have

$$(\vec{\sigma}_{-i}, \sigma_i') \leq_i \vec{\sigma}.$$

Intuitively, a Nash equilibrium formalises the idea that no player can be better off (have a beneficial deviation) provided that all other players do not change their strategies. Let $NE(G)$ be the set of Nash equilibria of the game G.

Remark 9. Note that since strategies or arenas can be non-deterministic—hence multiple outcomes can be induced, our definition of Nash equilibrium is phrased in terms of *sets* of outcomes, rather than in terms of single outcomes only. Even though the definition of equilibrium is given w.r.t. preferences over sets of outcomes, we can think of such a definition as based on a *preference relation* over strategy profiles instead, since strategy profiles induce sets of outcomes. Thus a preference relation allows one to define equilibria in a general way, not only for binary goals as in this paper.

We can think of equilibria with respect to the goals the players of the game wish to satisfy. To make this statement precise, we need to know which logic the goals of the players belong to and when a set of outcomes satisfy such goals, that is, we need to define a semantics of players' goals w.r.t. 2^{Out}—*i.e.*, with respect to the outcomes of a game.

We can then abstractly think of the existence of a *satisfaction* relation "$\models$" between sets of outcomes and logical formulae, that is, a binary relation indicating whether a given goal γ_i for player i is satisfied or not by a set of outcomes $Out(G,\vec{\sigma})$ in a game G played with a strategy profile $\vec{\sigma}$. Assuming the existence of a denotation function $[\![\cdot]\!]$ from goals to sets of outcomes, we can then write

$$Out(G,\vec{\sigma}) \models \gamma_i \qquad \text{if and only if} \qquad Out(G,\vec{\sigma}) \subseteq [\![\gamma_i]\!].$$

Again, as strategy profiles induce sets of outcomes, we abuse notation and write $\vec{\sigma} \models \gamma_i$ if $Out(G,\vec{\sigma}) \models \gamma_i$. And, in order to simplify notations used in the paper, we will also write $[\![\vec{\sigma}]\!]$ for either the set of outcomes or the associated set of infinite sequences of vertices in V_A^* induced by $\vec{\sigma}$; which one we are referring to will always be clear from the context.

Based on the definitions above one can now formally state with respect to the goals $(\gamma_i)_{i\in N}$ of the game, when $\vec{\sigma}$ is a Nash equilibrium. We say that $\vec{\sigma}$ is a Nash equilibrium if, for every player i and for every strategy σ_i', we have that

$$(\vec{\sigma}_{-i}, \sigma_i') \models \gamma_i \Longrightarrow \vec{\sigma} \models \gamma_i.$$

Remark 10. Since our model generalises Kripke structures and Labelled Transition Systems (among other structures), it can be used to give the standard semantics of all usual linear-time and branching-time temporal logics. In this paper, we will assume that players have *ω-regular goals*. In particular, in case of linear-time goals we will let each γ_i be either a linear-time μ-calculus or an LTL formula, strategies be deterministic, and outcomes be word outcomes; in case of branching-time goals, we will assume that the goals are either CTL or μ-calculus formulae, that the strategies can be non-deterministic, and that the outcomes are tree outcomes.

The details of the semantics of these logics need not be given to obtain the results in this paper. All one needs to know is the complexities of their satisfiability and model checking problems, which are as follows: for satisfiability CTL and the μ-calculus are EXPTIME, whereas LTL and the linear-time μ-calculus are PSPACE; for model checking w.r.t. a product of transition systems, the logics LTL, CTL, and linear-time μ-calculus are PSPACE, while the μ-calculus is EXPTIME. The next question becomes relevant:

Given: Game G, strategy profile $\vec{\sigma}$, goal γ.
STRATEGY-CHECKING: Is it the case that $\vec{\sigma} \models \gamma$?

Clearly, the answer to this question depends on the logic to which the formula γ belongs. The following lemma answers this question for various temporal logics.

Lemma 11. *The* STRATEGY-CHECKING *problem for LTL, CTL, and linear-time μ-calculus goals is PSPACE-complete. For modal μ-calculus goals the problem is in EXPTIME.*

Using STRATEGY-CHECKING we can, moreover, show that the following problem, namely, membership of a strategy profile in the set of Nash equilibria of a reactive game, may be harder only in the branching-time case.

Given: Game G, strategy profile $\vec{\sigma}$.
NE-CHECKING: Is it the case that $\vec{\sigma} \in NE(G)$?

Formally, we have

Lemma 12. *The* NE-CHECKING *problem for LTL and linear-time μ-calculus goals is PSPACE-complete. For CTL and μ-calculus goals the problem is EXPTIME-complete.*

Equivalences of equilibria. The characterisation of equilibrium with respect to the goals of the game given above provides a natural way of comparing strategy profiles, and hence of comparing equilibria, in a logical way. But first, we provide a notion of equivalence of strategy profiles purely based on the outcomes they induce and later on a weaker definition with a more logical flavour. Given a game G, we say that two strategy profiles $\vec{\sigma}$ and $\vec{\sigma}'$ are equivalent, and

write $\vec{\sigma} \sim \vec{\sigma}'$ in such a case, if and only if they induce the same set of outcomes, that is, iff $[\![\vec{\sigma}]\!] = [\![\vec{\sigma}']\!]$.

Even though the definition of $\sim$ immediately provides a definition for equivalence between equilibrium strategy profiles, such a definition is rather strong. Instead, one would like a definition where only the satisfaction of goals was taken into account. Having this in mind, we propose a weaker, logically based definition of equivalence between strategy profiles. Formally, given a game G, we say that two strategy profiles $\vec{\sigma}$ and $\vec{\sigma}'$ are logically equivalent, and write $\vec{\sigma} \sim_\gamma \vec{\sigma}'$ in such a case, if and only if, they induce sets of outcomes that satisfy the same set of goals of the game, that is, iff for every goal in $(\gamma_i)_{i\in N}$ of G we have that

$$\vec{\sigma} \models \gamma_i \Longleftrightarrow \vec{\sigma}' \models \gamma_i.$$

Formally, the decision problem one wants to solve is:

Given: Game G, strategy profiles $\vec{\sigma}, \vec{\sigma}'$.
EQUIVALENCE-CHECKING: Does $\vec{\sigma} \sim_\gamma \vec{\sigma}'$ hold?

An immediate consequence of Lemma 11 is:

Corollary 13. *The* EQUIVALENCE-CHECKING *problem for LTL, CTL, and linear-time μ-calculus goals is PSPACE-complete. For μ-calculus goals the problem is in EXPTIME.*

5 Equilibrium logics

We now introduce two logics for expressing equilibrium properties of (reactive) games: these two logics are closely related to the branching time logics CTL and CTL* (see, *e.g.*, (Emerson and Halpern 1986)). We refer to our basic logical framework as *Equilibrium Logic*, (EL), and will refer to the two versions of this logic as EL (roughly corresponding to CTL) and EL* (roughly corresponding to CTL*). Since EL* will be defined as an extension of CTL*, the logic LTL will also appear as a syntactic fragment.

The basic idea of Equilibrium Logic is to extend the logic CTL* by the addition of two modal quantifiers, which we will write as "$[\mathbf{NE}]$" and "$\langle\mathbf{NE}\rangle$". In Equilibrium Logic, the modalities $[\mathbf{NE}]$ and $\langle\mathbf{NE}\rangle$ quantify over *paths that could arise as the consequence of processes (agents/players) selecting strategies in equilibrium*. For example, if we are dealing with Nash equilibrium, then an EL formula $[\mathbf{NE}]\mathbf{F}p$ (where $\mathbf{F}$ is the "eventually" LTL modality) means that on all Nash equilibrium computations—*i.e.*, on all computations (paths or trees) that correspond to runs or plays where a set of agents/players use a strategy profile in equilibrium—eventually p will hold. In this way, we can use Equilibrium Logic to directly reason about the equilibrium properties of game-like concurrent systems. Equilibrium Logic is parameterised by a solution concept, which determines the outcomes over which the "equilibrium modalities" quantify. For now, we consider Nash equilibrium as our by-default solution concept, but of course others can be considered too.

Syntax. The syntax of Equilibrium Logic is defined w.r.t. a set X of propositions, by the following grammars:

Path Formulae: $\varphi ::= \psi \mid \theta \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi\,\mathbf{U}\,\varphi$
State Formulae: $\psi ::= x \mid \mathbf{A}\varphi$
Nash Formulae: $\theta ::= [\mathbf{NE}]\varphi$

where $x \in X$. Thus, as in CTL*, path formulae φ express properties of paths (*cf.* LTL), while state formulae ψ express properties of states. In addition, Nash equilibrium formulae also express properties of paths, but only if such paths are induced by strategy profiles in equilibrium. We take the universal modalities $\mathbf{A}$ and $[\mathbf{NE}]$ as primitives, and define the existential modalities $\mathbf{E}$ and $\langle\mathbf{NE}\rangle$ as their duals as usual:[5] $\mathbf{E}\varphi \equiv \neg\mathbf{A}\neg\varphi$ and $\langle\mathbf{NE}\rangle\varphi \equiv \neg[\mathbf{NE}]\neg\varphi$.

Semantics. The semantics of EL formulae is given here w.r.t. a reactive game $G = (N, C, (C_i)_{i\in N}, (\gamma_i)_{i\in N}, X, A)$, where $A = (V_A, E_A, v_A^0, \Omega_A = X, \Lambda_A = Ch, \omega_A, \lambda_A)$. The semantics of path formulae ("$\models_{\mathcal{P}}$") is essentially the same for LTL, and so given with respect to paths π in A, with two additional rules for state and equilibrium formulae, respectively. The semantics of state formulae ("$\models_{\mathcal{S}}$") is given with respect to states/vertices $v \in V$ of A. The semantics of equilibrium formulae ("$\models_{\mathcal{E}}$") is given with respect to the set of Nash equilibria of G. Let ϱ be a run of A, *i.e.*, an infinite sequence of states over V_A^* starting at v_A^0, and $t \in \mathbb{N}$. Define

$(G, \varrho, t) \models_{\mathcal{P}} \psi$	iff	$(G, \varrho, t) \models_{\mathcal{S}} \psi$ for state formulae ψ.
$(G, \varrho, t) \models_{\mathcal{P}} \theta$	iff	$(G, \varrho, t) \models_{\mathcal{E}} \theta$ for equilibrium formulae θ.
$(G, \varrho, t) \models_{\mathcal{P}} \neg\varphi$	iff	$(G, \varrho, t) \models_{\mathcal{P}} \varphi$ does not hold.
$(G, \varrho, t) \models_{\mathcal{P}} \varphi \vee \varphi'$	iff	$(G, \varrho, t) \models_{\mathcal{P}} \varphi$ or $(G, \varrho, t) \models_{\mathcal{P}} \varphi'$
$(G, \varrho, t) \models_{\mathcal{P}} \mathbf{X}\varphi$	iff	$(G, \varrho, t+1) \models_{\mathcal{P}} \varphi$
$(G, \varrho, t) \models_{\mathcal{P}} \varphi\,\mathbf{U}\,\varphi'$	iff	$(G, \varrho, t') \models_{\mathcal{P}} \varphi'$ for some $t' \geq t$ and $(G, \varrho, k) \models_{\mathcal{P}} \varphi$ for all $t \leq k < t'$.

The satisfaction relation "$\models_{\mathcal{S}}$" for state formulae is defined as follows:

$(G, \varrho, t) \models_{\mathcal{S}} x$	iff	$x \in \omega_A(\varrho[t])$
$(G, \varrho, t) \models_{\mathcal{S}} \mathbf{A}\varphi$	iff	$(G, \varrho', t) \models_{\mathcal{P}} \varphi$ for all ϱ' such that $\varrho[0 \ldots t) \in \varrho'$.

And the satisfaction relation "$\models_{\mathcal{E}}$" for equilibrium formulae is defined as follows:

$(G, \varrho, t) \models_{\mathcal{E}} [\mathbf{NE}]\varphi$	iff	$(G, \varrho', t) \models_{\mathcal{P}} \varphi$ for all $\varrho' \in [\![\vec{\sigma}]\!]$ with both $\vec{\sigma} \in NE(G)$ and $\varrho[0 \ldots t) \in \varrho'$

We say that G is a model of φ (in symbols $G \models \varphi$) if and only if $(G, \varrho, 0) \models_{\mathcal{P}} \varphi$ for all ϱ of G, that is, for all paths or sequences of states of A starting at v_A^0—sequences in V_A^*.

Example 14. The situation of Example 1 is modelled by the game $G = (N, C, C_1, C_2, \gamma_1, \gamma_2, X, A)$, where $X = \{x, y\}$, $C_1 = \{p\}$, $C_2 = \{q\}$ and A is the arena as in Figure 3. Intuitively, x and y signify player 1 and player 2 get the resource, respectively. Setting p to true corresponds to player 1 requesting the resource, while setting p to false means refraining from doing so. Similarly, for q and player 2. The goals

[5] All usual abbreviations for the Boolean operators not explicitly given in the grammars above are also assumed to be defined.

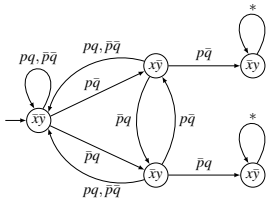


Figure 3: Formal model of the system in Example 1

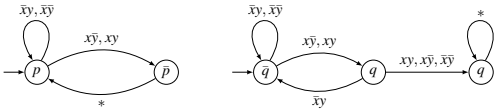


Figure 4: Strategies σ_1 (left) and σ_2 (right).

of the players are the LTL goals $\gamma_1 = \mathbf{GF}x$ and $\gamma_2 = \mathbf{GF}y$. The structures in Figure 4 are two strategies σ_1 and σ_2 for player 1 and 2, respectively. Strategy σ_1 requests the resource by playing p until it gets it, and then refrains from doing so by playing $\bar{p}$ once. Strategy σ_2 toggles but between $\bar{q}$ and q, beginning with the former, and additionally threatens to set q to true for ever if p is true (at least) twice in a row, which can be deduced from x being set to true or $\bar{y}$ while having set q to true previously. The strategy profile $\vec{\sigma} = (\sigma_1, \sigma_2)$ yields $\mathcal{L}_i^{\vec{\sigma}}(A) = \{(p\bar{q};\bar{p}q)^\omega\}$ and $\mathcal{L}_o^{\vec{\sigma}}(A) = \{\bar{x}\bar{y};(x\bar{y};\bar{x}y)^\omega\}$. The run $\rho = \bar{x}\bar{y}, x\bar{y}, \bar{x}y, x\bar{y}, \bar{x}y, \ldots$ in $\mathcal{L}_o^{\vec{\sigma}}(A)$ satisfies both players' goals and as such $\vec{\sigma}$ is a Nash equilibrium. Thus, we have $G \models \langle\mathbf{NE}\rangle(\gamma_1 \wedge \gamma_2)$. This is no coincidence, as we have for this game that $G \models [\mathbf{NE}](\gamma_1 \wedge \gamma_2)$: in all Nash equilibria both players' goals are satisfied. This contrasts sharply with $\mathbf{A}(\gamma_1 \wedge \gamma_2)$, which does not hold in the game G.

This example shows that even in small concurrent systems it is not obvious that a Nash equilibrium exists—let alone that a temporal property holds in some or all equilibrium computations of the system. The example also shows an important feature of game-like concurrent systems: that even though a desirable property may not hold in general (*cf.*, $\mathbf{A}(\gamma_1 \wedge \gamma_2)$), it may well be the case that one can design or automatically synthesise a communication protocol or a synchronisation mechanism—directly from a logical specification—so that the desirable property holds when restricted to agents acting rationally (*cf.*, $[\mathbf{NE}](\gamma_1 \wedge \gamma_2)$).

Indeed, Equilibrium logics are specifically designed to *reason about what can be achieved in equilibrium and what cannot*, while abstracting away from the particular strategies that the agents/players of the system/game may use.

Expressivity. Observe that apart from $[\mathbf{NE}]$ all other operators are intended to have the same meaning as in CTL*. However, note that the semantics of $\mathbf{A}\varphi$ is not quite the same as its counter-part in CTL* because of the additional condition "such that $\varrho[0\ldots t) \in \varrho'$." In the standard semantics of CTL* it should be "for all ϱ' starting at $\varrho[t]$" instead. In other words, in CTL* the path used to reach the state $\varrho[t]$ is forgotten. This information is actually needed only for the semantics of the equilibrium modality $[\mathbf{NE}]$, but must be remembered throughout. Formally, we have

Proposition 15. *Let G be a game and φ be a $[\mathbf{NE}]$-free EL* formula. Then, for all runs ϱ and $t \in \mathbb{N}$, we have that*

$$(G, \varrho, t) \models \mathbf{A}\varphi \text{ iff } (G, \varrho', 0) \models \varphi,$$

for all ϱ' starting at $\varrho[t]$.

Thus, because of Proposition 15, we know that the semantics of EL* conservatively extends that of CTL*. To be more precise, the fact that we must "remember" how we got to a particular state when evaluating an equilibrium formula in that state means that, technically, EL is a *memoryfull* extension of CTL*; see, *e.g.*, (Kupferman and Vardi 2006).

More interesting is the question as to whether EL* can be translated into a logic for strategic reasoning already in the literature. This is largely an open question. However, a few observations can be made at this point. Firstly it is known that neither ATL* (Alur, Henzinger, and Kupferman 2002) nor Strategy logic (Chatterjee, Henzinger, and Piterman 2010) can express the existence of a Nash equilibrium in a multi-player game with deterministic strategies and LTL players' goals. This property is simply stated as $\langle\mathbf{NE}\rangle\top$ in EL, where we restrict ourselves to reactive games with deterministic strategies and arenas and only LTL players' goals. Thus, we can conclude that both *Strategy logic and ATL* are not as expressive as Equilibrium logics*.

On the other hand, the existence of Nash equilibria in multi-player games and w.r.t. deterministic strategies and LTL goals can be expressed in the Strategy logic developed in (Mogavero, Murano, and Vardi 2010a). Using their logical specification of the existence of a Nash equilibrium, we can encode the $\langle\mathbf{NE}\rangle$ operator (and hence any $\langle\mathbf{NE}\rangle\psi$, with LTL ψ), again when restricted to LTL goals and deterministic games and strategies. This gives a 2EXPTIME upper bound for the model checking problem of this fragment of our logic. Note that the fragment is not only syntactic since the underlying reactive games model is also being restricted.

Letting EL*[LTL] be the fragment of EL* given above, we can show (w.r.t. deterministic arenas and strategies):

Proposition 16. *The model checking problem for EL*[LTL] formulae is in 2EXPTIME.*

Together with the hardness result that we will give in the next section, we obtain that model checking EL*[LTL] formulae is an 2EXPTIME-complete problem.

6 Complexity

In this section we show that the model checking problem for Equilibrium logics is 2EXPTIME-hard, even for LTL or CTL players' goals. More precisely, the model checking problem for Equilibrium logics is stated as follows:

Given: Game G, EL* formula φ.
EL* MODEL-CHECKING: Does $G \models \varphi$ hold?

In fact, we will show a very strong claim: that the EL* MODEL-CHECKING problem is 2EXPTIME-hard, even for the EL* formula $\varphi = \langle\mathbf{NE}\rangle\top$ (the simplest equilibrium logic formula one can write) and for either LTL or CTL goals (two of the simplest temporal logics in the literature).

Hardness in the linear-time framework. In order to show 2EXPTIME-hardness of checking equilibrium properties given that the players of the game have linear-time goals, we provide a reduction from the LTL games in (Alur, La Torre, and Madhusudan 2003). Formally, we have

Lemma 17. *The* EL* MODEL-CHECKING *problem where players' goals are LTL formulae is 2EXPTIME-hard.*

Proof. (Sketch) We reduce the LTL games studied in (Alur, La Torre, and Madhusudan 2003), which are 2EXPTIME-complete, to EL* model checking $\varphi = \langle \mathbf{NE} \rangle \top$.

LTL games. An LTL game (Alur, La Torre, and Madhusudan 2003) is a two-player zero-sum game given by (G, ψ, u), where ψ is an LTL formula over a set of properties Σ, and

$$G = (V_G, V_0, V_1, \gamma : V \to 2^V, \mu : V \to \Sigma)$$

is a graph with vertices in V_G which are partitioned in player 0 vertices V_0 and player 1 vertices V_1. The transitions of the graph are given by γ; if $v' \in \gamma(v)$, for some $v, v' \in V_G$, then there is a transition from v to v', which we may assume is labelled by v'. Each vertex v has a set of properties p associated with it, which are given by μ; thus $p \in \Sigma$ holds in vertex v if $p = \mu(v)$. The graph G is assumed to be total, that is, for every $v \in V_G$, we have $\gamma(v) \neq \emptyset$.

The game is played for infinitely many rounds by each player choosing a successor vertex whenever it is their turn: player 0 plays in vertices in V_0 and player 1 plays in vertices in V_1. The game starts in the initial vertex $u \in V_G$. Playing the game defines an infinite word/sequence of adjacent vertices $w = v_0, v_1, v_2, \ldots$, such that $v_0 = u$ and for each v_k, with $k \in \mathbb{N}$, we have $v_{k+1} \in \gamma(v_k)$. An LTL game is won by player 0 if $w \models \psi$; otherwise player 1 wins. LTL games are determined, that is, there is always a winning strategy either for player 0 or for player 1. Checking whether player 0 has a winning strategy in an LTL game is a 2EXPTIME-complete problem (Alur, La Torre, and Madhusudan 2003).

We reduce the LTL game (G, ψ, u) to model checking the EL* formula $\varphi = \langle \mathbf{NE} \rangle \top$ in two stages. We first construct a reactive game G_1 that simulates the behaviour exhibited in the LTL game (G, ψ, u), and then transform G_1 into another reactive game G_2 with the property that player 0 has a winning strategy in (G, ψ, u) if and only if there is a Nash equilibrium in G_2, that is, if and only if $G_2 \models \varphi$.

Reactive games. The first construction (the one for G_1) comprises two players, 0 and 1, who control variables t_0 and t_1, respectively, whose domain is the set of vertices in the LTL with two additional actions, δ_0 and δ_1, to delay/skip when it is not their turn to play in the corresponding LTL game. Notice that in a reactive game they always must play, at each round, concurrently and synchronously. Their goals are $\gamma_0 = (\psi \wedge \mathbf{G}\neg l_0) \vee \mathbf{F} l_1$ and $\gamma_1 = (\neg\psi \wedge \mathbf{G}\neg l_1) \vee \mathbf{F} l_0$, respectively, where l_0 and l_1 are two fresh atomic propositions which hold only on two new states where, intuitively, player 0 loses the LTL game (state l_0) and player 1 loses the LTL game (state l_1). This translation from (G, ψ, u) to G_1 is polynomial. There are $|V_G| + 2$ vertices and $|\mu| + \mathcal{O}(|V_G|^3)$ transitions in the arena where G_1 is played. This first translation, from (G, ψ, u) to G_1, transforms a sequential zero-sum LTL game into a non-zero sum reactive game (where players make choices concurrently and synchronously) that is able to simulate the original LTL game. Players in G_1 are not forced to play consistently with plays in (G, ψ, u), but these players can be punished if they do not do so.

Then, we transform G_1 into G_2 in order to ensure that all (and only the) Nash equilibria of the reactive game G_2 correspond to plays where player 0 wins. The game G_2 comprises four players. The two new players, 3 and 4, control two Boolean variables t_2 and t_3, and have goals $\gamma_2 = \psi \vee (\mathbf{X} e)$ and $\gamma_3 = \psi \vee (\mathbf{X}\neg e)$, where e is a new atomic proposition. The arena of G_2 is linear in the size of the arena of G_1. It contains a copy of the vertices in game G_1 where the proposition $\neg e$ holds. In all other vertices—*i.e.,* those in G_1—the proposition e holds. The choice of where the game proceeds after the first round, namely, to the vertices already in G_1 or their copies, is ultimately controlled by the choices of players 2 and 3. In particular, in the former case, we have $t_2 = t_3$ (and move to a vertex where e holds) whereas, in the latter, we have $t_2 \neq t_3$ (and move to a vertex where $\neg e$ holds). This power allows these two players to deviate whenever ψ is not satisfied. As a consequence, all Nash equilibria of G_2 must satisfy formula ψ and hence the goals of players 2 and 3. Using this fact and Corollary 6, we build a winning strategy in the LTL game iff there is a Nash equilibrium in G_2. □

Hardness in the branching-time framework. Now, in order for us to show the 2EXPTIME-hardness of checking equilibrium properties given that the players of the game have *branching-time goals*, we use a reduction from the control-synthesis problem with respect to reactive environments (Kupferman et al. 2000). The control-synthesis problem for reactive environments is similar to the LTL game. However, in such a game player 1 may challenge player 0 with sets of successor vertices rather than with singletons, as in the LTL game. As a consequence, a play of the game can produce a tree instead of a word of vertices. In this game, player 0 wins if such a tree satisfies formula ψ; otherwise, player 1 wins. This game is also 2EXPTIME-complete and can be reduced to a reactive game in a way similar to that for LTL games. In particular, in this case, we can use Lemma 5 instead of Corollary 6 to ensure that strategies in these games always exist. Formally, we have

Lemma 18. *The* EL* MODEL-CHECKING *problem where players' goals are CTL formulae is 2EXPTIME-hard.*

Since LTL and CTL are syntactic fragments of, respectively, the linear-time μ-calculus and the modal μ-calculus, the hardness results can be transferred, to obtain:

Corollary 19. *The* EL* MODEL-CHECKING *problem, with players' goals given by linear-time μ-calculus or by modal μ-calculus formulae, is 2EXPTIME-hard.*

Notice that model checking the equilibrium operators of EL* formulae may require the solution of a number of "internal" synthesis problems for $(\gamma_i)_{i \in N}$ so that $\vec{\sigma} \in NE(G)$ can be checked and a run $\varrho \in [\![\vec{\sigma}]\!]$ can be determined before an EL* formula can be checked. This problem, known as reactive synthesis, is 2EXPTIME-complete for both LTL and CTL specifications, and seems to be the main source of the high complexity of checking equilibrium properties.

7 Concluding Remarks and Related Work

Complexity. We have shown that checking the equilibrium properties of a concurrent and multi-agent system is a computationally very hard problem as any interesting property is at least in PSPACE. On the positive side, we have also shown that in most cases the difficulty of checking equilibrium properties is independent of whether the goals of the players are given by linear-time or branching time temporal formulae. A summary of our complexity results is in Table 1.

	ST–CHECK	NE–CHECK	EQ–CHECK	EL*–MC
LTL	PSPACE	PSPACE	PSPACE	2EXPT$-$h
CTL	PSPACE	EXPT	PSPACE	2EXPT$-$h
TL$_\mu$	PSPACE	PSPACE	PSPACE	2EXPT$-$h
L$_\mu$	iEXPT	EXPT	iEXPT	2EXPT$-$h

Table 1: Overview of computational complexity results. In this table TL$_\mu$ stands for the linear-time μ-calculus, L$_\mu$ for the modal μ-calculus, ST–CHECK for STRATEGY-CHECKING, NE–CHECK for NE-CHECKING, EQ–CHECK for EQUIVALENCE-CHECKING, EL*–MC for EL* MODEL CHECKING, and iEXPT for in EXPTIME.

Logic.
Our work relates to logics either that are to reason about the behaviour of game-like systems or that are memoryfull. In (Kupferman and Vardi 2006) a memoryfull branching-time logic, called mCTL*, that extends CTL* was introduced. This logic has a special atomic proposition ("present") that allows one to reason about computations starting in the present or computations starting somewhere in the past (in CTL* all formulae are interpreted w.r.t. computations that start in the present). This logic is not more powerful than CTL*, but can be exponentially more succinct. No equilibrium issues are addressed there. In the linear-time framework, memoryfull logics have also been studied. In (Laroussinie, Markey, and Schnoebelen 2002) and extension of LTL with past is extended with an operator ("Now") that allows one to "forget the past" when interpreting logical formulae. This logic is exponentially more succinct than LTL with past, which in turn is exponentially more succinct than LTL; all such logics are nevertheless equally expressive. Thus, it seems that adding memory to temporal logics, either in the linear-time or in the branching-time framework, can make them more succinct but not any more powerful. Another shared feature of memoryfull temporal logics is that their verification problem is at least EXPSPACE. It follows from our 2EXPTIME-hardness results that checking equilibrium properties, which requires the use of a memoryfull logic, is even a harder problem. Another memoryfull logic, this time one designed to reason about game-like scenarios, is Strategy logic (Chatterjee, Henzinger, and Piterman 2010). Strategy logic has strategies as first-class objects in the language and based on this feature one can indirectly reason about equilibrium computations of a system. In Equilibrium logic reasoning is done the other way around: we can directly refer to equilibrium computations of a system, and based on them reason about the strategic power of the players in a game. Verification is extremely hard in Strategy logic too: it is $(d+1)$EXPTIME-complete, where d is the alternation between universal and existential strategy quantifiers in the logic, which has LTL as base language—a logic in the linear-time framework. Strategy logic as defined in (Chatterjee, Henzinger, and Piterman 2010) is a logic with a two-player semantic game. This logic was later on extended to the multi-player setting in (Mogavero, Murano, and Vardi 2010a). Recent works, for instance (Mogavero et al. 2012; Mogavero, Murano, and Sauro 2013), have focused on the discovery of syntactic fragments of Strategy logic which are expressive enough, *e.g.*, to express the existence of equilibria, and yet with decidable satisfiability and model checking problems. As with Equilibrium logics, the exact complexity upper bound of model checking a syntactic fragment that is able to express Nash equilibria is, presently, an open problem. Finally, in (Mogavero, Murano, and Vardi 2010b) a memoryfull extension of ATL, called mATL*, is studied. Again, this logic identifies the need for memoryfull power when reasoning about strategic interactions. Since mATL* extends ATL, it is an alternating-time temporal logic (Alur, Henzinger, and Kupferman 2002). This logic, as mCTL*, extends ATL* with a "present" proposition to differentiate between (and reason about) computations that start in the present state of play and computations that start far in the past—in a state previously visited. The verification problem for this logic, as it is for ATL*, is in 2EXPTIME.

Future work.
It would be interesting to discover *"easy" classes of reactive games* for which the decision problems studied in the paper are computationally simpler. Because of the nature of the problems at hand, all relevant questions about equilibria are expected to be at least NP-hard. Another avenue for further work is the study of solution concepts specifically designed for the analysis of dynamic systems, in particular, we would like to investigate the well-known concept of *subgame-perfect Nash equilibrium*. We also leave for future work the study of richer preference relations as well as quantitative and imperfect information. Certain decision problems slightly discussed in this paper should also be studied in more detail. In this paper, we have investigated the complexity of some questions about reactive games and equilibrium computations in multi-agent and concurrent systems, and would like to understand better the *expressivity of equililibrium logics*. They certainly offer a different paradigm for reasoning about equilibria in infinite games, but whether they can be translated into one of the logics already in the literature (or the other way around) is an open and interesting question. Finally, our results naturally lead to further algorithmic solutions and a tool implementation, for instance, as done using logics such as ATL and probabilistic variants of CTL* in model checkers such as PRISM (Chen et al. 2013), MCMAS (Lomuscio, Qu, and Raimondi 2009), or MOCHA (Alur et al. 1998).

Acknowledgments

We thank the anonymous reviewers for their helpful comments. We also acknowledge the support of the ERC Ad-

vanced Investigator Grant 291528 ("RACE").

References

Alur, R.; Henzinger, T. A.; Mang, F. Y. C.; Qadeer, S.; Rajamani, S. K.; and Tasiran, S. 1998. MOCHA: modularity in model checking. In *CAV*, volume 1427 of *LNCS*, 521–525. Springer.

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713.

Alur, R.; La Torre, S.; and Madhusudan, P. 2003. Playing games with boxes and diamonds. In *CONCUR*, volume 2761 of *LNCS*, 127–141. Springer.

Chatterjee, K.; Henzinger, T. A.; and Piterman, N. 2010. Strategy logic. *Information and Computation* 208(6):677–693.

Chen, T.; Forejt, V.; Kwiatkowska, M. Z.; Parker, D.; and Simaitis, A. 2013. PRISM-games: a model checker for stochastic multi-player games. In *TACAS*, volume 7795 of *LNCS*, 185–191. Springer.

Clarke, E. M., and Emerson, E. A. 1981. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Logics of Programs*, volume 131 of *LNCS*, 52–71. Springer-Verlag: Berlin, Germany.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 2000. *Model Checking*. The MIT Press: Cambridge, MA.

Emerson, E. A., and Halpern, J. Y. 1986. 'Sometimes' and 'not never' revisited: on branching time versus linear time temporal logic. *Journal of the ACM* 33(1):151–178.

Emerson, E. A. 1990. Temporal and modal logic. In *Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands. 996–1072.

Gutierrez, J.; Harrenstein, P.; and Wooldridge, M. 2013. Iterated boolean games. In *IJCAI*. IJCAI/AAAI Press.

Kozen, D. 1983. Results on the propositional mu-calculus. *Theoretical Computer Science* 27:333–354.

Kupferman, O., and Vardi, M. Y. 2006. Memoryful branching-time logic. In *LICS*, 265–274. IEEE Computer Society.

Kupferman, O.; Madhusudan, P.; Thiagarajan, P. S.; and Vardi, M. Y. 2000. Open systems in reactive environments: Control and synthesis. In *CONCUR*, volume 1877 of *LNCS*, 92–107. Springer.

Laroussinie, F.; Markey, N.; and Schnoebelen, P. 2002. Temporal logic with forgettable past. In *LICS*, 383–392. IEEE Computer Society.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2009. MCMAS: a model checker for the verification of multi-agent systems. In *CAV*, volume 5643 of *LNCS*, 682–688. Springer.

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. Y. 2012. What makes ATL* decidable? A decidable fragment of strategy logic. In *CONCUR*, volume 7454 of *LNCS*, 193–208. Springer.

Mogavero, F.; Murano, A.; and Sauro, L. 2013. On the boundary of behavioral strategies. In *LICS*, 263–272. IEEE Computer Society.

Mogavero, F.; Murano, A.; and Vardi, M. Y. 2010a. Reasoning about strategies. In *FSTTCS*, volume 8 of *LIPIcs*, 133–144. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Mogavero, F.; Murano, A.; and Vardi, M. Y. 2010b. Relentful strategic reasoning in alternating-time temporal logic. In *LPAR*, volume 6355 of *LNCS*, 371–386. Springer.

Nielsen, M., and Winskel, G. 1995. Models for concurrency. In *Handbook of Logic in Computer Science*. Oxford University Press. 1–148.

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. The MIT Press: Cambridge, MA.

Parikh, R. 1985. The logic of games and its applications. In *Topics in the Theory of Computation*. Elsevier.

Pnueli, A. 1977. The temporal logic of programs. In *FOCS*, 46–57. IEEE.

Vardi, M. Y. 1988. A temporal fixpoint calculus. In *POPL*, 250–259. ACM Press.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

A Temporal Logic of Strategic Knowledge

Xiaowei Huang and **Ron van der Meyden**
The University of New South Wales

Abstract

The paper presents an extension of temporal epistemic logic that adds "strategic" agents in a way that allows standard epistemic operators to capture what agents could deduce from knowledge of the strategies of some subset of the set of agents. A number of examples are presented to demonstrate the broad applicability of the framework, including reasoning about implementations of knowledge-based programs, game theoretic solution concepts and notions from computer security. It is shown that notions from several variants of alternating temporal epistemic logic can be expressed. The framework is shown to have a decidable model checking problem.

Introduction

The design and study of distributed and multi-agent systems typically has to deal with agents who have a choice of action to perform, and have individual and possibly conflicting goals. This leads agents to act strategically, attempting to select their actions over time so as to guarantee these goals even in the face of other agents' behaviour. The choice of actions generally needs to be done on the basis of *imperfect* information concerning the state of the system.

These concerns have motivated the development of a variety of modal logics that aim to capture the essential notions of such settings. One of the earliest, dating from the 1980's was multi-agent epistemic logic (Halpern and Moses 1990; Parikh and Ramanujam 1985), which introduces modal operators that deal with imperfect information by providing a way to state what agents *know*. Combining such constructs with temporal logic (Pnueli 1977) constructs gives temporal-epistemic logics, which support reasoning about how agents' knowledge changes over time. Temporal-epistemic logic is an area about which a significant amount is now understood (Fagin et al. 1995).

Logics dealing with reasoning about strategies, which started to be developed in the same period (Parikh 1983), had a slower initial start, but have in recent years become the focus of intense study (Pauly 2002; Horty. 2001; Alur, Henzinger, and Kupferman 2002). Many subtle issues that arise concerning what agents know in settings where multiple agents act strategically. In the process of understanding these issues, there has been a proliferation of modal logics. In particular, for dealing with epistemic reasoning in strategic settings, there are multiple proposals, with different semantic and syntactic bases, for how to capture reasoning about the availability to groups of agents of strategies for achieving particular goals (Jonker 2003; Schobbens 2004; van Otterloo and Jonker 2005; Jamroga 2003; Jamroga and Ågotnes 2007).

We argue in this paper that this proliferation is unnecessary, and that an appropriate application of temporal epistemic logic, with a minor innovation, already has the power to deal with many issues of concern when dealing with epistemic reasoning in strategic settings.

In particular, whereas logics in this area frequently leave strategies implicit in the semantic modelling, we propose to work in an instance of a standard semantic framework for temporal epistemic logic, but with strategies explicitly represented in the semantic model. In fact, we are not the first to have applied this instance of the standard temporal-epistemic model (Halpern and O'Neill 2008). Our main innovation is small but, we claim, powerful: we introduce new agent names that refer to the strategies being used by the main players, and allow these new agent names to be included in (otherwise standard) operators for group knowledge.

We argue that this gives a logical approach with broad applicability. In particular, it can express many of the subtly different notions that have been the subject of proposals for alternating temporal epistemic logics. We demonstrate this by results that show how such logics can be translated into our setting. We also present a number of other examples including reasoning about possible implementations of knowledge-based programs, game theoretic solution concepts, and issues of concern in computer security. Moreover, as we show, our approach retains from alternating temporal epistemic logic the desirable property that model checking is decidable, in the case of an imperfect recall semantics for knowledge. We show that it is in fact PSPACE-complete, no more than the complexity of model checking the temporal logic LTL, on which we build, although we have a much richer expressiveness.

The structure of the paper is as follows. We first recall some standard definitions from temporal epistemic logic. We then present a semantic model (also standard) for the environments in which agents choose their actions. Build-

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

ing on this model, we show how to construct a model for temporal epistemic logic that builds in information about strategic behaviour, but which introduces the new "strategy" agents. The following sections deal with applications of this model. We show successively that it can express reasoning about implementations of knowledge-based programs, many notions that have been proposed in the area of alternating temporal epistemic logic, game theoretic solution concepts, and problems from computer security. Next, we show that the framework has a decidable model checking problem. We conclude with a discussion of related literature.

Temporal Epistemic Logic

Suppose that we are working with a system in which there is a finite set Ags of agent names. Let $Prop$ be a set of atomic propositions. The language CTL*K($Ags, Prop$) has the syntax:

$$\phi \equiv p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid A\phi \mid \bigcirc\phi \mid \phi_1 U \phi_2 \mid D_G\phi \mid C_G\phi$$

where $p \in Prop$ and $G \subseteq Ags$. Intuitively, the formula $D_G\phi$ expresses that agents in G have distributed knowledge of ϕ, i.e., they could deduce ϕ if they pooled their information, and $C_G\phi$ says that ϕ is common knowledge to group G. The formulas $\bigcirc\phi$ and $\phi_1 U \phi_2$ express the linear time temporal logic notions "ϕ at the next moment of time" and "ϕ_1 until ϕ_2", respectively. The operator A refers to branching temporal structure: $A\phi$ says that the formula ϕ holds for all possible evolutions from the present situation. Other operators can be obtained in the usual way, e.g., $\phi_1 \wedge \phi_2 = \neg(\neg\phi_1 \vee \neg\phi_2)$, $\Diamond\phi = (trueU\phi)$, $\Box\phi = \neg\Diamond\neg\phi$, etc. For an agent $i \in Ags$ write $K_i\phi$ for $D_{\{i\}}\phi$, this expresses that agent i knows the fact ϕ. The notion of everyone in group G knowing ϕ can then be expressed as $E_G\phi = \bigwedge_{i\in G} K_i\phi$. Using this, common knowledge $C_G\phi$ can be understood as the conjunction of $E^k_G\phi$ for all $k \geq 0$.

Semantics of the language is given in *interpreted systems*, which are tuples $\mathcal{I} = \langle \mathcal{R}, \pi \rangle$, where $\mathcal{R}$ is a set, called the set of *runs*, and $\pi : \mathcal{R} \times \mathbb{N} \to \mathcal{P}(Prop)$ is a propositional interpretation. It is required that there exist for each $i \in Ags$, a nonempty set L_i of *local states of agent* i, such that $\mathcal{R}$ is a subset of the set of functions $r : \mathbb{N} \to \Pi_{i\in Ags} L_i$. Intuitively, this says that a run maps each moment of time to a *global state*, which gives a local state for each agent i. The usual presentation (Fagin et al. 1995), also includes a component in the global state for the state of the environment. We will handle this by including the special agent $e \in Ags$, representing the environment, when required. For $n \leq m$ write $r[n \dots m]$ for the sequence $r(n)r(n+1)\dots r(m)$. Elements of $\mathcal{R} \times \mathbb{N}$ are called the *points* of $\mathcal{I}$. For each agent $i \in Ags$, we write $r_i(m)$ for the i-component of $r(m)$, and define an equivalence relation on points by $(r, m) \sim_i (r', m')$ if $r_i(m) = r'_i(m')$. For $G \subseteq Ags$, we define the equivalence relation $\sim_G$ on points by $(r, m) \sim_G (r', m')$ if $(r, m) \sim_i (r', m')$ for all $i \in G$. Note that $\sim_\emptyset$ is then the universal relation on $\mathcal{R}\times\mathbb{N}$, intuitively representing an agent who knows nothing, and $\sim_{Ags}$ is an agent with complete information about the global state. In particular, it follows for runs $r, r' \in \mathcal{R}$ that if $(r, m) \sim_{Ags} (r', m)$ for all $m \in \mathbb{N}$, then $r = r'$.

The semantics of the language CTL*K is given by a relation $\mathcal{I}, (r, m) \models \phi$, representing that formula ϕ holds at point (r, m) of the interpreted system $\mathcal{I}$. This defined inductively on the structure of the formula ϕ, as follows:

- $\mathcal{I}, (r, m) \models p$ if $p \in \pi(r, m)$;
- $\mathcal{I}, (r, m) \models \neg\phi$ if not $\mathcal{I}, (r, m) \models \phi$;
- $\mathcal{I}, (r, m) \models \phi \wedge \psi$ if $\mathcal{I}, (r, m) \models \phi$ and $\mathcal{I}, (r, m) \models \psi$;
- $\mathcal{I}, (r, m) \models A\phi$ if $\mathcal{I}, (r', m) \models \phi$ for all $r' \in \mathcal{R}$ with $r[0 \dots m] = r'[0 \dots m]$;
- $\mathcal{I}, (r, m) \models \bigcirc\phi$ if $\mathcal{I}, (r, m+1) \models \phi$;
- $\mathcal{I}, (r, m) \models \phi U \psi$ if there exists $m' \geq m$ such that $\mathcal{I}, (r, m') \models \psi$ and $\mathcal{I}, (r, k) \models \phi$ for all k with $m \leq k < m'$.
- $\mathcal{I}, (r, m) \models D_G\phi$ if $\mathcal{I}, (r', m') \models \phi$ for all points (r', m') with $(r, m) \sim_G (r'm')$;
- $\mathcal{I}, (r, m) \models C_G\phi$ if $\mathcal{I}, (r', m') \models \phi$ for all points (r', m') with $(r, m) \sim^C_G (r'm')$, where $\sim^C_G = (\cup_{i\in G} \sim_i)^*$.

We write $\mathcal{I} \models \phi$ to mean that $\mathcal{I}, (r, 0) \models \phi$ for all runs of $\mathcal{I}$.

Strategic Environments

In order to deal with agents that operate in an environment by strategically choosing their actions, we introduce a type of transition system that models the available actions and their effects on the state. An *environment* for agents Ags is a tuple $E = \langle S, I, Acts, \to, \{O_i\}_{i\in Ags}, \pi \rangle$, where

1. S is a set of states,
2. I is a subset of S, representing the initial states,
3. $Acts = \Pi_{i\in Ags} Acts_i$ is a set of joint actions, where each $Acts_i$ is a nonempty set of actions that may be performed by agent i,
4. $\to \subseteq S \times Acts \times S$ is a transition relation, labelled by joint actions,
5. for each $i \in Ags$, component $O_i : S \to L_i$ is an observation function,
6. $\pi : S \to \mathcal{P}(Prop)$ is a propositional assignment.

Since later constructions add agent e, it is assumed here that $e \notin Ags$. An environment is said to be finite if all its components, i.e., $S, Ags, Acts_i, L_i$ and $Prop$ are finite. Intuitively, a joint action $a \in Acts$ represents a choice of action $a_i \in Acts_i$ for each agent $i \in Ags$, performed simultaneously, and the transition relation resolves this into an effect on the state. We assume that $\to$ is serial in the sense that for all $s \in S$ and $a \in Acts$ there exists $t \in S$ such that $(s, a, t) \in \to$.

A *strategy* for agent $i \in Ags$ in such an environment E is a function $\alpha : S \to \mathcal{P}(Acts_i) \setminus \{\emptyset\}$, selecting a set of actions of the agent at each state.[1] We call these the actions *enabled* at the state. A strategy is *deterministic* if $\alpha(s)$ is a singleton for all s.

A *strategy* for a group $G \subseteq Ags$ is a tuple $\alpha_G = \langle \alpha_i \rangle_{i\in G}$ such that α_i is a strategy for agent i, for each $i \in G$. A group

[1] More generally, one can take a strategy to be a function of the history. We focus in this paper on strategies that depend only on the final state.

strategy is *deterministic* if $\alpha_i(s)$ is a singleton for all states s and all $i \in G$. A strategy α_i for agent i is *uniform* if for all states s, t, if $O_i(s) = O_i(t)$, then $\alpha_i(s) = \alpha_i(t)$. A strategy $\alpha_G = \langle \alpha_i \rangle_{i \in G}$ for a group G is *locally uniform (deterministic)* if α_i is uniform (respectively, deterministic) for each agent $i \in G$.[2]

Given an environment E, we write $\Sigma(E)$ for the set of all joint strategies for the group Ags of all agents, $\Sigma^{det}(E)$ for the set of deterministic strategies, $\Sigma^{unif}(E)$ for the set of all locally uniform joint strategies, and $\Sigma^{unif,det}(E)$ for the set of all deterministic locally uniform joint strategies.

We now introduce a notation for referring to an agent's strategy, that enables us to treat the strategy as itself an agent. Given an agent i, we write $\sigma(i)$ for a new agent that represents agent i's strategy. For a set $G \subseteq Ags$ of agents, we define $\sigma(G) = \{\sigma(i) \mid i \in G\}$. We note that agent $\sigma(i)$ will typically not be associated with any actions, it is intended primarily for use in epistemic operators in the semantics of an interpreted system constructed to allow reference to what can be deduced were agents to reason using information about each other's strategies.

Given an environment $E = \langle S, I, Acts, \rightarrow, \{O_i\}_{i \in Ags}, \pi \rangle$ for agents Ags, where $O_i : S \rightarrow L_i$ for each $i \in Ags$, and a set $\Sigma \subseteq \Pi_{i \in Ags} \Sigma_i$ of joint strategies for the group Ags, we define the *strategy space* interpreted system $\mathcal{I}(E, \Sigma) = (\mathcal{R}, \pi')$ for agents $Ags \cup \sigma(Ags) \cup \{e\}$. The system $\mathcal{I}(E, \Sigma)$ has global states $\mathcal{G} = S \times \Pi_{i \in Ags} L_i \times \Pi_{i \in Ags} \Sigma_i$. Intuitively, each global state consists of a state of the environment E, a local state for each agent i in E, and a strategy for each agent i. We index the components of this cartesian product by e, the elements of Ags and the elements of $\sigma(Ags)$, respectively. We take the set of runs $\mathcal{R}$ of $\mathcal{I}(E, \Sigma)$ to be the set of all runs $r : \mathbb{N} \rightarrow \mathcal{G}$ satisfying the following constraints, for all $m \in \mathbb{N}$ and $i \in Ags$:

1. $r_e(0) \in I$ and $\langle r_{\sigma(i)}(0) \rangle_{i \in Ags} \in \Sigma$,
2. $r_i(m) = O_i(r_e(m))$,
3. $(r_e(m), a, r_e(m+1)) \in \rightarrow$ for some joint action $a \in Acts$ such that for all $j \in Ags$ we have $a_j \in \alpha_j(r_j(m))$, where $\alpha_j = r_{\sigma(j)}(m)$, and
4. $r_{\sigma(i)}(m+1) = r_{\sigma(i)}(m)$.

The first constraint, intuitively, says that runs start at an initial state of E, and the initial strategy profile at time 0 is one of the profiles in Σ. The second constraint states that the agent i's local state at time m is the observation that agent i makes of the state of the environment at time m. The third constraint says that evolution of the state of the environment is determined at each moment of time by agents choosing an action by applying their strategy at that time to their local state at that time. The joint action resulting from these individual choices is then resolved into a transition on the state of the environment using the transition relation from E. The fourth constraint says that agents' strategies are fixed during the course of a run. Intuitively, each agent picks a strategy, and then sticks to it. The interpretation π' of $\mathcal{I}(E, \Sigma)$ is determined from the interpretation π of E by taking $\pi'(r, m) = \pi(r_e(m))$ for all points (r, m).

Our epistemic strategy logic is now just an instantiation of the extended temporal epistemic logic in the strategy space generated by an environment. That is, we start with an environment E and an associated set of strategies Σ, and then work with the language $\mathrm{CTL}^*\mathrm{K}(Ags \cup \sigma(Ags) \cup \{e\}, Prop)$ in the interpreted system $\mathcal{I}(E, \Sigma)$. We call this instance of the language $\mathrm{CTL}^*\mathrm{K}(Ags, Prop)$, or just $\mathrm{CTL}^*\mathrm{K}$ when the parameters are implicit.

The key point is that the system $\mathcal{I}(E, \Sigma)$ represents the possible temporal evolutions under *all* possible choices of joint strategies from Σ by the agents, and provides a way to refer, using distributed knowledge operators D_G where G contains the new strategic agents $\sigma(i)$, to what agents would know, should they take into account not just their own observations, but also information about other agent's strategies. For example, the distributed knowledge operator $D_{\{i,\sigma(i)\}}$ captures the knowledge that agent i has, taking into account the strategy that it is running. $D_{\{i,\sigma(i),\sigma(j)\}}$ captures what agent i would know, taking into account its own strategy and the strategy being used by agent j. Various applications of the usefulness of this expressiveness are given in the following sections.

In general, we will be most interested in the case where $\Sigma = \Sigma^{unif}(E)$ is the set of all locally uniform strategies in E, so we use the abbreviation $\mathcal{I}^{unif}(E)$ for $\mathcal{I}(E, \Sigma^{unif}(E))$, and similarly for the other strategy sets that we defined above.

We remark that the agent e has perfect information about the state of the environment, including the agent's observation, so $D_{\{e\} \cup G}\phi$, for $G \subseteq Ags$ is equivalent to $D_{\{e\}}\phi$. Thus, the operator $D_{\{e\} \cup \sigma(G)}$ captures what can be deduced from knowledge of the state of the environment and the strategies of agents in G.

Reasoning about Knowledge-Based Programs

Knowledge-based programs (Fagin et al. 1997) are a form of specification of a multi-agent system in the form of a program structure that specifies how an agent's actions are related to its knowledge. They have been shown to be a useful abstraction for several areas of application, including the development of optimal protocols for distributed systems (Fagin et al. 1997), robot motion planning (Brafman et al. 1997), and game theoretic reasoning (Halpern and Moses 2007).

Knowledge-based programs cannot be directly executed, since there is a circularity in their semantics: which actions are performed depends on what the agents know, which in turn depends on which actions the agents perform. The circularity is not vicious, and can be resolved by means of a fixed point semantics, but it means that knowledge-based programs may have multiple distinct implementations (or none), and the problem of reasoning about these implementations is quite subtle. In this section, we show that our framework can capture reasoning about the set of possible knowledge-based program implementations.

[2] We prefer the term "locally uniform" to just "uniform" in the case of groups, since we could say a strategy α for group G is *globally uniform* if for all states s, t, if $O_i(s) = O_i(t)$ for all $i \in G$, then $\alpha_i(s) = \alpha_i(t)$ for all $i \in G$. While we do not pursue this in the present paper, this notion would be interesting in settings where agents share information to collude on their choice of move.

We consider joint knowledge-based programs P (as defined by (Fagin et al. 1997)) where for each agent i we have a knowledge-based program

$$P_i = \textbf{do } \phi^i_1 \rightarrow a^i_1 \ [] \ \ldots [] \ \phi^i_{n_i} \rightarrow a^i_{n_i} \textbf{ od}$$

where each ϕ^i_j is a formula of CTL*K($Ags, Prop$) of the form $K_i\psi$, and each a_i appears just once.[3] Intuitively, this program says to repeat forever the following operation: nondeterministically execute one of the actions a^i_j such that the corresponding guard ϕ^i_j is true.

We present a formulation of semantics for knowledge-based programs that refactors the definitions of (Fagin et al. 1997), following the approach of (van der Meyden 1996) which uses the notion of environment defined above rather than the original notion of *context*. A *potential implementation* of a knowledge-based program P in an environment E is a joint strategy α in E. Given a potential implementation α in E, we can construct the interpreted system $\mathcal{I}(E, \{\alpha\})$, which captures the possible runs of E when the agents choose their actions according to the single possible joint strategy α. Given this interpreted system, we can now interpret the epistemic guards in P. Say that a state s of E is α-reachable if there is a point (r, m) of $\mathcal{I}(E, \{\alpha\})$ with $r_e(m) = s$. We note that for a formula $K_i\phi$, and a point (r, m) of $\mathcal{I}(E, \{\alpha\})$, the statement $\mathcal{I}(E, \{\alpha\}), (r, m) \models K_i\phi$ depends only on the state $r_e(m)$ of the environment at (r, m). For an α-reachable state s of E, it therefore makes sense to define satisfaction of $K_i\phi$ at s rather than at a point, by $\mathcal{I}(E, \{\alpha\}), s \models K_i\phi$ if $\mathcal{I}(E, \{\alpha\}), (r, m) \models K_i\phi$ for all (r, m) with $r_e(m) = s$. We define α to be an *implementation* of P in E if for all α-reachable states s of E and agents i, we have

$$\alpha_i(s) = \{a^i_j \mid 1 \leq j \leq n_i, \ \ \mathcal{I}(E, \{\alpha\}), s \models K_i\phi^i_j\} \ .$$

Intuitively, the right hand side of this equation is the set of actions that are enabled at s by P_i when the tests for knowledge are interpreted using the system obtained by running the strategy α itself. The condition states that the strategy is an implementation if it enables precisely this set of actions at every reachable state.

We now show that our framework for strategic reasoning can express the same content as a knowledge-based program by means of a formula, and that this enables the framework to be used for reasoning about knowledge-based program implementations.

We need one constraint on the environment. Say that an environment E is *action-recording* if for each $a \in Acts_i$ there exists an atomic proposition $did_i(a)$ such that for $s \in I$ we have $did_i(a) \notin \pi(s)$ and for all states s, t and joint actions a such that $(s, a, t) \in \rightarrow$, we have $did_i(b) \in \pi(t)$ iff $b = a_i$, for all agents i. It is easily seen that any environment can be made action-recording, just by adding a component to the states that records the latest joint action.

We can now express knowledge-based program implementations as follows. For a formula ψ, write $\psi^\$$ for the result of substituting for each occurrence of an operator D_G in ψ the operator $D_{G\cup\sigma(Ags)}$. Intuitively, this substitution says that the operator D_G in ψ is to be interpreted as if it is known that the current joint strategy is being played. To understand the motivation for this, note that in $\mathcal{I}(E, \{\alpha\})$ it is common knowledge that the joint strategy α is being played. Let

$$\textbf{imp}(P) = D_{\sigma(Ags)}(\bigwedge_{i\in Ags, j=1\ldots n_i} ((\phi^i_j)^\$ \Leftrightarrow EXdid_i(a^i_j))).$$

Intuitively, this formula says that the current joint strategy gives an implementation of the knowledge-based program P. More precisely, we have the following:

Proposition 1 *Suppose that P is a knowledge-based program that does not make use of common knowledge operators. Let α be a locally uniform joint strategy in E and let r be a run of $\mathcal{I}^{unif}(E)$, in which the agents are running joint strategy α, i.e., where $r_{\sigma(i)}(0) = \alpha_i$ for all $i \in Ags$. Let $m \in \mathbb{N}$. Then the strategy α is an implementation of knowledge-based program P in E iff $\mathcal{I}^{unif}(E), (r, m) \models \textbf{imp}(P)$.*

Note that here we check $\textbf{imp}(P)$ in the system $\mathcal{I}^{unif}(E)$ that contains *all* joint strategies, not just a single strategy α, as in the definition of implementation for a knowledge-based program. The point of this is that *a priori*, we do not have an implementation at hand.

In particular, as a consequence of this result, it follows that several properties of knowledge-based programs (that do not make use of common knowledge operators) can be expressed in the system $\mathcal{I}^{unif}(E)$:

1. The statement that there exists an implementation of P in E can be expressed by
$$\mathcal{I}^{unif}(E) \models \neg D_\emptyset \neg \textbf{imp}(P)$$
2. The statement that all implementations of P in E guarantee that formula ϕ holds at all times can be expressed by
$$\mathcal{I}^{unif}(E) \models D_\emptyset(\textbf{imp}(P) \Rightarrow D_{\sigma(Ags)}\phi^\$))$$

We remark that as a consequence of these encodings and the fact that the model checking problem described below is in PSPACE, we obtain that testing for the existence of an implementation of a knowledge-based program is in PSPACE, as is the problem of determining whether all implementations satisfy some formula. For testing existence, this result was known (Fagin et al. 1997), but the result on verification has not previously been noted (though it could also have been shown using the techniques in (Fagin et al. 1997).)

We remark that the reason for excluding common knowledge $C_G\phi$ from the conditions in the knowledge-based program in Proposition 1 is that this would require us to express a greatest fixedpoint X of the equation $X \equiv \bigwedge_{i\in G} D_{\{i\}\cup\sigma(Ags)}(X \wedge \phi)$, which is not the same as $C_{G\cup\sigma(Ags)}\phi$. One appropriate way to handle this in our framework would be to work with a mu-calculus extension. We do not pursue this here.

[3] The guards in (Fagin et al. 1997) are allowed to be boolean combinations of formulas $K_i\psi$ and propositions p local to the agent: since for such propositions $p \Leftrightarrow K_ip$, and the operator K_i satisfies positive and negative introspection, our form for the guards is equally general. They do not require that a_i appears just once, but the program can always be put into this form by aggregating clauses for a_i into one and taking the disjunction of the guards.

Connections to variants of ATEL

Alternating temporal logic (ATL) (Alur, Henzinger, and Kupferman 2002) is a generalization of the branching time temporal logic CTL that can express the capability of agent strategies to bring about temporal effects. Essentially, each branching construct $A\phi$ is generalized to an *alternating* construct $\langle\!\langle G\rangle\!\rangle\phi$ for a group G of agents, where ϕ is a "prefix temporal" formula such as $X\phi'$, $F\phi'$, $G\phi'$ or $\phi_1 U \phi_2$, as would be used to construct a CTL operator. Intuitively, $\langle\!\langle G\rangle\!\rangle\phi$ says that the group G has a strategy for ensuring that ϕ holds, irrespective of what the other agents do.

Alternating temporal epistemic logic (ATEL), adds epistemic operators to ATL (van der Hoek and Wooldridge 2002). As a number of subtleties arise in the formulation of such logics, several variants of ATEL have since been developed. In this section, we consider a number of such variants and argue that our framework is able to express the main strategic concepts from these variants. We begin by recalling ATEL as defined in (van der Hoek and Wooldridge 2002). Various modellings of the environments in which agents operate have been used in the literature; we base our modelling on the notion of environment introduced above.

The syntax of ATEL is given as follows:

$$\phi \equiv p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \langle\!\langle G\rangle\!\rangle \bigcirc \phi \mid \langle\!\langle G\rangle\!\rangle \Box \phi \mid \langle\!\langle G\rangle\!\rangle (\phi_1 U \phi_2) \mid K_i\phi \mid C_G\phi$$

where $p \in Prop$, $i \in Ags$ and $G \subseteq Ags$. The intuitive meaning of the constructs is as in CTL*K above, with additionally $\langle\!\langle G\rangle\!\rangle\phi$ having the intuitive reading that group G has a strategy for assuring that ϕ holds. The semantics is given by a relation $E, s \models^{\Sigma} \phi$, where $E = \langle S, I, Acts, \rightarrow, \{O_i\}_{i\in Ags}, \pi\rangle$ is an environment, $s \in S$ is a state of E, and ϕ is a formula. For reasons discussed below, we parameterize the definition on a set Σ of strategies for groups of agents in the environment E. For the definition, we need the notion of a path in E: this is a function $\rho : \mathbb{N} \rightarrow S$ such that for all $k \in \mathbb{N}$ there exists a joint action a with $(\rho(k), a, \rho(k+1)) \in \rightarrow$. A path ρ is *from* a state s if $\rho(0) = s$. A path ρ is *consistent* with a strategy α for a group G if for all $k \in \mathbb{N}$ there exists a joint action a such that $(\rho(k), a, \rho(k+1)) \in \rightarrow$ and $a_i \in \alpha_i(\rho(k))$ for all $i \in G$.

The relation $E, s \models^{\Sigma} \phi$ is defined inductively on the structure of the formula ϕ:

- $E, s \models^{\Sigma} p$ if $p \in \pi(s)$;
- $E, s \models^{\Sigma} \neg\phi$ if not $E, s \models^{\Sigma} \phi$;
- $E, s \models^{\Sigma} \phi \wedge \psi$ if $E, s \models^{\Sigma} \phi$ and $E, s \models^{\Sigma} \psi$;
- $E, s \models^{\Sigma} \langle\!\langle G\rangle\!\rangle \bigcirc \phi$ if there exists a strategy $\alpha_G \in \Sigma$ for group G such that for all paths ρ from s that are consistent with α_G, we have $E, \rho(1) \models^{\Sigma} \phi$;
- $E, s \models^{\Sigma} \langle\!\langle G\rangle\!\rangle \Box \phi$ if there exists a strategy $\alpha_G \in \Sigma$ for group G such that for all paths ρ from s that are consistent with α_G, we have $E, \rho(k) \models^{\Sigma} \phi$ for all $k \in \mathbb{N}$;
- $E, s \models^{\Sigma} \langle\!\langle G\rangle\!\rangle (\phi U \psi)$ if there exists a strategy $\alpha_G \in \Sigma$ for group G such that for all paths ρ from s that are consistent with α_G, there exists $m \geq 0$ such that $E, \rho(m) \models^{\Sigma} \psi$, and for all $k < m$, we have $E, \rho(k) \models^{\Sigma} \phi$.
- $E, s \models^{\Sigma} K_i\phi$ if $E, t \models^{\Sigma} \phi$ for for all $t \in S$ with $t \sim_i s$;
- $E, s \models^{\Sigma} C_G\phi$ if $E, t \models^{\Sigma} \phi$ for for all $t \in S$ with $(s, t) \in (\cup_{i\in G} \sim_i)^*$;

The specific version of ATEL defined in (van der Hoek and Wooldridge 2002) is obtained from the above definitions by taking $\Sigma = \Sigma^{det} = \{\sigma_G \mid G \subseteq Ags,\ \sigma_G$ a deterministic G-strategy in $E\}$. That is, following the definitions for ATL, this version works with arbitrary deterministic group strategies, in which an agent selects its action as if it has full information of the state. This aspect of the definition has been critiqued by Jonker (Jonker 2003) and (in the case of ATL without epistemic operators) by Schobbens (Schobbens 2004), who argue that this choice is not in the spirit of the epistemic extension, in which observations are intended precisely to represent that agents do not have full information of the state. They propose that the definition instead be based on the set $\Sigma^{\text{det},unif} = \{\sigma_G \mid G \subseteq Ags,\ \sigma_G$ a locally uniform deterministic G-strategy in $E\}$. This ensures that in choosing an action, agents are able to use only the information available in their observations.

We concur that the use of locally uniform strategies is the more appropriate choice, but in either event, we now argue that our approach using strategy space is able to express everything that can be expressed in ATEL. Consider the following translation from ATEL to CTL*K($Prop, Ags^+ \cup \sigma(Ags) \cup \{e\}$). For a formula ϕ, we write ϕ^* for the translation of ϕ, defined inductively on the construction of ϕ by the following rules

$$p^* = p \qquad (\neg\phi)^* = \neg\phi^* \qquad (\phi_1 \wedge \phi_2)^* = \phi_1^* \wedge \phi_2^*$$
$$(K_i\phi)^* = K_i\phi^* \qquad (C_G\phi)^* = C_G\phi^*$$
$$(\langle\!\langle G\rangle\!\rangle \bigcirc \phi)^* = \neg K_e \neg D_{\{e\}\cup\sigma(G)} \bigcirc \phi^*$$
$$(\langle\!\langle G\rangle\!\rangle \Box\phi)^* = \neg K_e \neg D_{\{e\}\cup\sigma(G)} \Box\phi^*$$
$$(\langle\!\langle G\rangle\!\rangle \phi_1 U \phi_2)^* = \neg K_e \neg D_{\{e\}\cup\sigma(G)} (\phi_1^* U \phi_2^*)$$

Given a strategy $\alpha = \langle\alpha_i\rangle_{i\in G}$ for a group of agents G in an environment E, define the *completion* of the strategy to be the joint strategy $comp(\alpha) = \langle\alpha'_i\rangle_{i\in G}$ with $\alpha'_i = \alpha_i$ for $i \in G$ and with $\alpha_i(s) = Acts_i$ for all $i \in Ags \setminus G$ and $s \in S$. Intuitively, this operation completes the group strategy to a joint strategy for all agents, by adding the random strategy for all agents not in G. Given a set of strategies Σ for groups of agents, we define $comp(\Sigma) = \{comp(\alpha) \mid \alpha \in \Sigma\}$. Say that a set Σ of group strategies is *restrictable* if for every $\alpha \in \Sigma$ for group of agents G and every group $G' \subseteq G$, the restriction $\alpha_{G'}$ of α to agents in G' is also in Σ. Say that Σ is *extendable* if for every strategy for a group $G' \subseteq G$, there exists a strategy $\alpha' \in \Sigma$ for group G whose restriction $\alpha'_{G'}$ to G' is equal to α. For example, the set of all group strategies, and the set of all locally uniform group strategies, are both restrictable and extendable.

For an environment E, write $E[S/I]$ for the environment obtained by making all states in E be initial, i.e., replacing the set of initial states I of E by the set of all states S of E. (This is a technical transformation that we require because E may have temporally unreachable states that would not occur in an interpreted system constructed from E, but that can be accessed via an equivalence relation $\sim_i$ in the semantics of ATEL.) We can then show the following.

Theorem 1 *For every environment E, nonempty set of group strategies Σ that is restrictable and extendable, for every state s of E and ATEL formula ϕ, we have $E, s \models^{\Sigma} \phi$ iff for all (equivalently, some) points (r, m) of $\mathcal{I}(E[S/I], comp(\Sigma))$ with $r_e(m) = s$ we have $\mathcal{I}(E[S/I], comp(\Sigma)), (r, m) \models \phi^*$.*

Similar translation results can be given for other alternating temporal epistemic logics from the literature. We sketch a few of these translations here.

Jamroga and van der Hoek (Jamroga and van der Hoek 2004) note that ATEL admits situations consisting of an environment E and a state s where $E, s \models K_i\langle\langle i\rangle\rangle\phi$, i.e., in every state consistent with agent i's knowledge, some strategy for agent i is guaranteed to satisfy ϕ, but still there is no strategy for agent i that agent i knows will work to achieve ϕ. They formulate a construct $\langle\langle G\rangle\rangle^{\bullet}_{\mathcal{K}(H)}\phi$ that says, effectively, that there is a strategy for a group G that another group H knows (for notion of group knowledge $\mathcal{K}$, which could be E for everyone knows, D for distributed knowledge, or C for common knowledge) to achieve goal ϕ. More precisely,

> $E, s \models \langle\langle G\rangle\rangle^{\bullet}_{\mathcal{K}(H)}\phi$ if there exists a uniform strategy α for group G such that for all states t with $s \sim^{\mathcal{K}}_{H} t$, we have that all paths ρ from t that are consistent with α satisfy ϕ.

Here $\sim^{\mathcal{K}}_{H}$ is the appropriate epistemic indistinguishability relation on states of E. The particular case $\langle\langle G\rangle\rangle^{\bullet}_{E(G)}\phi$ is also proposed as the semantics for the ATL construct $\langle\langle G\rangle\rangle\phi$ in (Schobbens 2004; Jonker 2003; Jamroga and Ågotnes 2007).

The construct $\langle\langle G\rangle\rangle^{\bullet}_{D(H)}\phi$ can be represented in our language as

$$\neg D_e \neg D_{H\cup\sigma(G)}\phi \ .$$

Intuitively, here the first modal operator $\neg D_e\neg$ switches the strategy of all the agents while maintaining the state s, thereby selecting a strategy α for group G in particular, and the next operator $D_{\{H,\sigma(G)\}}$ verifies that the group H knows that the strategy G being used by group G guarantees ϕ. Similarly, $\langle\langle G\rangle\rangle^{\bullet}_{E(H)}\phi$ can be represented as

$$\neg D_e \neg \bigwedge_{i\in H} D_{\{i\}\cup\sigma(G)}\phi \ .$$

This gives a reduction of these complex operators of (Jamroga and van der Hoek 2004) to a set of standard epistemic operators.[4] (An alternate approach to decomposing the operators $\langle\langle G\rangle\rangle^{\bullet}_{\mathcal{K}(H)}$ is proposed in (Jamroga and Ågotnes 2007). By comparison with our standard semantics, this proposal uses "constructive knowledge" operators which require a nonstandard semantics in which formulas are evaluated at sets of states rather than at individual states.)

(W. van der Hoek, Jamroga, and Wooldridge 2005) introduce constants that refer to strategies, and adds to ATL a new (counterfactual) modality $C_i(c, \phi)$, with the intended reading "if it were the case that agent i committed to the strategy denoted by c, then ϕ". The formula ϕ here is not permitted to contain further references to agent i strategies. To interpret the formula $C_i(c, \phi)$ in an environment E, the environment is first updated to a new environment E' by removing all transitions that are inconsistent with agent i running the strategy referred to by c, and then the formula ϕ is evaluated in E'. After introducing propositional constants $p_{i,c}$ that say that agent i is running the strategy referred to by c, the formula $C_i(c, \phi)$ could be expressed in our framework as $D_{\{e\}\cup\sigma(Ags\setminus\{i\})}(p_{i,c} \Rightarrow \phi^{+\sigma(i)})$ where in the translation $\phi^{+\sigma(i)}$ of ϕ we ensure that there is no further deviation from the strategy of agent i by adding $\sigma(i)$ to the group of every knowledge operator occurring later in the translation.

(W. van der Hoek, Jamroga, and Wooldridge 2005) do not include epistemic operators. Were they to be added in the obvious way, the effect would be to make the meaning of the knowledge operator (with respect to the agent's knowledge concerning the strategies in play) dependent on the context in the formula. In particular, in $(K_j\phi) \wedge C_i(c, K_j\psi)$, the first occurrence of K_j would refer to an agent j who does not know that i is playing c, whereas the second K_j would effectively refer to our $K_{\{j,\sigma(i)\}}$, i.e., j's knowledge taking into account that i is playing c. Our framework is more expressive in that we can continue to use the former meaning even in the second context.

Solution Concepts

It has been shown for a number of logics for strategic reasoning that they are expressive enough to state a variety of game theoretic solution concepts, e.g., (W. van der Hoek, Jamroga, and Wooldridge 2005; Chatterjee, Henzinger, and Piterman 2010) show that Nash Equilibrium is expressible. We now sketch the main ideas required to show that our framework also has this expressive power. We assume two players $Ags = \{0, 1\}$ in a normal form game, and assume that these agents play a deterministic strategy. The results in this section can be easily generalized to multiple players and extensive form games.

Given a game $\mathcal{G}$ we construct an environment $E_{\mathcal{G}}$ that represents the game. Each player has a set of actions that correspond to the moves that the player can make. We assume that $E_{\mathcal{G}}$ is constructed to model the game so that play happens in the first step from a unique initial state, and that subsequent transitions do not change the state.

Let u_i for $i \in \{0, 1\}$ be a variable denoting the utility gained by player i when play is finished. Let V_i be the set of possible values for u_i. We write $-i$ to denote the adversary of player i. We use formula

$$U_i(v) = \bigcirc(u_i = v)$$

to express that value v is player i's utility once play finishes.

Nash equilibrium (NE) is a solution concept that states no player can gain by unilaterally changing their strategy. We may write

$$BR_i(v) = U_i(v) \wedge K_{\sigma(-i)} \bigwedge_{v'\in V_i} (U_i(v') \Rightarrow v' \leq v)$$

to express that, given the current adversary strategy, the value v attained by player i's current strategy is the best possible utility attainable by player i, i.e., the present strategy of

[4] As with knowledge-based programs above, to handle common knowledge, it seems we require an extension to mu-calculus.

player i is a best response to the adversary. Thus

$$BR_i = \bigvee_{v \in V_i} BR_i(v)$$

says that player i is playing a best-response to the adversary's strategy. The following statement then expresses the existence of a Nash equilibrium for the game $\mathcal{G}$:

$$\mathcal{I}^{unif,det}(E_{\mathcal{G}}) \models \neg D_{\emptyset} \neg (BR_0 \wedge BR_1) \; .$$

That is, in a Nash equilibrium, each player is playing a best response to the other's strategy.

Perfect cooperative equilibrium (PCE) is a solution concept intended to overcome deficiencies of Nash equilibrium for explaining cooperative behaviour (Halpern and Rong 2010). It says that each player does at least as well as she would if the other player were best-responding. The following formula

$$BU_i(v) = D_{\emptyset}(\bigwedge_{v' \in V_i} ((BR_{-i} \wedge U_i(v')) \Rightarrow v' \leq v))$$

states that v is as good as any utility that i can obtain if the adversary always best-responds to whatever i plays. Thus,

$$BU_i = \bigvee_{v \in V_i} (U_i(v) \wedge BU_i(v))$$

says that i is currently getting a utility as good the best utility that i can obtain if the adversary is a best-responder. Now, the following formula expresses the existence of perfect cooperative equilibrium for the game $\mathcal{G}$:

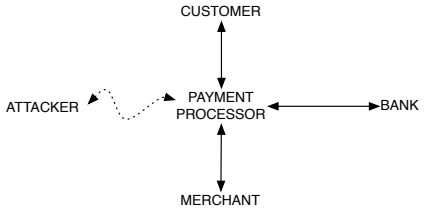
$$\mathcal{I}^{unif,det}(E_{\mathcal{G}}) \models \neg D_{\emptyset} \neg (BU_0 \wedge BU_1)$$

That is, in a PCE, no player has an incentive to change their strategy, on the assumption that the adversary will best-respond to any change.

Computer Security Example: Erasure policies

Formal definitions of computer security frequently involve reference to the strategies available to the players, and to agent's reasoning based on these strategies. In this section we sketch an example that illustrates how our framework might be applied in this context.

Consider the scenario depicted in the following diagram:



A customer C can purchase items at a web merchant M. Payment is handled by a trusted payment processor P (this could be a service or device), which interacts with the customer, merchant, and a bank B to securely process the payment. (To keep the example simple, we suppose that the customer and merchant use the same bank). One of the guarantees provided by the payment processor is to protect the customer from attacks on the customer's credit card by the merchant: the specification for the protocol that runs the transaction requires that the merchant should not obtain the customer's credit card number. In fact, the specification for the payment processor is that after the transaction has been successfully completed, the payment processor should *erase* the credit card data, to ensure that even the payment processor's state does not contain information about the customer's credit card number. The purpose of this constraint is to protect the customer against subsequent attacks by an attacker A, who may be able to use vulnerabilities in the payment processor's software to obtain access to the payment processor's state.

We sketch how one might use our framework to express the specification. To capture reasoning about all possible behaviours of the agents, and what they can deduce from knowledge of those behaviours, we work in $\mathcal{I}^{unif}(E)$ for a suitably defined environment E. To simplify matters, we take $Ags = \{C, M, P, A\}$. We exclude the strategy of the bank from consideration: this amounts to assuming that the bank has no actions and is trusted to run a fixed protocol. We similarly assume that the payment processor P has no actions, but in order to talk about what information is encoded in the payment processor's local state, we do allow that this agent has observations. The customer C may have actions such as entering the credit card number in a web form, pressing a button to submit the form to the payment processor, and pressing a button to approve or cancel the transaction. The customer observes variable `cc`, which records the credit card number drawn from a set `CCN`, and boolean variable `done` which records whether the transaction is complete (which could mean either committed or aborted).

We assume that the attacker A has some set of exploit actions, as well as some innocuous actions (e.g., setting a local variable or performing *skip*). The effect of the exploit actions is to exploit a vulnerability in the payment processor's software and copy parts of the local state of the payment processor to variables that are observable by the attacker. We include in the environment state a boolean variable `exploited`, which records whether the attacker has executed an exploit action at some time in the past. The merchant M may have actions such as sending cost information to the payment processor and acknowledging a receipt certifying that payment has been approved by the bank (we suppose this receipt is transmitted from the bank to the merchant via the payment processor).

We may then capture the statement that the system is *potentially* vulnerable to an attack that exploits an erasure flaw in the implementation of the payment processor, by the following formula:

$$\neg D_{\emptyset} \neg (\mathtt{done} \wedge \bigvee_{x \in \mathtt{CCN}} K_P(\mathtt{cc} \neq x)))$$

This says that there exist behaviours of the agents, which can (at least on some points in some runs) leave the payment processor in a state where the customer has received confirmation that the transaction is done, but in which the payment

processor's local state somehow still encodes *some* information about the customer's credit card number. This encoding could be direct (e.g., by having a variable *customer_cc* that still stores the credit card number) or indirect (e.g. by the local state including both a symmetric encryption key K and an encrypted version of the credit card number, *enc_customer_cc*, with value $\texttt{Encrypt}_K(\texttt{cc})$ that was used for secure transmission to the bank). Note that for a breach of security, it is only required that the information suffices to *rule out* some credit card number (so that, e.g., knowing the first digit of the number would constitute a vulnerability)

The vulnerability captured by this formula is only potential, because it does not necessarily follow that the attacker is able to obtain the credit card information. Whether this is possible can be checked using the formula

$$\neg D_{\emptyset}\neg(\texttt{done} \wedge \neg\texttt{exploited} \wedge EF \bigvee_{x\in\texttt{CCN}} D_{\{A,\sigma(A)\}}(\texttt{cc} \neq x)))$$

which says that it is possible for the attacker to obtain information about the credit card number even after the transaction is done. (To focus on erasure flaws, we deliberately wish to exclude here the possibility that the attack occurs during the processing of the transaction.) Note that here we assume that the attacker knows their own strategy when making deductions from the information obtained in the attack. This is necessary, because the attacker can typically write its own local variables, so it needs to be able to distinguish between a value it wrote itself and a value it copied from the payment processor.

However, even this formula may not be sufficiently strong. Suppose that the payment processor implements erasure by writing a random value to its variable *customer_cc*. Then, even if the attacker obtains a copy of this value, and it happens to be equal to the customer's actual credit card number, the attacker would not have any knowledge about the credit card number, since, as far as the attacker knows, it could be looking at a randomly assigned number. However, there may still be vulnerabilities in the system. Suppose that the implementation of the payment processor operates so that the customer's credit card data is not erased by randomization until the merchant has acknowledged the receipt of payment from the bank, but to avoid annoying the customer with a hanging transaction, the customer is advised that the transaction is approved (setting done true) if the merchant does not respond within a certain time limit. It is still the case that on observing the copied value of *customer_cc*, the attacker would not be able to deduce that this is the customer's credit card number, since it might be the result of erasure in the case that the merchant responded promptly. However, if the attacker knows that the merchant has not acknowledged the receipt, the attacker can then deduce that the value is not due to erasure. One way in which the attacker might know that the merchant has not acknowledged receipt is that the attacker is in collusion with the merchant, who has agreed to omit sending the required acknowledgement messages.

This type of attack can be captured by replacing the term $D_{\{A,\sigma(A)\}}(\texttt{cc} \neq x)$ by $D_{\{A,\sigma(A),\sigma(M)\}}(\texttt{cc} \neq x)$, capturing that the attacker reasons using knowledge of both its own strategy as well as the strategy of the merchant, or even $D_{\{A,\sigma(A),\sigma(M),M\}}(\texttt{cc} \neq x)$ for a collusion in which the merchant shares information observed. Similarly, to focus on erasure flaws in the implementation of the payment gateway, independently of the attackers capability, we would replace the term $K_P(\texttt{cc} \neq x)$ above by $D_{\{P,\sigma(M)\}}(\texttt{cc} \neq x)$.

We remark that in the case of the attacker's knowledge, it would be appropriate to work with a perfect recall semantics of knowledge, but when using knowledge operators to express information in the payment gateway's state for purposes of reasoning about erasure policy, the more appropriate semantics of knowledge is imperfect recall.

This example illustrates some of the subtleties that arise in the setting of reasoning about security and the way that our framework helps to represent them. Erasure policies have previously been studied in the computer security literature, beginning with (Chong and Myers 2005), though generally without consideration of strategic behaviour by the adversary. However, many other notions in the security literature do involve reasoning about strategies and agent's knowledge based on strategies, including nondeducibility on strategies (Wittbold and Johnson 1990) and robust declassification (Zdancewic and Myers 2001). We leave the further exploration of such notions using our approach for future work.

Model Checking

We now consider the problem of model checking in strategy space. Recall that, given an interpreted system $\mathcal{I}$, we write $\mathcal{I} \models \phi$ when $\mathcal{I}, (r, 0) \models \phi$ for all runs r of $\mathcal{I}$. Since interpreted systems are infinite, we need a finite representation for a decision problem, for this we use a finite state environment E together with an associated set of strategies Σ, and consider the problem of determining whether $\mathcal{I}(\mathcal{E}(E, \Sigma)) \models \phi$. Typically, Σ would be a set such as the set $\Sigma^{unif}(E)$ of all uniform strategies for E.

For generality, we abstract Σ further, to a paramaterized class such that for each environment E, the set $\Sigma(E)$ is a set strategies for E. We say that the parameterized class $\Sigma(E)$ is *PTIME-presented*, if it is presented by means of an algorithm that runs in time polynomial in the size of E and verifies if a given strategy α is in $\Sigma(E)$. (We assume that strategies α are given as a list of tuples $(s, a) \in S \times Acts$ such that $a \in \alpha(s)$.) For example, the class $\Sigma(E)$ of all strategies for E can be PTIME-presented, as can $\Sigma^{unif}(E)$, $\Sigma^{det}(E)$ and $\Sigma^{unif,det}(E)$.

A naive model checking algorithm would construct an extended transition system over the state space $S \times \Sigma(E)$ and then apply standard model checking techniques for temporal epistemic logic. Note that a joint strategy for an environment E can be represented in space $|S| \times |Acts|$, where S is the set of states of E and $Acts$ the set of joint actions. Thus, such an extended transition system generally has exponentially many states as a function of the size of E. This means that the naive procedure would require not less than exponential time. In fact it, is possible to do better than this (assuming that PSPACE is better than EXPTIME). We show the following:

Theorem 2 *Let $\Sigma(E)$ be a PTIME presented class of strate-*

*gies for environments E. The complexity of deciding, given an environment E, and a CTL*K formula ϕ for agents $\{e\} \cup Ags(E) \cup \sigma(Ags(E))$, whether $\mathcal{I}(E, \Sigma(E)) \models \phi$, is PSPACE-complete.*

It is interesting to note that, although we have significantly more expressive power than the temporal logic LTL on which we build, we achieve this without an increase in complexity: model checking LTL is already PSPACE-complete (Sistla and Clarke 1985). We note that this result also strongly suggests that we do have a strictly stronger expressive power than ATEL, since the complexity of model checking ATEL, assuming uniform strategies, is P^{NP}-complete. The class P^{NP} consists of problems solvable by PTIME computations with access to an NP oracle). For ATEL, model checking can be done with a polynomial time (with respect to the size of formula) computation with access to an oracle that is in NP with respect to both the number of states and the number of joint actions. In particular, (Schobbens 2004) proves this upper bound and (Jamroga and Dix 2006) proves a matching lower bound.

Conclusions

We refer the reader to the sections above for references and comparison to related work on each of the topics that we cover. Beside these references, the following are also worth mentioning.

Strategy Logic (Chatterjee, Henzinger, and Piterman 2010) is a (non-epistemic) generalization of ATL for perfect information strategies in which strategies may be explicitly named and quantified. Our logic has implicit quantification over strategies using the strategic agents in the epistemic operators. Our logic could be further generalized to make the reference to strategies explicit, but we note that strategy logic has a non-elementary model checking problem, so this could come at the cost of a large increase in complexity. Work on identification of more efficient variants of quantified strategy logic includes (Mogavero, Murano, and Vardi 2010), who formulate a variant with a 2-EXPTIME-complete model checking problem.

(van Eijck 2013) introduces a variant of propositional dynamic logic (PDL) for describing strategy profiles in normal form games subject to preference relations, but does not cover temporal aspects as we have in this paper. Another approach based on PDL is given in (Ramanujam and Simon 2008), which describes strategies by means of formulas.

(Schnoor 2010) defines a very rich generalization of ATEL for probabilistic environments, that includes variables that refer to *strategy choices*, and in which the strategic operators refer to these variables, so that statements of the form "when coalition A runs the strategy determined by variable S1, and coalition B runs the strategy determined by S2, then probability that ϕ holds is at least δ" can be expressed. Here a strategy choice uniformly maps each state, coalition and formula to a uniform strategy for the coalition. The strategic variables may be quantified, but only in the prefix of the formula. Because of the complexity of this framework, we leave a more detailed comparison for later work, but we note that the epistemic operators in this approach are interpreted with respect to a fixed strategy choice, compared to our approach in which the epistemic operators work over the entire strategy space.

Our focus on this paper has been on an observational, or imperfect recall, semantics for knowledge. Other semantics for knowledge are also worth consideration, but left for future work. We note one issue in relation to the connection to ATEL that we have established, should we consider a perfect recall version of our logic. ATEL operators effectively allow reference to situations in which agents switch their strategy after some actions have already been taken, whereas in our model an agent's strategy is fixed for the entire run. When switching to a new strategy, there is the possibility that the given state is not reachable under this new strategy. We have handled this issue in our translation by assuming that all states are initial, so that the run can be reinitialized if necessary to make the desired state reachable. This is consistent with an imperfect recall interpretation of ATEL, but it is not clear that this approach is available on a perfect recall interpretation. We leave a resolution of this issue to future work.

In several places in the present paper (Proposition 1 and Theorem 1), we have made a technical restriction to formulas not involving the common knowledge operator in order to establish a result about the expressive power of our logic, and remarked that an extension of the logic appears to be required for these results to encompass common knowledge operators. In a subsequent work (Huang and van der Meyden 2014), we have developed an extension that has the required expressiveness, by introducing a capacity for quantifying over strategies.

References

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-Time Temporal Logic. *Journal of the ACM* 49(5):672–713.

Brafman, R. I.; Latombe, J.-C.; Moses, Y.; and Shoham, Y. 1997. Applications of a Logic of Knowledge to Motion Planning under Uncertainty. *JACM* 44(5).

Chatterjee, K.; Henzinger, T. A.; and Piterman, N. 2010. Strategy logic. *Inf. Comput.* 208(6):677–693.

Chong, S., and Myers, A. C. 2005. Language-based information erasure. In *IEEE Computer Security Foundations Workshop*, 241–254.

Fagin, R.; Halpern, J.; Moses, Y.; and Vardi, M. 1995. *Reasoning About Knowledge*. The MIT Press.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1997. Knowledge-based programs. *Distributed Computing* 10(4):199–225.

Halpern, J. Y., and Moses, Y. 1990. Knowledge and Common Knowledge in a Distributed Environment. *JACM* 37(3):549–587.

Halpern, J. Y., and Moses, Y. 2007. Characterizing Solution Concepts in Games Using Knowledge-Based Programs. In *the 20nd International Joint Conference on Artificial Intelligence (IJCAI2007)*, 1300–1307.

Halpern, J. Y., and O'Neill, K. R. 2008. Secrecy in Multiagent Systems. *ACM Transactions on Information and System Security* 12(1).

Halpern, J. Y., and Rong, N. 2010. Cooperative Equilibrium (Extended Abstract). In *9th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*.

Horty., J. F. 2001. *Agency and deontic logic*. Oxford University Press.

Huang, X., and van der Meyden, R. 2014. An epistemic strategy logic. In *2nd Int. Workshop on Strategic Reasoning*, EPTCS. to appear.

Jamroga, W., and Ågotnes, T. 2007. Constructive knowledge: what agents can achieve under imperfect information. *Journal of Applied Non-Classical Logics* 17(4):423–475.

Jamroga, W., and Dix, J. 2006. Model checking abilities under incomplete information is indeed delta2-complete. In *the 4th European Workshop on Multi-Agent Systems (EUMAS'06)*.

Jamroga, W., and van der Hoek, W. 2004. Agents that Know How to Play . *Fundamenta Informaticae* 62:1–35.

Jamroga, W. 2003. Some Remarks on Alternating Temporal Epistemic Logic. In *Proceedings of Formal Approaches to Multi-Agent Systems (FAMAS 2003)*.

Jonker, G. 2003. Feasible strategies in alternating-time temporal. Master's thesis, University of Utrech, The Netherlands.

Mogavero, F.; Murano, A.; and Vardi, M. Y. 2010. Reasoning about strategies. In *FSTTCS*, 133–144.

Parikh, R., and Ramanujam, R. 1985. Distributed Processes and the Logic of Knowledge. In *Logics of Programs 1985*, 256–268.

Parikh, R. 1983. Propositional game logic. In *IEEE Symp. on Foundations of Computer Science*, 195–200.

Pauly, M. 2002. A modal logic for coalitional power in games. *Journal of Logic and Computation* 12(1):149–166.

Pnueli, A. 1977. The Temporal Logic of Programs. In *Symp. on Foundations of Computer Science*, 46–57.

Ramanujam, R., and Simon, S. E. 2008. Dynamic logic on games with structured strategies. In *KR*, 49–58.

Schnoor, H. 2010. Explicit strategies and quantification for atl with incomplete information and probabilistic games. Technical Report 1008, Institut für Informatik, Christian-Albrechts Universität zu Kiel.

Schobbens, P.-Y. 2004. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science* 85(2):82–93.

Sistla, A. P., and Clarke, E. M. 1985. The complexity of propositional linear temporal logics. *J. ACM* 32(3):733–749.

van der Hoek, W., and Wooldridge, M. 2002. Tractable multiagent planning for epistemic goals. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, 1167–1174.

van der Meyden, R. 1996. Knowledge Based Programs: On the Complexity of Perfect Recall in Finite Environments. In *6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 1996)*, 31–49.

van Eijck, J. 2013. PDL as a multi-agent strategy logic. In *Proc. Conf. on Theoretical Aspects of Reasoning about Knowledge*. published in CoRR, http://arxiv.org/abs/1310.6437.

van Otterloo, S., and Jonker, G. 2005. On Epistemic Temporal Strategic Logic. *Electronic Notes in Theoretical Computer Science (ENTCS)* 126:77–92.

W. van der Hoek; Jamroga, W.; and Wooldridge, M. 2005. A logic for strategic reasoning. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS'05)*, 157–164.

Wittbold, J. T., and Johnson, D. M. 1990. Information flow in nondeterministic systems. In *Proc. IEEE Symp. on Security and Privacy*, 144–161.

Zdancewic, S., and Myers, A. C. 2001. Robust declassification. In *IEEE Computer Security Foundations Workshop*, 15.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

An Abstraction Technique for the Verification of Multi-Agent Systems Against ATL Specifications

Alessio Lomuscio and Jakub Michaliszyn
Department of Computing, Imperial College London, UK

Abstract

We introduce an abstraction methodology for the verification of multi-agent systems against specifications expressed in alternating-time temporal logic (ATL). Inspired by methodologies such as predicate abstraction, we define a three-valued semantics for the interpretation of ATL formulas on concurrent game structures and compare it to the standard two-valued semantics. We define abstract models and establish preservation results on the three-valued semantics between abstract models and their concrete counterparts. We illustrate the methodology on the large state spaces resulting from a card game.

1 Introduction

In logic-based approaches to multi-agent systems (MAS) there is a long tradition concerned with the development and use of formalisms aimed at expressing the *strategic abilities* of agents in a system. This interest goes back to earlier work in philosophical logic where formalisms such as STIT have been developed to analyse what agents can bring about in a multi-agent system (Belnap and Perloff 1990). Against this context the framework of Alternating-Time Temporal Logic (ATL) was developed in the late 90s in theoretical computer science to reason about paths representing particular outcomes in games (Alur, Henzinger, and Kupferman 2002). For example, by means of an ATL formula one can express whether two agents in a game can always enforce a certain state of affairs irrespective of the actions of the others.

The semantics of ATL is traditionally given in terms of concurrent game structures, which are transition-based systems whereby all choices of the agents as well as their strategies are explicitly provided. Concurrent game structures are very close to interpreted systems, a popular semantics for reasoning about knowledge in multi-agent systems (Fagin et al. 1995). Combining the two approaches provides a natural and intuitive setting for the practical verification of MAS epistemic and ATL specifications under incomplete information (Lomuscio and Raimondi 2006b).

One key attractiveness of ATL is that key system properties such as controllability and realisability can be formulated as model checking particular ATL formulas. In the original setting, i.e., without fairness constraints, the algorithm for model checking models against CTL formulas can easily be extended to ATL, thereby resulting in the ATL model checking problem being linear against the size of the model and the formula to be considered. The complexity increases for ATL* and more powerful logics. Indeed in the past ten years a range of logics which extend ATL have been put forward, ranging from ATL$^+$ to ATL*, to combinations with knowledge, and to the very recent family of strategic logic (Chatterjee, Henzinger, and Piterman 2007; Mogavero et al. 2012), and their complexity analysed. Furthermore, different assumptions can be made on the observability of the states in the game and the states of the players, as well as their ability to recall the moves they played, thereby leading to several possible variants. The precise set-up has considerable consequences on the resulting complexity of the model checking problem with certain cases, e.g., perfect recall under incomplete information, resulting to be undecidable. For more details we refer to (Alur, Henzinger, and Kupferman 2002; Schobbens 2004; Jamroga and Ågotnes 2007; Bulling, Dix, and Chesñevar 2008).

Much is known at theoretical level about these variants. However, *practical* model checking of systems against ATL specifications has somewhat lagged behind. JMOCHA (Alur et al. 2001), a revised version of the original MOCHA model checker for reactive modules against ATL specifications, supports advanced features such as compositional and modular verification. MCMAS (Lomuscio and Raimondi 2006b; Lomuscio, Qu, and Raimondi 2009), a model checker for multi-agent systems, supports epistemic and ATL specifications in the context of incomplete information. However, they both lack abstraction features thereby making them more prone to be constrained by the state-explosion problem.

The aim of this paper is to lay the foundations for an abstraction methodology for verifying multi-agent systems with incomplete information against ATL specifications.

Contribution. Since our overall aim is to develop under-Section 2 by developing a novel three-valued semantics for ATL which we show to be a conservative extension of the classic two-valued semantics. In Section 3 we define abstractions of ATL models. Abstraction is conducted modularly both on the local states and the local actions associated with the agents, thereby generating considerably smaller models. We show that values true and false are preserved in three-valued semantics between abstract and concrete models. We

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

turn to the model checking problem in Section 4 by exploring its complexity in a three-valued setting and by providing constructive definitions for the initial abstraction and algorithms for successive refinements. We exemplify these constructions in Section 5 by showing how the technique put forward here can reduce the state space considerably. We conclude in Section 6 where we also discuss the future work.

Related work. As discussed above, while model checking systems against ATL specifications has been explored in the literature, practical model checking has received considerable less attention. While model checkers such as JMOCHA and MCMAS have been made available, one of the key problems remains the state-space explosion. To the best of our knowledge the only previous paper in the literature tackling abstraction in the context of ATL is (Köster and Lohmann 2012). However, their approach is based on modular interpreted systems where complete information is assumed. By contrast here we deal with the general class of interpreted systems and we work in a more general, three-valued semantics, thereby laying the foundation for predicate abstraction techniques.

2 Three-Valued Alternating-Time Temporal Logic

In this section we recall the technical set-up for ATL and provide it with a novel, three-valued semantics.

We assume a set of agents $Ag = \{1, \ldots, m\}$. For simplicity, and without loss of generality, note we do not consider the environment here. We use $\Gamma \subseteq Ag$ to denote subsets of agents. By $\overline{\Gamma}$ we denote the complementation of Γ, i.e., $Ag \setminus \Gamma$.

We also assume a set of propositional variables $\mathcal{V}$; by $\overline{\mathcal{V}}$ we denote the set of all the literals containing propositional variables from $\mathcal{V}$, i.e., $\overline{\mathcal{V}} = \{q, \neg q \mid q \in \mathcal{V}\}$.

2.1 Interpreted systems and associated models

We are interested in the following structures.

Definition 1 (Interpreted system). *An* interpreted system *is a tuple $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$ such that for each agent i:*

- *L_i is a set of possible* local states*;*
- *Act_i is a set of possible* local actions*;*
- *$P_i : L_i \to 2^{Act_i} \setminus \{\emptyset\}$ is a* local protocol*;*
- *$t_i \subseteq L_i \times ACT \times 2^{L_i}$ is a* local transition relation *with $ACT = Act_1 \times \cdots \times Act_m$;*
- *$I \subseteq L_1 \times \cdots \times L_m$ is a set of* global initial states*;*
- *$\Pi : L_1 \times \cdots \times L_m \to \overline{\mathcal{V}}$ is a* labelling function *such that for any q, s we either have $q \notin \Pi(s)$ or $\neg q \notin \Pi(s)$.*

Note that the transition relation is normally considered to be a function. We here consider a more liberal definition to allow for a uniform presentation with abstract models (see Section 2.4). For the same reason the definition of a labelling function is more general than usual as it allows a state to be labelled by neither q nor $\neg q$.

For a tuple $t = (t_1, \ldots, t_m)$, by $t.i$ we denote its ith element t_i, with $i \leq m$. We will use this notation to identify individual local states and local actions.

We associate standard temporal models to interpreted systems.

Definition 2 (Model). *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$, its* associated model *$M_{IS} = (S, T, I, \Pi)$ is a tuple such that:*

- *$S \subseteq L_1 \times \cdots \times L_m$ is the set of global states reachable via T from the set of initial global states $I \subseteq S$,*
- *$T \subseteq S \times ACT \times S$ is a global transition relation such that $T((l_1, \ldots, l_m), a, (l'_1, \ldots, l'_m))$ iff $t_i(l_i, a, l'_i)$ and $a.i \in P_i(l_i)$ for all $i \in Ag$.*

When IS is clear from the context we write M for M_{IS}.

In the following we assume that all locally enabled joint actions can be executed, i.e., that for all global states $s \in S$ and joint actions $a \in ACT$ such that $a.i \in P_i(s.i)$ for all $i \in Ag$, there exists an $s' \in S$ such that $T(s, a, s')$.

2.2 Alternating-time temporal logic

Alternating-Time Temporal Logic (ATL) was introduced to reason about agents and their strategies in the context of perfect information (Alur, Henzinger, and Kupferman 2002). In this setting agents' strategies depend on the global state of the system. ATL's semantics has been recast on incomplete information in the context of interpreted systems (see, e.g., (Schobbens 2004)).

The syntax of ATL is given by the following BNF:
$\varphi ::= q \mid \mathrm{tt} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle\Gamma\rangle\!\rangle X\varphi \mid \langle\!\langle\Gamma\rangle\!\rangle(\varphi U \varphi) \mid \langle\!\langle\Gamma\rangle\!\rangle G\varphi$
where q is a propositional variable and $\Gamma \subseteq Ag$. We abbreviate $\langle\!\langle\Gamma\rangle\!\rangle(\mathrm{tt} U \varphi)$ by $\langle\!\langle\Gamma\rangle\!\rangle F\varphi$.

The formula $\langle\!\langle\Gamma\rangle\!\rangle X\varphi$ is read as "$\langle\!\langle\Gamma\rangle\!\rangle$ has a strategy to enforce φ in the next state (irrespective of the actions of the agents in $Ag \setminus \Gamma$)"; $\langle\!\langle\Gamma\rangle\!\rangle G\varphi$ means "$\langle\!\langle\Gamma\rangle\!\rangle$ has a strategy to enforce φ forever in the future"; and $\langle\!\langle\Gamma\rangle\!\rangle \varphi_1 U \varphi_2$ represents "$\langle\!\langle\Gamma\rangle\!\rangle$ has a strategy to enforce that φ_2 holds at some point in the future and can ensure that φ_1 holds until then."

2.3 Two- and three-valued semantics for ATL

Let $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$ be an interpreted system. A *strategy* for an agent $i \in \Gamma$ is a function $f_i : L_i^+ \to 2^{Act_i} \setminus \{\emptyset\}$ such that for each sequence of states $l_1 \ldots l_s$, all the actions in $f_i(l_1 \ldots l_s)$ are in $P_i(l_s)$.

A strategy is called *memoryless* if it depends only on the last state, i.e., if $f_i(l_1 \ldots l_s l) = f_i(l)$ for any natural number s and any states $l_1, \ldots, l_s, l$.

Given a set of agents Γ and an indexed set of strategies $F_\Gamma = \{f_i \mid i \in \Gamma\}$, we define $out(s, F_\Gamma)$ to be the set of infinite paths $s_0 s_1 \ldots$ such that $s_0 = s$ and for all j there exists an action a such that $T(s_j, a, s_{j+1})$ and for all $i \in \Gamma$, $a.i \in f(s_0.i \ldots s_j.i)$.

Agents anre Interpreted systems with *imperfect recall* are those in which agents are required to use only memoryless strategies, while interpreted systems with *perfect recall* are those in which agents are allowed to use arbitrary strategies.

For a path $p = s_0 s_1 \ldots$, by p^i we denote s_i, the i-th element of p.

The following is the standard semantics for ATL.

Definition 3 (Two-valued semantics). *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$, its associated*

model M and a global state $s \in S$, the two-valued satisfaction relation $\models_{\overline{2}}$ is inductively defined as follows.

$M, s \models_{\overline{2}} q$ *iff* $q \in \Pi(s)$

$M, s \models_{\overline{2}} \neg\varphi$ *iff* $M, s \not\models_{\overline{2}} \varphi$

$M, s \models_{\overline{2}} \langle\langle\Gamma\rangle\rangle X\varphi$ *iff for some* F_Γ *and all* $p \in out(s, F_\Gamma)$ *we have* $M, p^1 \models_{\overline{2}} \varphi$

$M, s \models_{\overline{2}} \varphi_1 \wedge \varphi_2$ *iff* $M, l \models_{\overline{2}} \varphi_1$ *and* $M, s \models_{\overline{2}} \varphi_2$

$M, s \models_{\overline{2}} \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2$ *iff for some* F_Γ *and all* $p \in out(s, F_\Gamma)$, *there is a* $k \geq 0$ *s.t. we have* $M, p^k \models_{\overline{2}} \varphi_2$ *and for all* $0 \leq j < k, M, p^j \models_{\overline{2}} \varphi_1$

$M, s \models_{\overline{2}} \langle\langle\Gamma\rangle\rangle G\varphi$ *iff for some* F_Γ *and all* $p \in out(s, F_\Gamma)$ *and for all* $i \geq 0$ *we have* $M, p^i \models_{\overline{2}} \varphi$

We now introduce a novel, three-valued semantics for ATL. Each ATL formula φ is associated with three possible values (tt for true, ff for false, uu for undefined) when evaluated on an interpreted system IS at a given state. We will later show that the three-valued semantics agrees with the two-valued one whenever the truth value is not undefined.

Definition 4 (Three-valued semantics). *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$, its associated model M and a global state $s \in S$, the three-valued satisfaction relation $\models_{\overline{3}}$ is inductively defined as follows.*

$$M, s \models_{\overline{3}} q = \begin{cases} \mathrm{tt} \textit{ if } q \in \Pi(s), \\ \mathrm{ff} \textit{ if } \neg q \in \Pi(s), \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

$$M, s \models_{\overline{3}} \neg\varphi = \begin{cases} \mathrm{tt} \textit{ iff } M, s \models_{\overline{3}} \varphi = \mathrm{ff} \\ \mathrm{ff} \textit{ iff } M, s \models_{\overline{3}} \varphi = \mathrm{tt} \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

$$M, s \models_{\overline{3}} \varphi_1 \wedge \varphi_2 = \begin{cases} \mathrm{tt} \textit{ iff } M, s \models_{\overline{3}} \varphi_1 = \mathrm{tt} \textit{ and } M, s \models_{\overline{3}} \varphi_2 = \mathrm{tt} \\ \mathrm{ff} \textit{ iff } M, s \models_{\overline{3}} \varphi_1 = \mathrm{ff} \textit{ or } M, s \models_{\overline{3}} \varphi_2 = \mathrm{ff} \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

$$M, s \models_{\overline{3}} \langle\langle\Gamma\rangle\rangle X\varphi = \begin{cases} \mathrm{tt} \textit{ iff for some strategy } F_\Gamma \textit{ and all } p \in out(s, F_\Gamma) \textit{ we have } M, p^1 \models_{\overline{3}} \varphi = \mathrm{tt} \\ \mathrm{ff} \textit{ iff for some strategy } F_{\overline{\Gamma}} \textit{ and all } p \in out(s, F_{\overline{\Gamma}}) \textit{ we have } M, p^1 \models_{\overline{3}} \varphi = \mathrm{ff} \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

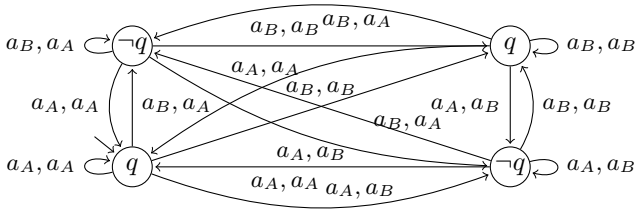
$$M, s \models_{\overline{3}} \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2 = \begin{cases} \mathrm{tt} \textit{ if for some strategy } F_\Gamma \textit{ and all } p \in out(s, F_\Gamma) \textit{ there is } k \in \mathbb{N} \textit{ s.t. } M, p^k \models_{\overline{3}} \varphi_2 = \mathrm{tt} \textit{ and for all } j < k, M, p^j \models_{\overline{3}} \varphi_1 = \mathrm{tt} \\ \mathrm{ff} \textit{ if for some strategy } F_{\overline{\Gamma}} \textit{ and all } p \in out(s, F_{\overline{\Gamma}}), k \in \mathbb{N} \textit{ we have } M, p^k \models_{\overline{3}} \varphi_2 = \mathrm{ff} \textit{ or there is } j < k \textit{ s.t. } M, p^j \models_{\overline{3}} \varphi_1 = \mathrm{ff} \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

$$M, s \models_{\overline{3}} \langle\langle\Gamma\rangle\rangle G\varphi = \begin{cases} \mathrm{tt} \textit{ iff for some strategy } F_\Gamma \textit{ and all } p \in out(s, F_\Gamma), i \in \mathbb{N} \textit{ we have } M, p^i \models_{\overline{3}} \varphi = \mathrm{tt} \\ \mathrm{ff} \textit{ iff for some strategy } F_{\overline{\Gamma}} \textit{ and all } p \in out(s, F_{\overline{\Gamma}}), \textit{ there is } i \in \mathbb{N} \textit{ s.t. } M, p^i \models_{\overline{3}} \varphi = \mathrm{ff} \\ \mathrm{uu} \textit{ otherwise} \end{cases}$$

We write $IS \models_{\overline{2}} \varphi$ iff for all the initial states s we have that $M, s \models_{\overline{2}} \varphi$. Similarly, $IS \models_{\overline{3}} \varphi = \mathrm{tt}$ iff for all the initial states s, $M, s \models_{\overline{3}} \varphi = \mathrm{tt}$. We illustrate the differences between the two semantics on an example.

Example 1. Consider the interpreted system IS defined on $Ag = \{1, 2\}$. Each agent $i \in \{1, 2\}$ is associated with two actions $Act_i = \{a_A, a_B\}$, two local states $L_i = \{l_A, l_B\}$, a protocol function P_i s.t. $P_i(l) = Act_i$ for any $l \in L_i$, and the transition relation $t_i = \{(l_A, a_A, l_A), (l_A, a_B, l_B), (l_B, a_A, l_A), (l_B, a_B, l_B)\}$.

Assume that $I = \{(l_A, l_A)\}$ and Π is the labelling function such that $\Pi((l_A, l_A)) = \Pi((l_B, l_B)) = \{q\}$ and $\Pi((l_B, l_A)) = \Pi((l_A, l_B)) = \{\neg q\}$. Then, in the model M_{IS} of IS we have $T = \{((l_a, l_b), (a_c, a_d), (l_c, l_d)) \mid a, b, c, d \in \{A, B\}\}$.



Consider a strategy f_1 for agent 1. If $a_A \in f_1((l_A))$, then $out((l_A, l_A), \{f_1\})$ contains a path p such that $p^1 = (l_A, l_B)$. If $a_A \notin f_1((l_A))$, then $a_B \in f_1((l_A, l_A))$, and therefore $out((l_A, l_A), \{f_1\})$ contains a path p' such that $p'^1 = (l_B, l_A)$. Therefore, $IS \not\models_{\overline{2}} \langle\langle 1\rangle\rangle Xq$ and it is not the case that $IS \models_{\overline{3}} \langle\langle 1\rangle\rangle Xq = \mathrm{tt}$.

Note that $\overline{\{1\}} = \{2\}$. Consider a strategy f_2 for agent 2. If $a_B \in f_2((l_A))$, then there is $p \in out((l_A, l_A), \{f_2\})$ s.t. $p^1 = (l_B, l_A)$. Otherwise, $a_A \in f_2((l_A))$ and there is $p \in out((l_A, l_A), \{f_2\})$ s.t. $p^1 = (l_A, l_B)$. So it is not the case that $IS \models_{\overline{3}} \langle\langle 1\rangle\rangle Xq = \mathrm{ff}$, and therefore $IS \models_{\overline{3}} \langle\langle 1\rangle\rangle Xq = \mathrm{uu}$.

In other words, $IS \not\models_{\overline{2}} \langle\langle 1\rangle\rangle Xq$ means that agent 1 has no strategy to guarantee q in the next state, while $IS \models_{\overline{3}} \langle\langle 1\rangle\rangle Xq = \mathrm{ff}$ represents that agent 2 has a strategy to avoid q (so it is not the case).

2.4 Relationship between $\models_{\overline{3}}$ and $\models_{\overline{2}}$

We show that the three-value semantics agrees with the two-valued semantics whenever the three-valued semantics returns tt or ff.

Theorem 5. *Consider an interpreted system IS, an ATL formula φ and a global state s. We have the following:*

1. *If $M_{IS}, s \models_{\overline{3}} \varphi = \mathrm{tt}$, then $M_{IS}, s \models_{\overline{2}} \varphi$.*
2. *If $M_{IS}, s \models_{\overline{3}} \varphi = \mathrm{ff}$, then $M_{IS}, s \not\models_{\overline{2}} \varphi$.*

In the proof we use the following lemma.

Lemma 6. *For any disjoint groups of agents Γ, Γ', any state s and any strategies $F_\Gamma, F_{\Gamma'}$ we have $out(s, F_\Gamma \cup F_{\Gamma'}) \subseteq out(s, F_\Gamma)$.*

The proof follows from the fact that $out(s, F_\Gamma \cup F_{\Gamma'})$ represents the paths that result when agents from Γ play according to F_Γ and agents from Γ' play according to $F_{\Gamma'}$, while $out(s, F_\Gamma)$ represents the paths that occur when agents from Γ play according to F_Γ.

Proof of Theorem 5. The proof is by induction on φ.

If $M_{IS}, s \models_3 q = \mathrm{tt}$, then $q \in \Pi(s)$, and so $M_{IS}, s \models_2 q$.

If $M_{IS}, s \models_3 q = \mathrm{ff}$, then $\neg q \in \Pi(s)$. By the definition of the labelling function, $q \notin \Pi(s)$. Therefore, $M_{IS}, s \not\models_2 q$.

Assume that $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle X\varphi' = \mathrm{tt}$. Then, there is F_Γ such that for all $p \in out(s, F_\Gamma)$ we have $M_{IS}, p^1 \models_3 \varphi' = \mathrm{tt}$. By the inductive hypothesis, we have $M_{IS}, p^1 \models_2 \varphi'$, and therefore $M_{IS}, s \models_2 \langle\langle\Gamma\rangle\rangle X\varphi'$.

Assume that $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle X\varphi' = \mathrm{ff}$. So there is a strategy $F_{\overline{\Gamma}}$ s.t. for all $p \in out(s, F_{\overline{\Gamma}})$ we have $M_{IS}, p^1 \models_3 \varphi' = \mathrm{ff}$. Consider any strategy F_Γ of Γ and a path $p \in out(s, F_\Gamma \cup F_{\overline{\Gamma}})$. By Lemma 6, $p \in out(s, F_{\overline{\Gamma}})$, so $M_{IS}, p^1 \models_3 \varphi = \mathrm{ff}$. By the inductive hypothesis, $M_{IS}, p^1 \not\models_2 \varphi'$. By Lemma 6, $p \in out(s, F_\Gamma)$. So we showed that for any strategy F_Γ there is a path p such that $M_{IS}, p^1 \not\models_2 \varphi'$; so $M_{IS}, s \not\models_2 \langle\langle\Gamma\rangle\rangle X\varphi'$.

Assume that $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2 = \mathrm{tt}$. Then, there is F_Γ such that for all $p \in out(s, F_\Gamma)$ there is k s.t. $M_{IS}, p^k \models_3 \varphi_2 = \mathrm{tt}$ and for all $j < k$ we have $M_{IS}, p^j \models_3 \varphi_1 = \mathrm{tt}$. By the inductive hypothesis we have that $M_{IS}, p^k \models_2 \varphi_2$ and for all $j < k$ we have $M_{IS}, p^j \models_2 \varphi_1$. Therefore $M_{IS}, s \models_2 \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2$.

Assume that $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2 = \mathrm{ff}$ and $F_{\overline{\Gamma}}$ is such that for all $p \in out(s, F_{\overline{\Gamma}})$ and all k we have $M_{IS}, p^k \models_3 \varphi_2 = \mathrm{ff}$ or there exists $j < k$ s.t. $M_{IS}, p^j \models_3 \varphi_1 = \mathrm{ff}$. Consider any strategy F_Γ of Γ, and let $p \in out(s, F_\Gamma \cup F_{\overline{\Gamma}})$. By Lemma 6, $p \in out(s, F_{\overline{\Gamma}})$ and $p \in out(s, F_\Gamma)$. By the inductive assumption, for all k, $M_{IS}, p^k \not\models_2 \varphi_2$ or there exists $j < k$ s.t. $M_{IS}, p^j \not\models_2 \varphi_1$. Since F_Γ was an arbitrary strategy, we conclude that $M_{IS}, s \not\models_2 \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2$.

The remaining inductive cases can be shown in a similar manner. □

Notice that the above proof remains valid if we consider only memoryless strategies.

In the next Section we will explore how the novel, three-valued semantics introduced here can be used as part of an abstraction methodology in order to verify multi-agent systems against ATL specifications.

3 Abstraction

In this section we provide an abstraction methodology for constructing abstract interpreted systems. We introduce two techniques: *action abstraction* and *state abstraction*.

Recall that a function is *decomposable* if for any $x_1, \ldots, x_m$ we have that $f((x_1, \ldots, x_m)) = (f_1(x_1), \ldots, f_m(x_m))$, for some $f_i, i = 1, \ldots, m$. By $f.i$ we denote the function f_i.

Given an interpreted system IS an *action abstraction function* is a surjective and decomposable function $\alpha : ACT \to ACT^\alpha$, for some set ACT^α. The set ACT is the set of joint actions whereas ACT^α constitutes the set of *abstract joint actions* for the corresponding abstract interpreted system. This enables us to define the action abstraction of an interpreted system as follows.

Definition 7 (Action abstraction). *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$ and an action abstraction function α, the* action abstraction *of IS w.r.t. α is an interpreted system $IS^\alpha = (\{L_i^\alpha, Act_i^\alpha, P_i^\alpha, t_i^\alpha\}_{i \in Ag}, I^\alpha, \Pi^\alpha)$ such that $I^\alpha = I$, $\Pi^\alpha = \Pi$, and for each agent i:*

- $L_i^\alpha = L_i$,
- $Act_i^\alpha = \alpha.i(Act_i)$,
- $P_i^\alpha(l) = \alpha.i(P_i(l))$ *for all states* $l \in L_i$,
- *for all $l, l' \in l_i$ and $a^\alpha \in ACT^\alpha$, $t_i^\alpha(l, a^\alpha, l')$ iff for some $a \in ACT$ s.t. $\alpha.i(a) = a^\alpha$ we have $t_i(l, a, l')$.*

In other words the application of an action abstraction function results in the reduction of the number of actions in an interpreted system, thereby generating an abstract one.

We now explore a similar concept related to states.

Given an interpreted system IS, a *state abstraction function* is a surjective and decomposable function $\sigma : S \to S^\sigma$, such that for each agent i and any two local states $l, l' \in L_i$, if $\sigma.i(l) = \sigma.i(l')$, then $P_i(l) = P_i(l')$, for some set of states S^σ. The set S is the set of global states of the associated model M_{IS} and S^σ constitutes the set of *abstract global states* for the corresponding abstract interpreted system.

Definition 8 (State abstraction). *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$ and a state abstraction function σ, the* state abstraction *of IS w.r.t. σ is the interpreted system $IS^\sigma = (\{L_i^\sigma, Act_i^\sigma, P_i^\sigma, t_i^\sigma\}_{i \in Ag}, I^\sigma, \Pi^\sigma)$, where $I^\sigma = \sigma(I)$, $\Pi^\sigma(s^\sigma) = \bigcap_{s \in \sigma^{-1}(\{s^\sigma\})} \Pi(s)$ and for each agent i*

- $L_i^\sigma = \sigma.i(L_i)$,
- $Act_i^\sigma = Act_i$,
- $P_i^\sigma(l^\sigma) = \bigcup_{l \in \sigma.i^{-1}(\{l^\sigma\})} P_i(l)$,
- $t_i^\sigma(l^\sigma, a, l'^\sigma)$ *iff for some l, l' such that $\sigma.i(l) = l^\sigma$ and $\sigma.i(l') = l'^\sigma$ we have $t_i(l, a, l')$.*

Example 2. Assume that we have two states: s labelled by $\{p, \neg q\}$ and s' labelled by $\{p, q\}$ such that $\sigma(s) = \sigma(s')$. The resulting state will be labelled by $\Pi^\sigma(\sigma(s)) = \{p\}$.

Similarly to the case above the application of a state abstraction function results in the reduction of the number of states in an interpreted system, thereby generating an abstract one.

Given an interpreted system IS, an *abstraction function* is a pair $\delta = (\alpha, \sigma)$, where α is an action abstraction function of IS and σ is an state abstraction function of IS^α.

Definition 9 (Abstraction). *Given an interpreted system $IS = (\{L_i, Act_i, P_i, t_i\}_{i \in Ag}, I, \Pi)$ and an abstraction function $\delta = (\alpha, \sigma)$, the* abstraction *of IS w.r.t. δ is the interpreted system IS^δ that is the state abstraction of IS^α w.r.t. σ.*

3.1 Preservation Theorem

If σ is a state abstraction function and p is a path in an interpreted system IS, then by $\sigma(p)$ we denote the path $\sigma(p^0)\sigma(p^1)\ldots$, i.e., the corresponding path in the abstract interpreted system IS^σ.

The following lemma illustrates the relationship between paths in an interpreted system and its state abstraction.

Lemma 10. *Given an interpreted system IS, let σ be a state abstraction function of IS, $s \in S$ be a state and F^σ_Γ be a strategy for a group of agents $\Gamma \subseteq Ag$. Then there is a strategy F_Γ for the agents in Γ such that $\sigma(out(s, F_\Gamma)) \subseteq out(\sigma(s), F^\sigma_\Gamma)$.*

Proof. Consider IS^σ and a strategy $F^\sigma_\Gamma = \{f^\sigma_i \mid i \in \Gamma\}$. Let $F_\Gamma = \{f_i \mid i \in \Gamma\}$ where $f_i(l^0_i \ldots l^k_i) = \sigma.i^{-1}(f^\sigma_i(\sigma.i(l^0_i)) \ldots \sigma.i(l^k_i)))$. It can be checked that F_Γ is a strategy on IS.

Let $p \in out(s, F_\Gamma)$. So for each state p^j there is a joint action a such that $T(p^j, a, p^{j+1})$. By definition we have that $T^\sigma(\sigma(p^j), a, \sigma(p^{j+1}))$ on IS^σ, and therefore $\sigma(p) \in out(\sigma(s), F^\sigma_\Gamma)$. □

Lemma 11. *Given an interpreted system IS, let α be an action abstraction function of IS, $s \in S$ be a state and F^α_Γ be a strategy for a group of agents $\Gamma \subseteq Ag$. Then there is a strategy F_Γ for Γ such that $out(s, F_\Gamma) \subseteq out(s, F^\alpha_\Gamma)$.*

Proof. Let $\alpha^{-1} \circ F^\alpha_\Gamma$ denote the set $\{\alpha^{-1} \circ f^\alpha_i \mid i \in \Gamma\}$, where α^{-1} is the counter-image of the surjective function α.

It can be checked from the definition of action abstraction that $out(s, \alpha^{-1} \circ F^\alpha_\Gamma) \subseteq out(s, F^\alpha_\Gamma)$. □

We prove the following theorem.

Theorem 12. *Let $\delta = (\alpha, \sigma)$ be an abstraction function for an interpreted system IS . Then, for any state $s \in S$*

1. *If $M^\delta_{IS}, \sigma(s) \models_3 \varphi = \mathrm{tt}$, then $M_{IS}, s \models_3 \varphi = \mathrm{tt}$.*
2. *If $M^\delta_{IS}, \sigma(s) \models_3 \varphi = \mathrm{ff}$, then $M_{IS}, s \models_3 \varphi = \mathrm{ff}$.*

Proof. The proof is by induction on φ.

Consider a state s and the case $\varphi = q$. If $M^\delta_{IS}, \sigma(s) \models_3 q = \mathrm{tt}$, then $q \in \Pi^\delta(s)$. It follows by definition that $q \in \bigcap_{s' \in \sigma^{-1}(\sigma(s))} \Pi^\alpha(\sigma(s'))$, and so $q \in \Pi^\alpha(s)$. Since $\Pi^\alpha = \Pi$, $q \in \Pi(s)$, we conclude that $M_{IS}, s \models_3 q = \mathrm{tt}$.

If $M^\delta_{IS}, \sigma(s) \models_3 q = \mathrm{ff}$, then $\neg q \in \Pi^\delta(s)$. By similar considerations we can conclude that $\neg q \in \bigcap_{s' \in \sigma^{-1}(s)} \Pi^\alpha(s')$, so $\neg q \in \Pi^\alpha(s) = \Pi(s)$ and $M_{IS}, s \models_3 q = \mathrm{ff}$.

Assume that $\varphi = \neg\varphi'$. If $M^\delta_{IS}, \sigma(s) \models_3 \neg\varphi' = \mathrm{tt}$, then $M^\delta_{IS}, \sigma(s) \models_3 \varphi' = \mathrm{ff}$. By the inductive assumption $M_{IS}, s \models_3 \varphi' = \mathrm{ff}$, and therefore $M_{IS}, s \models_3 \neg\varphi' = \mathrm{tt}$. Similarly, if $M^\delta_{IS}, \sigma(s) \models_3 \neg\varphi' = \mathrm{ff}$, then $M_{IS}, s \models_3 \neg\varphi' = \mathrm{ff}$.

The case for $\varphi = \varphi_1 \wedge \varphi_2$ is straightforward through by the inductive hypothesis.

The other inductive cases can be shown by using Lemma 10 and 11. We only report the case of $M^\delta_{IS}, \sigma(s) \models_3 \langle\!\langle\Gamma\rangle\!\rangle \varphi_1 U \varphi_2 = \mathrm{tt}$; the others can be proven similarly.

Assume that $M^\delta_{IS}, \sigma(s) \models_3 \langle\!\langle\Gamma\rangle\!\rangle \varphi_1 U \varphi_2 = \mathrm{tt}$ and let $F^\delta_\Gamma = \{f^\delta_i \mid i \in \Gamma\}$ be a strategy such that for all $p^\delta \in out(\sigma(s), F^\delta_\Gamma)$, there is $k \geq 0$ s.t. $M^\delta_{IS}, p^k \models_3 \varphi_2 = \mathrm{tt}$ and for all $0 \leq j < k$ we have $M^\delta_{IS}, p^j \models_3 \varphi_1 = \mathrm{tt}$.

Let F^α_Γ be a strategy obtained by applying Lemma 10 to F^δ_Γ, and F_Γ be a result of applying Lemma 11 to F^α_Γ.

Consider any $p \in out(s, F_\Gamma)$. By Lemmas 10 and 11, $\sigma(p) \in out(\sigma(s), F^\delta_\Gamma)$, and therefore for some $k \geq 0$ we have $M^\delta_{IS}, \sigma(p)^k \models_3 \varphi_2 = \mathrm{tt}$ and for all $0 \leq j < k$ we have $M^\delta_{IS}, \sigma(p)^j \models_3 \varphi_1 = \mathrm{tt}$.

By the inductive hypothesis we have that $M_{IS}, p^k \models_3 \varphi_2 = \mathrm{tt}$ and for all $0 \leq j < k$, $M_{IS}, p^j \models_3 \varphi_1 = \mathrm{tt}$. It follows that $M_{IS}, s \models_3 \langle\!\langle\Gamma\rangle\!\rangle \varphi_1 U \varphi_2 = \mathrm{tt}$. □

Observe that the theorem above holds irrespective of any assumption on the strategies being memoryless or memoryfull.

Remark. There are an interpreted system IS, a state abstraction function σ, an action abstraction function α, and an ATL property φ s.t. $IS \models_2 \varphi$ but $IS^\alpha \not\models_2 \varphi$ and $IS^\sigma \not\models_2 \varphi$.

Indeed, consider the IS from Example 1 but with a different labelling function: $\Pi((l_A, l_A)) = \Pi((l_A, l_B)) = \{q\}$ and $\Pi((l_B, l_A)) = \Pi((l_B, l_B)) = \{\neg q\}$. Let $\varphi = \langle\!\langle 1\rangle\!\rangle X q$. Clearly, $IS \models_2 \varphi$. Consider an action abstraction function α such that $\alpha((a_1, a_2)) = (\epsilon, \epsilon)$ for all $a_1, a_2 \in \{a_A, a_B\}$. We have $IS^\alpha \not\models_2 \varphi$ since in the initial state 1 can only play action ϵ which may lead to the state (l_B, l_B) labelled with $\neg q$. Similarly, consider a state abstraction function σ such that $\sigma((s_1, s_2)) = (\epsilon, \epsilon)$ for all $s_1, s_2 \in \{l_A, l_B\}$. The state abstraction IS^σ of IS w.r.t. σ contains only one state (ϵ, ϵ) s.t. $\Pi^\sigma((\epsilon, \epsilon)) = \emptyset$, and therefore $IS^\sigma \not\models_2 \varphi$.

4 Model Checking Using Abstraction

In this section we put forward a methodology based on the abstraction results of the previous Section to model check multi-agent systems against ATL specifications. Recall that the model checking problem is defined as follows.

Definition 13 (Model checking problem). *Given an interpreted system IS and an ATL specification φ, the* model checking problem *involves establishing whether $IS \models_2 \varphi$.*

The following theorem was stated in (Alur, Henzinger, and Kupferman 2002) and, to the best of our knowledge, showed in (Dima and Tiplea 2011).

Theorem 14. *Model checking interpreted systems with perfect recall and imperfect information against ATL specifications is undecidable.*

However, the problem is decidable if memoryless (as opposed to perfect recall) strategies are considered (Schobbens 2004). Recall that $\Delta^P_2 = \mathrm{P}^{\mathrm{NP}}$ is the class of problems that can be solved in polynomial time by a deterministic Turing machine with an NP oracle.

Theorem 15. *Model checking interpreted systems with imperfect information and memoryless strategies against ATL specifications is decidable in Δ^P_2.*

The above complexity is given in the size of the model associated with an interpreted system (observe that the complexity on the size of the interpreted system itself is likely to be harder (Lomuscio and Raimondi 2006a)). However, note

that models of interpreted systems may be exponential in size of the interpreted system that generates them. Since model checkers typically work on the generated models, this creates the well-known difficulty of the *state-space explosion*.

To mitigate this difficulty in the following we develop a technique that allows us to avoid the construction of the (concrete) model associated with an interpreted system and consider its abstraction instead. By the results of the previous section, in several cases this will prove sufficient.

More specifically we proceed as follows. Given an interpreted system and a formula to be verified, firstly we build an action abstraction, then a state abstraction from it (see Subsection 4.2) and attempt verification on the result (see Subsection 4.1). If the result is either tt or ff we know that (via Theorem 12) this is also the result of the check on the concrete model. Otherwise, we perform a refinement procedure (see Subsection 4.3) to obtain a further abstract model and attempt verification again. We repeat this procedure until we obtain a model in which the specification can be verified.

4.1 Three-valued model checking interpreted systems

We extend the classical model checking problem recalled above into the more general formulation below.

Definition 16 (Three-valued model checking problem). *Given an interpreted system IS, an ATL specification φ and a truth value $b \in \{\mathrm{ff}, \mathrm{uu}, \mathrm{tt}\}$, the* three-valued model checking problem *involves establishing whether $IS \models_3 \varphi = b$.*

We can show that model checking interpreted systems against ATL specifications under a three-valued semantics and memoryless strategies is also in Δ_2^P.

Proposition 17. *Three-valued model checking interpreted systems with imperfect information and memoryless strategies against ATL specifications is decidable in Δ_2^P.*

Proof. For each subformula, starting from the propositions, label all the states with the value of the formula in this state, using the values of its subformulas. This is straightforward in all cases except the subformulas of the form $\langle\langle\Gamma\rangle\rangle$; in these cases call a non-deterministic oracle a polynomial number of times to obtain a strategy as required. Since $\Delta_2^P = P^{NP}$, we obtain the result. □

Notice that the above procedure establishes the value of each subformula in each state. We will use this property later to define the refinement procedure.

4.2 Initial abstraction

Let $A(\varphi)$ denote the set of agents in φ and $\mathcal{V}(\varphi)$ be the set of propositional variables in φ. For an interpreted system IS and a local state $l_i \in L_i$, the labelling of l_i is $\Pi(l_i) = \bigcap_{s \in S, s.i = l_i} \Pi(s)$.

For a set of agents Γ, we define $\alpha^I_{IS}(\Gamma)$ to be the action abstraction function α for IS such that for every agent $i \in \Gamma$ we define $\alpha.i(a_i) = a_i$, and for every $i \notin \Gamma$ we let $\alpha.i(a_i) = \epsilon$. So, in other words, $\alpha^I_{IS}(\Gamma)$ returns an interpreted system disregarding all actions not in Γ.

Given an action abstraction function α and a set of literals V, let $\sigma^I_{IS}(\alpha, V)$ denote the state abstraction function σ for IS^α such that for each agent $i \in Ag$, $\sigma.i(l) = \sigma.i(l')$ iff $\Pi(l) \cap V = \Pi(l') \cap V$ and $P_i^\alpha(l) = P_i^\alpha(l')$. So $\sigma^I_{IS}(\alpha, V)$ defines an interpreted system in which states agreeing on literals in V are collapsed onto a single abstract state as long as all agents have the same protocol in these states.

Definition 18 (Initial abstraction function). *Let IS be an interpreted system and φ be an ATL specification. The* initial abstraction function *for IS is the pair $\delta^I_{IS}(\varphi) = (\alpha^I_{IS}(A(\varphi)), \sigma^I_{IS}(\alpha^I_{IS}(A(\varphi)), \overline{\mathcal{V}(\varphi)}))$. $IS^{\delta^I_{IS}(\varphi)}$ denotes the* initial abstraction *of the interpreted system IS w.r.t. $\delta^I_{IS}(\varphi)$.*

In summary the initial abstraction function collapses all actions of agents not referred to in the specification; from this interpreted system all states with the same labelling and the same protocol are further collapsed.

We exemplify this definition in Section 5.

4.3 Refinement

As mentioned earlier, it may be that $IS^{\delta^I_{IS}(\varphi)} \models_3 \varphi = \mathrm{uu}$, i.e., the initial abstraction is not sufficient to determine the value of φ. In these cases we need to refine $\delta^I_{IS}(\varphi)$. Unlike the classic technique (Clarke et al. 2000), our refinement procedure focuses on the states for which the value of some subformula of φ is uu.

Let $\preceq$ be any linear order on formulas of ATL such that if ψ is a subformula of ψ', then $\psi \preceq \psi'$. Let $Sub(\varphi)$ stand for the set of all the subformulas of φ (including φ itself). An *evaluation* of φ on an interpreted system IS is a function $L : S \times Sub(\varphi) \to \{\mathrm{ff}, \mathrm{uu}, \mathrm{tt}\}$ such that for all $s \in S, \varphi' \in Sub(\varphi)$ we have $L(s, \varphi')$ is the result of the $M_{IS}, s \models_3 \varphi'$. Note that one can easily adjust the model checking algorithm presented in Proposition 17 to return the evaluation.

We now discuss the refinement procedure REFINE (see Algorithm 1) that takes as input an interpreted system IS, an abstraction function δ, a specification φ, an evaluation L of φ on IS^σ and returns an abstraction function δ'.

Firstly, REFINE computes the set Φ of subformulas φ' of φ s.t. for some s^δ s.t. there are at least two states $s, s' \in S$ s.t. $\sigma(s) = \sigma(s') = s^\delta$ we have $L(s^\delta, \varphi) = \mathrm{uu}$. If Φ is not empty, we take the smallest element ψ and split all the abstract states s^δ s.t. $L(s^\delta, \psi) = \mathrm{uu}$ by using the function SPLITS_STATES. If Φ is empty, we refine the action abstraction function.

The function SPLITS_STATES(IS, σ, S_{uu}) defines a state abstraction function σ' that agrees with σ on all the local states not in S_{uu}. For the remaining states, there are two possibilities. If there are two states $s, s' \in S_{\mathrm{uu}}$ and an agent i such that $\sigma.i(s.i) = \sigma.i(s'.i)$ and $\Pi(s.i) \neq \Pi(s'.i)$, then σ' splits the abstracted states into states corresponding to different labelling; otherwise σ' separates all the local states in S_{uu}.

The refinement procedure above focuses mostly on the state abstraction function, revising the action abstraction function only if no further updates to the state abstraction function are useful. This is optimised for ATL specifications of the form $\langle\langle\Gamma\rangle\rangle\varphi'$, where φ' does not contain a further $\langle\langle\rangle\rangle$ operator. Most concrete ATL specifications follow this pattern. Indeed,

Algorithm 1 The refinement procedure.

procedure REFINE(IS, $\delta = (\alpha, \sigma)$, φ, L)
  $MS^\delta \leftarrow \{s^\delta \in S^\delta \mid |\sigma^{-1}(\{s^\delta\})| > 1\}$
  $\Phi \leftarrow \{\varphi' \in Sub(\varphi) \mid \exists s^\delta \in MS^\delta . L(s^\delta, \varphi') = \mathrm{uu}\}$
  if Φ is not empty **then**
    $\alpha' \leftarrow \alpha$
    $S_{\mathrm{uu}} \leftarrow \{s \in S \mid L(\sigma(s), \min_{\preceq}(\Phi)) = \mathrm{uu}\}$
    $\sigma' \leftarrow$ SPLIT_STATES(IS, σ, S_{uu})
  else if $\alpha = \alpha^I_{IS}(A(\varphi))$ **then**
    $\alpha' \leftarrow \alpha^I_{IS}(Ag \setminus A(\varphi))$
    $\sigma' \leftarrow \sigma^I_{IS}(\alpha', \overline{\mathcal{V}(\varphi)})$
  else if $\alpha = \alpha^I_{IS}(Ag \setminus A(\varphi))$ **then**
    $\alpha' \leftarrow id$
    $\sigma' \leftarrow \sigma^I_{IS}(\alpha', \overline{\mathcal{V}(\varphi)})$
  else $\sigma' \leftarrow id$, $\alpha' \leftarrow id$
  return (α', σ')
procedure SPLIT_STATES(IS, σ, S_{uu})
  $split\_all \leftarrow \forall s, s' \in S_{\mathrm{uu}} . \forall i \in Ag . \sigma.i(s.i) = \sigma.i(s'.i) \Rightarrow \Pi(s.i) = \Pi(s'.i)$
  for $i \in Ag, l_i \in L_i$ **do**
    if $\exists s \in S_{\mathrm{uu}}\ s.i = l_i$ **then**
      if $split\_all$ **then** $\sigma'.i(l_i) \leftarrow l_i$
      else $\sigma'.i(l_i) \leftarrow (\sigma.i(l_i), \Pi(l_i))$
    else $\sigma'.i(l_i) \leftarrow \sigma.i(l_i))$
  return σ'

when evaluating whether $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle\varphi' = \mathrm{tt}$ the labels of the actions for the agents not in Γ are irrelevant. This is why the initial abstraction collapses them. Conversely, when evaluating $M_{IS}, s \models_3 \langle\langle\Gamma\rangle\rangle\varphi' = \mathrm{ff}$ the actions that matter are those of $Ag \setminus \Gamma$; this is reflected in the first refinement of the action abstraction function. If none of the above is effective, we use the identity.

4.4 Model Checking with Refinement

In view of the constructions above, we can now give the overall model checking algorithm for verifying an interpreted system IS against an ATL specification φ as the procedure VERIFY at Algorithm 2. The procedure calls the function EVALUATE which labels the structure, as discussed in Proposition 17, and returns the appropriate evaluation function.

Algorithm 2 The verification procedure.

procedure VERIFY(IS, φ)
  $\delta \leftarrow \delta^I_{IS}(\varphi)$
  while $\delta \neq (id, id)$ **do**
    $L \leftarrow$EVALUATE(IS^δ, φ)
    if $\exists s \in I . L(\sigma(s), \varphi) = \mathrm{ff}$ **then return** ff
    else if $\forall s \in I . L(\sigma(s), \varphi) = \mathrm{tt}$ **then return** tt
    else $\delta \leftarrow$REFINE(IS, δ, L)
  return uu

The algorithm proposed first constructs the initial abstraction δ^I_{IS}. A first check is made for the truth value of φ on δ^I_{IS}. If one abstract state corresponding to a concrete initial state does not satisfy φ, by Theorem 12 we deduce that the specification does not hold on the concrete interpreted system. Similarly if φ is satisfied in all abstract states corresponding to all concrete initial states, we deduce that the specification is satisfied on the original model. If neither of these cases holds, we refine the initial abstraction by following the REFINE procedure above and repeat the evaluation tests. Note that the procedure always terminates; in the worst case it will attempt to perform the verification on the original, concrete model.

While in the worst case the procedure is worse than a single initial check on the concrete model, it is possible that either the initial abstraction or one of its refinements will determine the value of the specifications. The reduced models may be much smaller than the original concrete model, which may be too large to be verified even by using symbolic techniques. Even if the concrete model can be handled by concrete techniques, the procedure above offers an attractive methodology for reducing the time and memory footprint of the verification step.

5 Verification of Simbridge

In this section we exemplify the methodology above on *Simbridge*, a simple two player trick-taking game. We encode the game as an interpreted system and verify a specification by using the abstraction technique presented above.

The game is played with a deck of 26 cards (13 red and 13 black). Cards are numbered from 1 to 13. The goal of each player is to win the most number of rounds, or *tricks*.

Each player is initially given 13 cards by the croupier. The game is played in 13 rounds. The first player is called the *lead player* of the first round; in all the other rounds the lead player is the player who won the previous trick. In each round the lead player plays any card from his hand; the other player has to respond by playing a card in the same colour. The player who played the card with the higher number wins the trick. If a player cannot play a card in the colour of the card played by the first player, he can respond by playing any card and he loses the trick. If a player has a card in the required colour but plays a card in a different colour, he loses for cheating.

5.1 Game encoding

We model the game as an interpreted system IS comprising four agents: Croupier, Player 1, Player 2 and Scorekeeper.

Croupier. The game begins with Croupier distributing the cards to the players as follows. For each card, starting from 1 Red, Croupier assigns the card to a player by either using the action 1 or 2. If a player has already been given 13 cards, all remaining cards are given, one by one, to the other player. After distributing all the cards, Croupier plays the action ϵ for the rest of the game.

Croupier's 14^2 local states are tuples where each local state (k, l) represents the situation where Player 1 was given k cards and Player 2 was given l cards. The Croupier's protocol is defined as follows: in a state (k, l) Croupier can perform action 1 (resp. 2) if $k < 13$ (resp. $l < 13$). If action 1 (resp. 2) is performed, Croupier moves to the state $(k + 1, l)$

Structure	Actions	States
Original structure (IS)	2187	10^{27}
Initial abstraction (IS^{σ})	3	10^{3}
Refinement (IS^{R})	3	10^{8}

Table 1: Estimated sizes of the interpreted systems from Section 5.

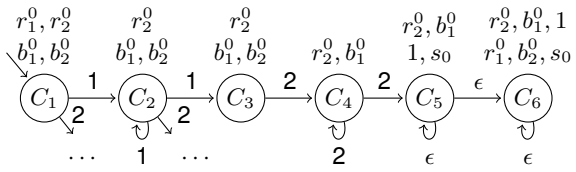


Figure 1: A fragment M^{δ}_{IS}, the model associated with the initial abstraction.

(resp. $(k, l+1)$) regardless on the other agents' actions. Once Croupier is in the state $(13, 13)$ he loops in this state by performing the action ϵ. Note that in this formalisation Croupier does not remember the cards' distribution.

Players. The players are symmetrical and follow the same formalisation. We represent the player's local states as (lp, pc, H), where H is the set of cards in the player's hand, $lp \in \{1, 2, C\}$ and $pc \in \{\mathrm{Red}_1, \ldots, \mathrm{Black}_{13}\} \cup \{\emptyset\}$. If $lp = C$ then the game is in the phase when the cards are being distributed and pc represents the previously distributed card. If $lp \neq C$, then tricks are being played, lp represents the current lead player and pc encodes the first card played in the current round. It can be computed that there are approximately 2^{25} combinations; so each player has approximately 2^{31} possible local states.

The set of local actions contains ϵ, which players use when it is not their turn, and the actions $\mathrm{Red}_1, \ldots, \mathrm{Black}_{13}$ that the players use to play a card when it is their turn. We assume that the protocol only insists on playing cards in the players' hands, i.e., players may not follow the colour played.

The transitions are defined as expected. Players begin with $lp = C$ and update their state while cards are being distributed. Then players start playing cards from their hands and update their state accordingly.

Scorekeeper. The scorekeeper's local states contain the states $Cheated_1$, $Cheated_2$ to represent situations where some of the players have cheated during the game, and tuples of the form (lp, pc, SG) where $lp \in \{1, 2, C\}$ represents the lead player, $pc \in \{\mathrm{Red}_1, \ldots, \mathrm{Black}_{13}\} \cup \{\emptyset\}$ represents the first card played in the current round, if $lp \neq C$ and the last card distributed if $lp = C$. SG is a tuple $SG = (S, R1, R2, B1, B2)$ representing the *state of the game* after each round where no player has cheated. Ri represents the number of red cards in the hand of Player i; Bi represents the number of black cards in the hand of Player i; S represents the number of tricks won by Player 2. Some of the local states of Scorekeeper are not reachable during a game (e.g., states where $R1 + R2 > 13$). By considering this we can estimate that there are approximately $2.5 \times 10^6 \approx 2^{21}$ states possible local states for Scorekeeper.

Scorekeeper starts in the state $(C, \emptyset, (0, 0, 0, 0, 0))$ and then updates his states on the basis of the actions of Croupier and the players.

Global states and variables. We can define the set of global states for the system by considering tuples containing the local states for the agents defined above. The initial states for the system are those obtained by considering the initial local states of the agents defined above. We can ascertain there are at least $196 \cdot 2^{31} \cdot 2^{31} \cdot 2^{21} \approx 10^{27}$ global states defined as above. This constitutes a size that would normally not be verifiable even by an advanced, symbolic model checker.

We use the following propositional variables: 1, 2, s^i, r_1^i, r_2^i, b_1^i, b_2^i for $i \in \{0, \ldots, 13\}$. Their truth value depends on the Scorekeeper's state. So if in a global state the Scorekeeper is in the local state $(lp, pc, (S, R1, R2, B1, B2))$, then the variables $lp, pc, s^S, r_1^{R1}, r_2^{R2}, b_1^{B1}, b_2^{B2}$ hold at that state. The variable ch_i holds at any global state where Scorekeeper's local state includes $Cheated_i$.

5.2 A specification of interest

From the description above it is easy to see that Croupier can ensure that Player 1 wins the game. Croupier can for instance give all the red cards to Player 1; since Player 1 starts the game, Player 2 will never win a trick.

For convenience we use the following Boolean formulas.

$$\begin{aligned} GameOver &= ((1 \vee 2) \wedge r_1^0 \wedge r_2^0 \wedge b_1^0 \wedge b_2^0) \vee ch_1 \vee ch_2 \\ Wins_1 &= s^0 \vee \cdots \vee s^6 \vee ch_2 \end{aligned}$$

$GameOver$ represents that all rounds have been played, i.e., either the cards have been distributed and no player has any card left, or a player has cheated. $Wins_1$ encodes a situation where Player 1 has won the game, i.e., either Player 2 has cheated or he has won fewer than seven tricks.

Consider the following specifications:

$$\Phi_1 = \langle\!\langle Croupier \rangle\!\rangle G(\neg GameOver \vee Wins_1)$$
$$\Phi_2 = \langle\!\langle Croupier \rangle\!\rangle F(GameOver \wedge Wins_1)$$

The first formula states that Croupier has a strategy to ensure that forever in the future either Player 1 wins or the game does not end. The second states that Croupier can force the system to reach a final state where Player 1 has won the game. We now use the abstraction technique to verify the system against both specifications. As we show below, the initial abstraction is sufficient to verify Φ_1, whereas to verify Φ_2 we need to employ the refinement procedure.

5.3 Abstraction for Φ_1

Consider the initial abstraction function $\delta = (\alpha, \sigma)$ for Φ_1. Since $A(\Phi_1) = \{Croupier\}$, the initial action abstraction α collapses the actions of all the agents except $Croupier$. This entails that the set of abstract actions in IS^{δ} is $ACT^{\delta} = \{(\epsilon, \epsilon, \epsilon, \epsilon), (\epsilon, 1, \epsilon, \epsilon), (\epsilon, 2, \epsilon, \epsilon)\}$.

The initial abstraction function is built by considering the variables $\mathcal{V}(\Phi_1) = \{1, 2, r_1^0, r_2^0, b_1^0, b_2^0, ch_1, ch_2, s^0, \ldots, s^6\}$. We do not present the whole abstract model, but only the fraction that is needed to establish the value of Φ_1.

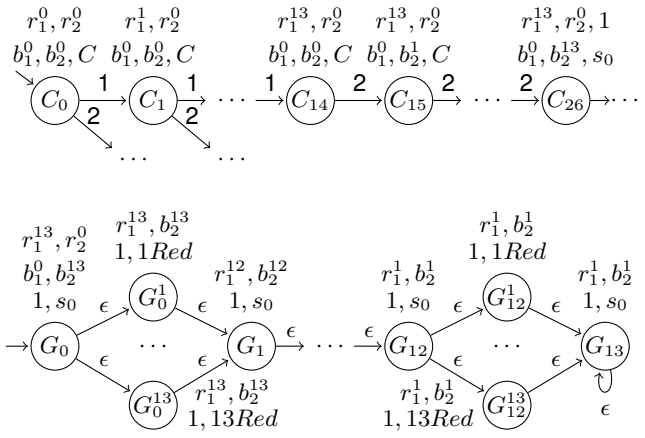


Figure 2: A fragment of the refined structure. For readability, some labels are omited. G_0 and C_{26} denote the same state.

We first describe the part of the structure representing the card distribution. Figure 1 depicts five states of this structure. C_1 is the initial state of M_{IS}^{δ} corresponding to exactly one state in M_{IS}. The abstract state C_2 corresponds to the states in M_{IS} representing situations in which Player 1 received between 1 and 12 red cards and Player 2 did not receive any. State C_3 corresponds to exactly the state in M_{IS} in which Player 1 received 13 red cards and Player 2 did not receive any. Note that the only difference between these states is that Croupier's protocol in C_3 does not allow for 1 to be played. States C_4 and C_5 represent configurations in which Player 1 has all the red cards, and Player 2 received either between 1 and 12 (C_4), or 13 (C_5) black cards.

Consider $F = \{f_{Croupier}\}$ where $f_{Croupier}$ is the following strategy for Croupier: select action 1 in states C_1 and C_2, play action 2 in states C_3 and C_4, and ϵ in all the other states. Then, in $out(C_1, F)$ we have $C_1C_2^{\omega}$, $C_1C_2^+C_3C_4^{\omega}, C_1C_2^+C_3C_4^+C_5^{\omega}, C_1C_2^+C_3C_4^+C_5^+C_6^{\omega}$. It can be easily verified that all the paths satisfy $G\neg(GameOver \wedge Wins_2)$.

Therefore, we have that $IS^{\delta} \models_3 \Phi_1 = \mathrm{tt}$; so by Theorems 5 and 12 we deduce that $IS \models_2 \Phi_1$. The system IS^{δ} is much smaller than IS; its number of states can be bounded by the number of possible valuations of variables from $\mathcal{V}(\Phi_1)$ multiplied by the number of possible protocols; this results in $242 \cdot 4 \approx 10^3$ states. This is a size that can easily be verified on a model checker.

5.4 Abstraction for Φ_2

Since $\mathcal{V}(\Phi_2) = \mathcal{V}(\Phi_1)$ the initial abstraction function δ is the same as the δ for Φ_1 (see Figure 1). It can be checked that Croupier does not have a strategy to force $GameOver$. For example consider $F = \{f_{Croupier}\}$ where $f_{Croupier}$ is a strategy according to which action 1 is selected in C_1 and C_2. Then we have that $C_1C_2^{\omega} \in out(C_1, F)$, but $GameOver$ never holds on this path.

Note that the subformulas of Φ_2 are $\langle\!\langle Croupier \rangle\!\rangle F(GameOver \wedge Wins_1)$, $GameOver \wedge Wins_1$, $GameOver$, $Wins_1$, and all the variables in $GameOver$ and $Wins_1$. Of these, the only subformula that is undefined in some of the states of the abstracted structure is Φ_2. Φ_2 is not undefined only where $GameOver$ is true. It follows that we cannot establish whether Φ_2 holds on the system.

We now therefore execute the refinement procedure REFINE. This returns the same action abstraction function $\alpha' = \alpha$ and the state abstraction function σ' such that $\sigma'(s) = \sigma(s)$ for s satisfying $GameOver$ and $\sigma'(s) = (\sigma(s), \Pi(s))$ for all the remaining states. Let IS^R be the abstract system obtained by refining the initial abstraction by means of (α', σ').

Like IS^{δ}, the system IS^R has only three global actions; its number of states can be bounded by $30079730 \cdot 4 \approx 10^8$. A fragment of this structure is presented in Figure 2. This is a size that can easily be verified by a modern model checker.

Consider now $F = \{f_{Croupier}\}$ where $f_{Croupier}$ is a strategy for Croupier that selects action 1 in states $C_0, \ldots, C_{13}$, action 2 in states $C_{14}, \ldots, C_{25}$ and ϵ at other points. Figure 2 shows that $out(C_0, F)$ consists of the paths of the form $C_0 \ldots C_{26}G_0G_0' \ldots G_{12}G_{12}'G_{13^{\omega}}$, where $G_i' = (G_i^1 + \cdots + G_i^{13})$. It can be checked that all these paths satisfy $F(GameOver \wedge Wins_1)$. It follows that Φ_2 is satisfied in the refinement and therefore, by Theorem 12, we have that it also holds on the concrete model IS.

6 Conclusions

Model checking multiagent systems is now a well-established area of research with results ranging from theoretical investigations to sophisticated implementations. While two notable implementations, jMOCHA and MCMAS, support ATL as a specification language, neither provides abstraction functionalities. Yet, it is well known that abstraction is a fundamental technique to apply when verifying concrete systems.

Abstraction, including refinement, has been developed for epistemic specifications (Cohen et al. 2009) and, of course, temporal ones (Clarke, Grumberg, and Long 1994; Clarke et al. 2000). But in the context of ATL specifications, most of the work has so far focused on the theoretical aspects of the model checking problem, including realistic settings with incomplete information. Yet, to use ATL specifications in practice there is a compelling need for powerful abstraction techniques to reduce the state-spaces of concrete systems. In this paper we intended to make a contribution in this direction.

Our long-term research objective is to verify concrete systems via under-/over-approximations. We have laid the foundations of this by defining a novel three-valued semantics for ATL. We have shown this to be a conservative extension of the classical two-valued one. We have then provided a constructive definition to derive a first, coarse, yet potentially very effective abstraction, and a refinement algorithm that splits states potentially resolving uncertainty in the truth value of the specifications being checked. Applying the procedure to a card game of incomplete information. That showed the first abstraction reduced the state-space by a factor of over 10^{24} to a well-manageable state-space of approximately 10^3. The first specification of interest was satisfied on the first abstraction; the second required us to execute the refinement

algorithm which resulted in a model of approximately 10^8 states. The refined model enabled us to decide on the truth of the second specification; also note that state-spaces in the region of 10^8 can be verified by a model checker for ATL such as MCMAS.

References

Alur, R.; de Alfaro, L.; Grosu, R.; Henzinger, T. A.; Kang, M.; Kirsch, C. M.; Majumdar, R.; Mang, F.; and Wang, B.-Y. 2001. jMocha: A model checking tool that exploits design structure. In *Proceedings of the 23rd International Conference on Software Engineering (ICSE01)*, 835–836. IEEE.

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713.

Belnap, N., and Perloff, M. 1990. Seeing to it that: A canonical form for agentives. In *Knowledge Representation and Defeasible Reasoning*, volume 5 of *Studies in Cognitive Systems*, 167–190. Springer.

Bulling, N.; Dix, J.; and Chesñevar, C. I. 2008. Modelling coalitions: ATL + argumentation. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS08)*, 681–688. IFAAMAS.

Chatterjee, K.; Henzinger, T.; and Piterman, N. 2007. Strategy logic. In *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR07)*, volume 4703 of *Lecture Notes in Computer Science*, 59–73. Springer.

Clarke, E. M.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2000. Counterexample-guided abstraction refinement. In *Proceedings of the 12th International Conference on Computer Aided Verification (CAV00)*, volume 1855 of *Lecture Notes in Computer Science*, 154–169. Springer.

Clarke, E. M.; Grumberg, O.; and Long, D. 1994. Model checking and abstractions. *ACM Transactions on Programming Languages and Systems* 16(5):1512–1542.

Cohen, M.; Dam, M.; Lomuscio, A.; and Russo, F. 2009. Abstraction in model checking multi-agent systems. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS09)*, 945–952. IFAAMAS Press.

Dima, C., and Tiplea, F. L. 2011. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *arXiv preprint arXiv:1102.4225*.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning about Knowledge*. MIT Press.

Jamroga, W., and Ågotnes, T. 2007. Modular interpreted systems. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS07)*, 131–138. IFAAMAS.

Köster, M., and Lohmann, P. 2012. Abstraction for model checking modular interpreted systems over ATL. In *Programming Multi-Agent Systems*, volume 7217 of *Lecture Notes in Computer Science*. Springer. 95–113.

Lomuscio, A., and Raimondi, F. 2006a. The complexity of model checking concurrent programs against CTLK specifications. In *DALT06*, volume 4327 of *Lecture Notes in Computer Science*, 29–42.

Lomuscio, A., and Raimondi, F. 2006b. Model checking knowledge, strategies, and games in multi-agent systems. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS06)*, 161–168. ACM Press.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2009. MCMAS: A model checker for the verification of multi-agent systems. In *Proceedings of the 21th International Conference on Computer Aided Verification (CAV09)*, volume 5643 of *Lecture Notes in Computer Science*, 682–688. Springer.

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. Y. 2012. What makes ATL* decidable? A decidable fragment of strategy logic. In *Proceedings of the 23th International Conference on Concurrency Theory (CONCUR12)*, volume 7454 of *Lecture Notes in Computer Science*. Springer. 193–208.

Schobbens, P.-Y. 2004. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science* 85(2):82–93.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

How to Progress Beliefs in Continuous Domains

Vaishak Belle and **Hector J. Levesque**

Dept. of Computer Science
University of Toronto
Toronto, Ontario M5S 3H5, Canada
{vaishak, hector}@cs.toronto.edu

Abstract

When Lin and Reiter introduced the progression of basic action theories in the situation calculus, they were essentially motivated by long-lived robotic agents functioning over thousands of actions. However, their account does not deal with probabilistic uncertainty about the initial situation nor with effector or sensor noise, as often needed in robotic applications. In this paper, we obtain results on how to progress continuous degrees of belief against continuous effector and sensor noise in a semantically correct fashion. Most significantly, and perhaps surprisingly, we identify conditions under which our account is not only as efficient as the filtering mechanisms commonly used in robotics, but considerably more general.

Introduction

Reiter's (2001) reconsideration of the situation calculus (McCarthy and Hayes 1969) has proven enormously useful for the design of logical agents, essentially paving the way for *cognitive robotics* (Lakemeyer and Levesque 2007). Among other things, it incorporates a simple monotonic solution to the frame problem, leading Reiter to define the notion of *regression* for basic action theories (Waldinger 1977). But for long-lived agents like robots, Lin and Reiter (1997) argue that the notion of *progression*, that of continually updating the current view of the state of the world, is perhaps better suited. They show that progression is always second-order definable, and in general, it appears that second-order logic is unavoidable (Vassos and Levesque 2008). However, Lin and Reiter also identify some first-order definable cases by syntactically restricting situation calculus basic action theories, and since then, a number of other special cases have been studied (Liu and Lakemeyer 2009).

While Lin and Reiter intended their work to be used on robots, one criticism leveled at their work, and indeed at much of the work in cognitive robotics, is that the theory is far removed from the kind of continuous uncertainty and noise seen in typical robotic applications. What exactly filtering mechanisms (such as Kalman filters (Thrun, Burgard, and Fox 2005)) have to do with Lin and Reiter's progression has gone unanswered, although it has long been suspected that the two are related.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

This paper remedies this situation. Building on a recent extension to the situation calculus for dealing with continuous uncertainty by Belle and Levesque (2013a) (henceforth BL), we investigate progression in continuous domains. In this paper, we make the following contributions:

1. We introduce a property of basic action theories called *invertibility*, closely related to invertible functions in real analysis (Trench 2003). We identify syntactic restrictions on basic action theories that guarantee invertibility.
2. For our central result, we show a first-order *progression* of continuous degrees of belief against continuous noise in effectors and sensors for action theories that are invertible.
3. Finally, we prove that this account of progression is *efficient*. Perhaps surprisingly, under the additional assumption of context-completeness (Liu and Levesque 2005), it is as efficient as commonly used filtering mechanisms in the robotics literature, while considerably more general.

For the structure of this paper, we begin by introducing the formalism, followed by results along the lines of the above three points and conclude after discussion and related work.

The Situation Calculus

The language $\mathcal{L}$ of the situation calculus is a many-sorted dialect of the predicate calculus, with sorts for *physical actions*, *sensing actions, situations* and *objects*, which is a catch-all sort for everything else. A situation represents a history as a sequence of actions. A set of initial situations correspond to the ways the world might be before any action; successor situations are obtained as a result of doing actions. We use α to denote terms of the action sort, and so the term $do(\alpha, s)$ denotes the unique situation obtained on doing α in s. The term $do(\vec{\alpha}, s)$, where $\vec{\alpha}$ is the sequence $\alpha_1 \cdots \alpha_n$, abbreviates $do(\alpha_n, do(\ldots, do(\alpha_1, s)\ldots))$. Initial situations, then, are those without a predecessor:

$$Init(s) \doteq \neg\exists a, s'.\ s = do(a, s').$$

We let the constant S_0 denote the actual initial situation and use the variable ι to range over initial situations only.

To reason about changing properties, $\mathcal{L}$ is assumed to include *fluents*, whose last argument is always a situation term. Without loss of generality, we restrict ourselves to functional

fluents. In addition, we assume that $f_1, \ldots, f_k$ are all the fluents in $\mathcal{L}$, and that these take no arguments other than a single situation term.[1] Note that this is not a propositional theory in that we allow the *values* of the fluents $f_1, \ldots, f_k$ to range over any set. As we are primarily interested in continuous domains, we will make the simplifying assumption that fluents are all real-valued.

Following (Reiter 2001), we call a formula ϕ *uniform* in a situation term σ if σ is the only situation term in ϕ and *rooted* in σ if every situation term in ϕ is of the form $do(\vec{\alpha}, \sigma)$ for some action sequence $\vec{\alpha}$.[2] We often suppress the situation argument in a formula ϕ, or use a distinguished variable *now*. Either way, $\phi[\sigma]$ is used to denote the formula with that variable replaced by the situation term σ.

Basic Action Theory

Again following Reiter, we model dynamic domains by means of a set of sentences $\mathcal{D}$ called a *basic action theory*, consisting of:[3]

- sentences $\mathcal{D}_0$ describing what is true initially;
- precondition axioms describing the executability of actions using a distinguished symbol *Poss*;[4]
- successor state axioms that describe the change to fluents after actions, of the form $f(do(a, s)) = u \equiv \gamma_f(a, u)[s]$;
- domain-agnostic foundational axioms and the unique name axioms for actions, the details of which need not concern us here (Reiter 2001).

We often lump the components in $\mathcal{D}$ other than $\mathcal{D}_0$ as Σ. Properties of the formalism follow as entailments of $\mathcal{D}$, for which standard Tarskian models suffice. We will however be assuming that models assign the usual interpretations to $=, <, >, 0, 1, +, \times, /, -, e, \pi$ and x^y (exponentiation).

The language $\mathcal{L}$ used in this paper is the situation calculus as characterized in (Reiter 2001). Nonetheless, it is very useful to have certain abbreviations that will macro expand to $\mathcal{L}$-formulas. These abbreviations are used as logical terms, that is, as arguments to functions or predicates. If E is such an expression, its expansion is characterized by a definition of the form $E = u \doteq \phi(u)$, where u is a variable, and $\phi(u)$ is an $\mathcal{L}$-formula with u free. This should be interpreted as follows. Let $\rho(u)$ be any atomic formula with u free. Then the expression $\rho(E)$ should be understood as standing for the formula $\exists u(\phi(u) \wedge \rho(u))$. To see this in action, consider the definition of the *conditional-if* expression: $u = \text{If } \exists z\phi \text{ Then } t_1 \text{ Else } t_2$ defined as $(\exists z\phi \supset \forall z(\phi \supset u = t_1)) \wedge (\neg\exists z\phi \supset u = t_2)$. This allows us to write, for example,

$$P(a, \text{If } Q(f, 3) \text{ Then } 4 \text{ Else } 5, b)$$

meaning $(Q(f, 3) \supset P(a, 4, b)) \wedge (\neg Q(f, 3) \supset P(a, 5, b))$. As a consequence of this convention, we will be able to write Reiter's successor state axioms as equality-expressions, and use the RHS as terms than can be substituted in formulas.

For example, imagine a robot moving in a 2-dimensional plane, parallel to a wall, as shown in Figure 1. Let its coordinate be given by (h, v), where h and v are real-valued fluents. Consider an action $mv(z)$ that moves the robot away from the X-axis by z units. Its effect on v might be described by:

$$v(do(a, s)) = u \equiv \exists z(a = mv(z) \wedge u = v(s) + z) \vee u = v(s) \wedge \neg\exists z(a = mv(z)).$$

This says that $mv(z)$ is the only action affecting v, thereby incorporating a solution to the frame problem (Reiter 2001). We would now equivalently write this as:

$$v(do(a, s)) = \text{If } \exists z(a = mv(z)) \text{ Then } v(s) + z \text{ Else } v(s). \quad (1)$$

Thus, successor state axioms axioms are henceforth assumed to be of the form $f(do(a, s)) = E_f(a)[s]$.

We will also use expressions $\int_x t$ and $\sum_x t$ for integration and summation, which are abbreviations defined using second-order logic in BL and (Bacchus, Halpern, and Levesque 1999). Finally, we will often refer to common density functions, such as $\mathcal{N}$ and $\mathcal{U}$ (Gaussian and uniform respectively), which are simply abbreviations for $\mathcal{L}$-terms built from arithmetic symbols and reals.

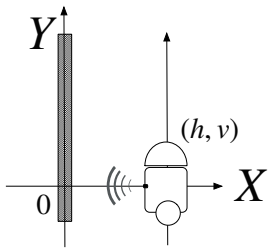


Figure 1: Robot moving parallel to the wall.

Noise and Likelihood

Building on (Bacchus, Halpern, and Levesque 1999), our treatment of noisy acting and sensing will benefit from Reiter's solution to the frame problem. It will involve a new distinguished symbol l for likelihood, where the modeler provides l-axioms in $\mathcal{D}$, one per action $A(\vec{x})$, of the form:

$$l(A(\vec{x}), s) = \Upsilon_A(\vec{x})[s].$$

Let us illustrate its usage with noisy sensors. (Noisy actions are discussed later.) The concern here is that the value read from a sensor may differ from the true value, but in some reasonable way. Imagine a sonar sensor aimed at the wall, and so its reading would correspond to the true value of fluent h. If noisy, one might assume that the likelihood of a reading z is obtained from a normal curve whose mean is h. We then say it has additive Gaussian noise, and this is modeled as:

$$l(sonar(z), s) = \mathcal{N}(z; h, 4)[s] \quad (2)$$

[1] See BL for a discussion of this particular limitation.

[2] Formulas that are rooted in σ are about σ and its future. More general formulas about the future of a situation are considered in (Vassos and Levesque 2008), but not here.

[3] Free variables, here and everywhere else, are assumed to be implicitly quantified from the outside.

[4] For simplicity, we will say no more about *Poss* here, and assume that actions are always executable.

which gives the sensor's reading a variance of 4. Note that the l-axioms are more expressive than those typically seen in probabilistic formalisms, and can be context-dependent. For example, to model a sensor with systematic bias at subzero temperatures, we might have

$$l(sonar(z), s) = \text{If } temp(s) > 0 \text{ Then } \mathcal{N}(z; h, 1) \text{ Else } \mathcal{N}(z; h+2, 1).$$

Degrees of Belief

Probabilistic uncertainty about the initial situation and changing degrees of belief are captured using a single distinguished symbol p. The p fluent can be seen as a numeric variant of the K fluent in the Scherl and Levesque scheme (2003). Intuitively, $p(s', s)$ gives us the probabilistic *density* that the agent attributes to s' when at s. In fully discrete domains, p is the *mass* given to s'.

As part of $\mathcal{D}_0$, the modeler would provide a characterization of the density attributed by the agent to the initial situations. For example,

$$p(\iota, S_0) = \mathcal{U}(h; 2, 12) \times \mathcal{N}(v; 0, 1)[\iota] \tag{3}$$

says that the agent does not know the initial values of h and v, but thinks of them as drawn independently from a uniform distribution on [2, 12], and from a standard normal. Since p is just like any other fluent, the framework is more expressive than many probabilistic formalisms. For example,

$$\begin{array}{c} \forall \iota (p(\iota, S_0) = \mathcal{U}(h; 2, 3)[\iota]) \\ \vee \\ \forall \iota (p(\iota, S_0) = \mathcal{U}(h; 10, 20)[\iota]) \end{array} \tag{4}$$

says that the agent takes h to be uniformly distributed on [2,3] or on [10,20], but without saying which.

To give p the intended properties, the following nonnegative constraint is assumed to be included in $\mathcal{D}$ (Bacchus, Halpern, and Levesque 1999):

$$\forall \iota, s.\ p(s, \iota) \geq 0 \wedge (p(s, \iota) > 0 \supset Init(s)). \tag{P1}$$

This says that p values are never negative, and that they are non-zero only for initial situations. While this is a stipulation about initial situations, the following successor state axiom for p ensures that the nonnegative constraint holds everywhere:

$$p(s', do(a, s)) = \text{If } \exists s''.\ s' = do(a, s'') \wedge Poss(a, s'') \text{ Then } p(s'', s) \times l(a, s'') \text{ Else } 0 \tag{P2}$$

That is, the density of a situation s' relative to $do(a, s)$ is the density of its predecessor s'' times the likelihood of a contingent on the successful execution of a at s''. One consequence of (P1) and (P2) is that $(p(s', s) > 0)$ will be true only when s' and s share the same history of actions.

As discussed in BL, to deal with continuity, the following axiom explicating a precise space of initial situations is needed (Baker 1989; Levesque, Pirri, and Reiter 1998):

$$[\forall \vec{x} \exists \iota \bigwedge f_j(\iota) = x_j] \wedge [\forall \iota, \iota'. \bigwedge f_j(\iota) = f_j(\iota') \supset \iota = \iota'] \tag{P3}$$

This states that for every combination of fluents values, there is a unique initial state where the fluents have these values.

With l and p axioms specified in $\mathcal{D}$, the *degree of belief* in a formula is defined as follows:[5]

Definition 1: Let ϕ be a formula and σ be a situation term, both rooted in *now*. Then $Bel(\phi, \sigma[S_0])$ is defined as

$$\frac{1}{\eta} \int_{\vec{x}} \text{If } \exists \iota \bigwedge f_j(\iota) = x_j \wedge \phi[\sigma[\iota]] \text{ Then } p(\sigma[\iota], \sigma[S_0]) \text{ Else } 0$$

where η is the *normalization* factor and is the same expression as the numerator but with ϕ replaced by *true*.

This says that belief is the result of integrating the density function p over all possible initial values for the real-valued fluents, and hence by (P3), over all possible initial situations. Note that this definition of belief is restricted to cases where the situation term is rooted in S_0.

To summarize, all that was needed to reason about degrees of belief and continuity in the situation calculus were the following components:

- $\mathcal{D}_0$ as usual (possibly mentioning p);
- the action components and foundational axioms as before, including the l-axioms, and p-constraints (P1), (P2) and (P3) (lumped together as Σ);
- abbreviations for *Bel*, integrals, and summation.

It is easy to see that this framework generalizes the categorical knowledge treatment of (Scherl and Levesque 2003). It can also be shown that the belief change mechanism exhibited by the framework subsumes Bayesisan conditioning (Pearl 1988). Finally, the scheme is also applicable to discrete domains: for a discrete fluent f, one would simply replace $\int_x$ with $\sum_x$ ranging over the possible values of f.

Noisy actions

The account of belief in BL left noisy actions for future work. The idea behind noisy actions is that an agent might attempt a physical move of 3 units, say, but as a result of the limited accuracy of effectors, actually move 3.094 units. Thus, unlike sensors, where the reading is nondeterministic, *observable,* but does not affect fluents, the outcome of noisy actions is nondeterministic, *unobservable* and changes fluent properties. Of course, when attempting to move 3 units, the agent knows that an actual move by 3.094 units is much more likely than 30.94 units.

We suggest a simple specification of noisy actions where instead of an action like $mv(x)$, we use an action $mv'(x, y)$ where x is the intended motion (3 units) known to the agent and y is the actual motion (perhaps 3.094 units) unknown to the agent. The successor state axiom for h would be written to reflect the fact that its value is changed by the y value. Unlike noise-free actions where the likelihood is always 1, noisy actions would have non-trivial l-axioms like this one:

$$l(mv'(x, y), s) = \mathcal{N}(y; x, 1 + wet)[s].$$

[5]This is a minor reworking of the definition given in BL, but as it turns out, it has the advantage of working seamlessly with noisy actions as well.

This is a context-dependent axiom that says that the actual value moved is normally distributed around the intended value, but with a variance that depends on the *wet* fluent (a measure of how wet the floor is).

To capture the fact that the agent need not know the actual amount moved, we use *fluents arguments* to the action. For example, to find out the degree of belief in ϕ after a noisy move action, instead of asking for the value of

$$Bel(\phi, do(mv'(3, 3.094), S_0)),$$

we instead ask for the value of something like

$$Bel(\phi, do(mv'(3, g(S_0)), S_0)),$$

where g is a fluent whose value is not known, that stands for the actual amount moved. The fluent g here represents nature's choice of a value, and as it is unaffected by any action, its successor state axiom should be $g(do(a, s)) = g(s)$. The fluent will have different values in different initial states, of course, and so the likelihood of the corresponding mv' action will also vary, as will the density given to the successor situation by (P2). The definition of belief will then integrate over all possible values of g as appropriate.

In general, a fluent can be used like this to handle any argument to an action whose value is not known to the agent. We believe this account is much simpler than the version of noisy actions given in (Bacchus, Halpern, and Levesque 1999) which relied on Golog.

Invertible Action Theories

What we are after is an account of progression for belief in the presence of continuous noise. However, subtleties arise with the p fluent even in simple domains. For example, imagine a noise-free action *towall*(z) that moves the robot towards the wall but stopping when the robot hits the wall:

$$h(do(a, s)) = \text{If } \exists z(a = towall(z)) \text{ Then } \max(0, h(s) - z) \text{ Else } h(s). \quad (5)$$

If the robot were to begin with (3) and perform the action *towall*(4), beliefs about the new value of h become much more complex. Roughly, those points where $h \in [2, 4]$ initially are mapped to a single point $h = 0$ that should then obtain a probability *mass* of .2, while the other points retain their initial *density* of .1. In effect, a probability density on h is transformed into a mixed density / distribution (and the (P3) assumption no longer holds). BL deal with this issue using *regression*: beliefs are regressed to the initial state where (P3) does hold, and all the actual belief calculations can be done in the initial state (Belle and Levesque 2013b).

In this paper, we develop a logical theory of progression for basic action theories where such mixed distributions do not arise. In this section, we provide a simple definition to support this, and then discuss general syntactic restrictions that satisfy this requirement.

Definition 2: Given a basic action theory $\mathcal{D}$, a fluent f is said to be *invertible* if there is an expression $H_f(a)$ uniform in *now* such that $\mathcal{D} \models f(s) = H_f(a)[do(a, s)]$. We say that $\mathcal{D}$ is invertible if every fluent in the theory is invertible.

Intuitively, a fluent is invertible when we can find a dual formulation of its successor state axiom, that is, where we can characterize the *predecessor* value of a fluent in terms of its current value.

There are three syntactic conditions on a basic action theory $\mathcal{D}$ that are sufficient to guarantee its invertibility:

i. There is an ordering on fluents such that all the fluents that appear in $E_f(a)$ other than f are earlier in the ordering.

ii. Any situation term in $E_f(a)$ appears as an argument to one of the fluents.

iii. The mapping from the value of $f(s)$ to the value of $f(do(a, s))$ given by $E_f(a)$ is bijective (Trench 2003). (This is understood in the usual set-theoretic sense.)

Before considering some examples, here is the result:

Theorem 3: *If a basic action theory satisfies (i), (ii) and (iii) above, then it is invertible.*

Proof sketch: The proof is by induction on the ordering given by (i). By (iii), we can take $f(do(a, s)) = E_f(a)[s]$ and solve for $f(s)$, obtaining an equation $f(s) = H$, where H mentions $f(do(a, s))$ and possibly other fluents $f'(s)$ that appear earlier in the ordering. By induction, each $f'(s)$ in H can be replaced by $H_{f'}(a)[do(a, s)]$. By (ii), the result will then be uniform in $do(a, s)$, and thus we obtain a formula $H_f(a)$ where $\mathcal{D} \models f(s) = H_f(a)[do(a, s)]$ as desired. ∎

Example 4: Consider (1). This trivially satisfies (i) and (ii). The mapping from $v(s)$ to $v(do(a, s))$ is bijective and so (iii) is satisfied also. (In general, any $E_f(a)$ that is restricted to addition or multiplication by constants will satisfy (iii).) So the fluent is invertible and we have $v(s) = H_v(a)[do(a, s)]$, where $H_v(a)$ is If $\exists z(a = mv(z))$ Then $v - z$ Else v.

Example 5: Consider (5). Here the mapping is not bijective because of the *max* function and the fluent h is not invertible. If $h(do(\alpha, s)) = 0$ where $\alpha = towall(4)$, then the value of $h(s)$ cannot be determined and can be anything less than 4.

Example 6: Consider a successor state axiom like this:

$$v(do(a, s)) = \text{If } \exists z(a = mv(z)) \text{ Then } (v(s))^z \text{ Else } v(s).$$

For $\alpha = mv(2)$, we obtain a squaring function, which is not bijective. Indeed, from $v(do(\alpha, s)) = 9$, one cannot determine whether $v(s)$ was -3 or 3. So the fluent is not invertible.

Example 7: Consider this successor state axiom:

$$v(do(a, s)) = \text{If } \exists z(a = rapid(z)) \wedge wax(s) \leq .3 \text{ Then } v(s) + (fuel(s))^z \text{ Else } v(s).$$

Suppose further:

$$fuel(do(a, s)) = \text{If } \exists z(a = fill(z)) \text{ Then } z \cdot fuel(s) \text{ Else } fuel(s).$$

$$wax(do(a, s)) = \text{If } a = clean \text{ Then } wax(s)/10 \text{ Else } wax(s).$$

Here, the fuel tank and the wax on the floor determine the distance moved, and one can clean the floor wax and fill the tank. This theory is invertible, and $H_v(a)$ is given by

$$H_v(a) = \text{If } \exists z(a = rapid(z)) \wedge wax \leq .3 \text{ Then } v - fuel^z \text{ Else } v.$$

That is, because *rapid*(z) does not affect *wax* and *fuel*, we simply invert the successor state axiom for v and relativize everything to $do(a, s)$. If (say) the action *rapid(z)* also affected *fuel*, by the ordering in (i), we would first obtain the H-expression for *fuel* and use it in the H-expression for v.

Finally, note that the bijection property does not prevent us from using non-bijective functions, such as squares, in the successor state axiom of v, provided that these only apply to the other fluents. (The remaining fluents essentially behave as constants at any given situation.) In our experience, many commonly occurring successor state axioms are invertible.

Progression

In this section we propose a definition of progression that applies to any invertible basic action theory. Note that the definition of invertibility imposes no constraint on $\mathcal{D}_0$. So the definition in this section is general in that the p may appear in $\mathcal{D}_0$ in an unrestricted way, such as in (4). Given such a theory $\mathcal{D}_0 \cup \Sigma$ and a ground action α, we define a transformation $\mathcal{D}_0'$ such that $\mathcal{D}_0' \cup \Sigma$ agrees with $\mathcal{D}_0 \cup \Sigma$ on the future of α. Then, in the next section, we will consider how $\mathcal{D}_0'$ grows as a result of this progression.

Classical Progression

Let us first consider the simpler case of progression for a $\mathcal{D}_0$ that does not mention the p fluent (and the quantification over initial situations that comes with it), and so where $\mathcal{D}_0$ is uniform in S_0. In this case, because we are assuming a finite set of nullary fluents, any basic action theory can be shown to be *local-effect* (Liu and Lakemeyer 2009), where progression is first-order definable. The new theory is computed by appealing to the notion of *forgetting* (Lin and Reiter 1994). If the basic action theory is invertible, however, the progression can *also* be defined in another way. Let $\mathcal{D}_0'$ be $\mathcal{D}_0$ but with any $f(S_0)$ term in it replaced by $H_f(\alpha)[S_0]$.

Theorem 8: *Let $\mathcal{D}_0 \cup \Sigma$ be any invertible basic action theory not mentioning p and α any ground action. Then for any $\mathcal{L}$-formula ϕ rooted in now*

$$\mathcal{D}_0 \cup \Sigma \models \phi[do(\alpha, S_0)] \;\; \textit{iff} \;\; \mathcal{D}_0' \cup \Sigma \models \phi[S_0].$$

Example 9: Consider (1), and the $H_v(a)$ from Example 4. Suppose $\mathcal{D}_0 = \{v(S_0) > 10\}$. Then:

$$\begin{aligned}\mathcal{D}_0' &= (H_v(mv(3))[S_0]) > 10\\ &= (\textsc{If}\ \exists z(mv(z) = mv(3))\\ &\qquad \textsc{Then}\ v(S_0) - z\ \textsc{Else}\ v(S_0)) > 10\\ &= (v(S_0) - 3 > 10)\\ &= (v(S_0) > 13).\end{aligned}$$

Therefore, as expected, the progression of $v(S_0) > 10$ wrt a noise-free motion of 3 units is $v(S_0) > 13$. (The unique name axiom and arithmetic are used in the simplification.)

Progressing Degrees of Belief

There are two main complications when progressing beliefs wrt noisy sensors and actions. First, the p fluent will have to take the likelihood of the action α into account. Second, $\mathcal{D}_0$ need not be uniform in S_0, since p typically requires quantification over initial situations (as in (3), for example). This leads to the following definition:

Definition 10: Let $\mathcal{D}_0 \cup \Sigma$ be an invertible basic action theory and α be a ground action of the form $A(\vec{t})$ where $\vec{t}$ is uniform in *now*.[6] Then $Pro(\mathcal{D}_0, \alpha)$ is defined as $\mathcal{D}_0$ with the following substitutions:

- $p(\iota, S_0)$ is replaced by $\dfrac{p(\iota, S_0)}{\Upsilon_A(\vec{t})[\iota]}$;
- every other fluent term $f(u)$ is replaced by $H_f(\alpha)[u]$.

Here, $\Upsilon_A(\vec{x})$ refers to the RHS of the likelihood axiom for $A(\vec{x})$.

The main result of this paper is the correctness of this definition of progression:

Theorem 11: *Under the conditions of the definition above, let $\mathcal{D}_0' = Pro(\mathcal{D}_0, \alpha)$. Suppose that $\mathcal{D}_0 \models (\Upsilon_A(\vec{t}) \neq 0)[S_0]$. Then for any $\mathcal{L}$-formula ϕ rooted in now,*

$$\mathcal{D}_0 \cup \Sigma \models \phi[do(\alpha, S_0)] \;\; \textit{iff} \;\; \mathcal{D}_0' \cup \Sigma \models \phi[S_0].$$

Proof sketch: The proof is based on ideas in (Lin and Reiter 1997). To show that $\mathcal{D}_0' \cup \Sigma$ is the progression of $\mathcal{D}_0 \cup \Sigma$, we need to show that for any model M, M is a model of $\mathcal{D}_0' \cup \Sigma$ iff there is a model M' of $\mathcal{D}_0 \cup \Sigma$ such that for any ϕ, $M \models \phi[S_0]$ iff $M' \models \phi[do(\alpha, S_0)]$. We then use the fact that for any situation term σ, $f(\sigma)$ and $f(do(\alpha, \sigma))$ are related precisely to each other in terms of $E_f(\alpha)$ and $H_f(\alpha)$ to satisfy this model-theoretic property. ∎

This theorem gives us the desired property for *Bel* (which is defined in terms of p) as a corollary:

Corollary 12: *Suppose $\mathcal{D}_0, \Sigma, \mathcal{D}_0', \phi$, and α are as above. Then for all real numbers n:*

$$\begin{aligned}&\mathcal{D}_0 \cup \Sigma \models \textit{Bel}(\phi, do(\alpha, S_0)) = n\\ &\qquad \textit{iff} \quad \mathcal{D}_0' \cup \Sigma \models \textit{Bel}(\phi, S_0) = n.\end{aligned}$$

Thus the degree of belief in ϕ after a physical or sensing action is equal to the initial belief in ϕ in a progressed theory.

We now present some examples, considering, in turn, noise-free actions, noisy sensing and finally noisy actions.

Example 13: Let us consider an action theory with a vertical action *mv*(z), a sensing action *sonar*(z) and two horizontal actions: *towards* moves the robot halfway towards the wall and *away* moves the robot halfway away from the wall. Formally, let $\mathcal{D}_0 \cup \Sigma$ be an action theory where $\mathcal{D}_0$ contains just (3), and Σ includes

- foundational axioms and (P1)-(P3) as usual;
- a l-axiom for *sonar*(z), namely (2);
- l-axioms for the other actions, which are noise-free, and so these simply equal 1;
- a successor state axiom for v, namely (1);
- the following successor state axiom for h:

$$\begin{aligned}&h(do(a, s)) =\\ &\quad \textsc{If}\ a = \textit{away}\ \textsc{Then}\ 3/2 \cdot h(s)\\ &\quad \textsc{Else If}\ a = \textit{towards}\ \textsc{Then}\ 1/2 \cdot h(s)\\ &\quad \textsc{Else}\ h(s).\end{aligned}$$

[6] In the most common case (like noise-free or sensing actions), the arguments to the action would simply be a vector of constants.

We noted that (5) does not satisfy our invertibility property. This variant, however, is invertible. The H-expression for v was derived in Example 4. The H-expression for h is:

$$H_h(a) = \text{If } a = \mathit{away} \text{ Then } 2/3 \cdot h \text{ Else If } a = \mathit{towards} \text{ Then } 2 \cdot h \text{ Else } h.$$

We now consider the progression of $\mathcal{D}_0$ wrt the action *away*. First, the instantiated H-expressions would simplify to:

- $H_h(\mathit{away}) = 2/3 \cdot h$;
- $H_v(\mathit{away}) = v$.

Next, since *away* is noise-free, we have $\Upsilon_{\mathit{away}} = 1$. Putting this together, we obtain $\mathcal{D}_0' = \mathit{Pro}(\mathcal{D}_0, \mathit{away})$ as:

$$\begin{aligned} p(s, S_0) &= \mathcal{U}(2/3 \cdot h; 2, 12) \times \mathcal{N}(v; 0, 1)\ [s] \\ &= \mathcal{U}(h; 3, 18) \times \mathcal{N}(v; 0, 1)\ [s] \end{aligned}$$

That is, the new p is one where h is uniformly distributed on $[3, 18]$ and v is independently drawn from a standard normal distribution (as before). This leads to a *shorter* and *wider* density function, as depicted in Figure 2. Here are three simple properties to contrast the original *vs.* the progressed:

- $\mathcal{D}_0 \cup \Sigma \models \mathit{Bel}(h \geq 9, S_0) = .3$.

  The *Bel* term expands as:

  $$\frac{1}{\eta} \int_x \int_y \text{If } \exists \iota (h = x \wedge v = y \wedge h \geq 9)[\iota] \text{ Then } \mathcal{U}(h; 2, 12) \times \mathcal{N}(v; 0, 1)[\iota] \text{ Else } 0$$

  which simplifies to the integration of a density function:

  $$\begin{aligned} &\frac{1}{\eta} \int_x \int_y \begin{cases} .1 \times \mathcal{N}(y; 0, 1) & \text{if } x \in [2, 12], x \geq 9 \\ 0 & \text{otherwise} \end{cases} \\ = &\frac{1}{\eta} \int_x \int_y \begin{cases} .1 \times \mathcal{N}(y; 0, 1) & \text{if } x \in [9, 12] \\ 0 & \text{otherwise} \end{cases} \\ = &.3. \end{aligned}$$

  Only those situations where $h \in [2, 12]$ initially are given non-zero p values and by the formula in the *Bel*-term, only those where $h \geq 9$ are to be considered.

- $\mathcal{D}_0 \cup \Sigma \models \mathit{Bel}(h \geq 9, \mathit{do}(\mathit{away}, S_0)) = .6$.

  For any initial situation ι, $h[\mathit{do}(\mathit{away}, \iota)] \geq 9$ only when $h[\iota] \geq 6$, which is given an initial belief of .6.

- $\mathcal{D}_0' \cup \Sigma \models \mathit{Bel}(h \geq 9, S_0) = .6$.

  Basically, *Bel* simplifies to an expression of the form:

  $$\frac{1}{\eta} \int_x \int_y \begin{cases} 1/15 \cdot \mathcal{N}(y; 0, 1) & \text{if } x \in [3, 18], x \geq 9 \\ 0 & \text{otherwise} \end{cases}$$

  giving us .6.

Example 14: Let $\mathcal{D}_0 \cup \Sigma$ be exactly as above, and consider its progression wrt *towards*. It is easy to verify that for instantiated H-expressions we get:

- $H_h(\mathit{towards}) = 2 \cdot h$;
- $H_v(\mathit{towards}) = v$;

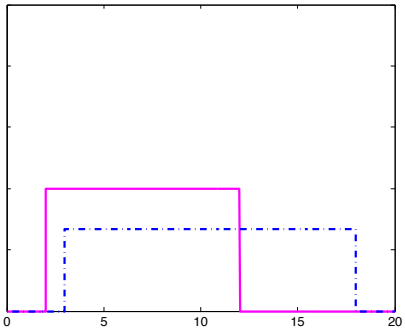


Figure 2: Belief change about h: initially (solid magenta) and after moving away (dotted blue).

Here too, because *towards* is noise-free, $\Upsilon_{\mathit{towards}}$ is simply 1, which is to say the $\mathcal{D}_0' = \mathit{Pro}(\mathcal{D}_0, \mathit{towards})$ is defined as:

$$\begin{aligned} p(s, S_0) &= \mathcal{U}(2 \times h; 2, 12) \times \mathcal{N}(v; 0, 1)\ [s] \\ &= \mathcal{U}(h; 1, 6) \times \mathcal{N}(v; 0, 1)\ [s]. \end{aligned}$$

The new distribution on h is *narrower* and *taller*, as shown in Figure 3. Here we might contrast $\mathcal{D}_0$ and $\mathcal{D}_0'$ as follows:

- $\mathcal{D}_0 \cup \Sigma \models \mathit{Bel}(h \in [2, 3], S_0) = .1$.
- $\mathcal{D}_0' \cup \Sigma \models \mathit{Bel}(h \in [2, 3], S_0) = .2$.

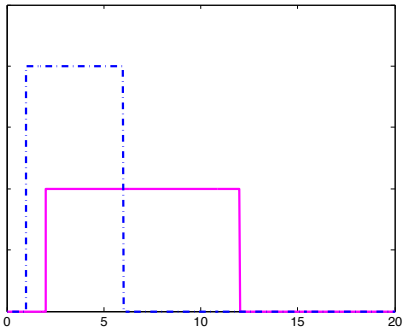


Figure 3: Belief change about h: initially (solid magenta) and moving towards the wall (dotted blue).

Example 15: Let $\mathcal{D}_0 \cup \Sigma$ be as in the previous examples. Consider its progression wrt the action *sonar*(5). Sensing actions do not affect fluents, so for H-expressions we have:

- $H_h(\mathit{sonar}(5)) = h$;
- $H_v(\mathit{sonar}(5)) = v$.

Here $\mathit{sonar}(z)$ is noisy, and we have $\Upsilon_{\mathit{sonar}}(5) = \mathcal{N}(5; h, 4)$. This means that the progression $\mathcal{D}_0' = \mathit{Pro}(\mathcal{D}_0, \mathit{sonar}(5))$ is

$$\frac{p(\iota, S_0)}{\mathcal{N}(5; h, 4)[\iota]} = \mathcal{U}(h; 2, 12) \times \mathcal{N}(v; 0, 1)[\iota],$$

which simplifies to the following:

$$p(\iota, S_0) = \mathcal{U}(h; 2, 12) \times \mathcal{N}(v; 0, 1) \times \mathcal{N}(5; h, 4)\ [\iota].$$

As can be noted in Figure 4, the robot's belief about h's true value around 5 has sharpened. Consider, for example, that:

- $\mathcal{D}_0 \cup \Sigma \models Bel(h \leq 9, S_0) = .7$.
- $\mathcal{D}'_0 \cup \Sigma \models Bel(h \leq 9, S_0) \approx .97$.

If we were to progress $\mathcal{D}'_0$ further wrt a second sensing action, say *sonar*(5.9), we would obtain the following:

$$p(\iota, S_0) = \mathcal{U}(h; 2, 12) \times \mathcal{N}(v; 0, 1) \times \mathcal{N}(5; h, 4) \times \mathcal{N}(5.9; h, 4)\ [\iota].$$

As can be seen in Figure 4, the robot's belief about h would sharpen significantly after this second sensing action. If we let $\mathcal{D}''_0 = Pro(\mathcal{D}'_0, sonar(5.9))$ then:

- $\mathcal{D}''_0 \cup \Sigma \models Bel(h \leq 9, S_0) \approx .99$.

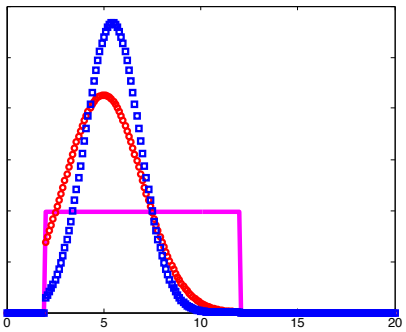


Figure 4: Belief change about h: initially (solid magenta), after sensing 5 (red circles), and after sensing twice (blue squares).

Example 16: Let $\mathcal{D}_0$ be the conjunction of (3) and $v(S_0) = 3$. Let Σ be the union of:

- (P1)-(P3) and domain-independent foundational axioms;
- a successor state axiom for h as above;
- a noisy move action mv' with the following l-axiom:

$$l(mv'(x, y), s) = \mathcal{N}(y; x, 2)$$

- a successor state axiom for v using this noisy move:

$$v(do(a, s)) = \text{If } \exists x, y(a = mv'(x, y)) \text{ Then } v(s) + y \text{ Else } v(s).$$

(Recall that for a noisy move $mv'(x, y)$, x is the intended motion and y is the actual motion.) This is inverted using the same idea as in Example 4.

Consider the progression of $\mathcal{D}_0 \cup \Sigma$ wrt $mv'(2, g(S_0))$. The simplified H-expressions are as follows:

- $H_h(mv'(2, g)) = h$;
- $H_v(mv'(2, g)) = v - g$.

By definition, occurrences of $v(u)$ in $\mathcal{D}_0$ are to be replaced by $H_v(mv'(2, g))[u]$. Also, $\Upsilon_{mv'}(2, z) = \mathcal{N}(z; 2, 2)$. Therefore, $\mathcal{D}'_0 = Pro(\mathcal{D}_0, mv'(2, g(S_0)))$ is defined to be

$$(v - g)[S_0] = 3 \wedge \left(\frac{p(\iota, S_0)}{\mathcal{N}(g; 2, 2)[\iota]} = \mathcal{U}(h; 2, 12) \times \mathcal{N}(v - g; 0, 1)[\iota]\right)$$

This simplifies to the conjunction of these two sentences:

- $v(S_0) = 3 + g(S_0)$; and
- $p(\iota, S_0) = \mathcal{U}(h; 2, 12) \times \mathcal{N}(v; g, 1) \times \mathcal{N}(g; 2, 2)\ [\iota]$.

Thus the noisy action has had two effects: the actual position has shifted by $g(S_0)$ units, and the belief about the position has also shifted by an amount g drawn from a normal distribution centered around 2. This leads to a *shifted* and *wider* curve seen in Figure 5. As expected, the agent is considerably less confident about its position after a noisy move. Here, for example, are the degrees of belief about being located within 1 unit of the best estimate (that is, the mean):

- $\mathcal{D}_0 \cup \Sigma \models Bel(v \in [-1, 1], S_0) \approx .68$.
- $\mathcal{D}'_0 \cup \Sigma \models Bel(v \in [1, 3], S_0) \approx .34$.

Basically, *Bel* expands to an expression of the form

$$\frac{1}{\eta} \int_{x,y,z} \begin{cases} .1\mathcal{N}(y; z, 1) \cdot \mathcal{N}(z; 2, 2) & \text{if } x \in [2, 12], y \in [1, 3] \\ 0 & \text{otherwise} \end{cases}$$

where η is

$$\int_{x,y,z} \begin{cases} .1\mathcal{N}(y; z, 1) \cdot \mathcal{N}(z; 2, 2) & \text{if } x \in [2, 12] \\ 0 & \text{otherwise} \end{cases}$$

leading to .34.

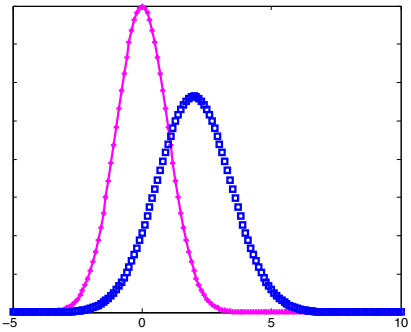


Figure 5: Belief change about v: initially (solid magenta) and after a noisy move of 2 units (blue squares).

Constant Growth

As seen in the examples, the result of progression is a theory $\mathcal{D}'_0$ which is essentially obtained by adding H-expressions, and so, its size is linear in the size of the *action theory*:

Theorem 17: *Suppose $\mathcal{D} = \mathcal{D}_0 \cup \Sigma$ is any invertible basic action theory. After the iterative progression of $\mathcal{D}_0 \cup \Sigma$ wrt a sequence σ, the size of the new initial theory is $O(|\mathcal{D}| \times |\sigma|)$.*

Therefore, progression is both computable and efficient in the sense of (Liu and Lakemeyer 2009). But for realistic robotic applications, even this may not be enough, especially over millions of actions. Consider, for example, that to calculate a degree of belief it will be necessary to integrate the numerical expression for p. What we turn to in this section is a special case that would guarantee that over any length of action sequences, the size of the progressed theory does not change beyond a constant factor. It will use the notion of *context-completeness* (Liu and Levesque 2005) and a few simplification rules.

Definition 18: Suppose $F \subseteq \{f_1, \ldots, f_k\}$ is any set of fluents, and $\mathcal{D}_0 \cup \Sigma$ is any invertible basic action theory. We say that $\mathcal{D}_0$ is *complete* wrt F if for any $\phi \in \mathcal{L}_F$, either $\mathcal{D}_0 \models \phi$ or $\mathcal{D}_0 \models \neg\phi$, where $\mathcal{L}_F$ is the sublanguage of $\mathcal{L}$ restricted to the fluents in F.

Definition 19: An invertible basic action theory $\mathcal{D}_0 \cup \Sigma$ is said to be *context-complete* iff

- for every fluent f, $\mathcal{D}_0$ is complete wrt every fluent other than f appearing in the successor state axiom of f;
- $\mathcal{D}_0$ is complete wrt every fluent appearing in a conditional expression in the likelihood axioms.

That is, there is sufficient information in $\mathcal{D}_0$ to simplify all the conditionals appearing in the context formulas of the successor state axioms and the likelihood axioms.

STRIPS actions are trivially context-complete, and so are Reiter's *context-free* successor state axioms where only rigid symbols appear in the RHS (Reiter 2001). In Example 7, if $\mathcal{D}_0$ is complete wrt the fluents *fuel* and *wax*, then the theory would be context-complete. Note that $\mathcal{D}_0$ does not need to be complete wrt the fluent v in that example, and this is precisely why they are interesting. Indeed, both (5) and (1) are also context-complete because, by definition, E_f may mention f, and (say) use its previous value.[7] The reader may further verify that all the density change examples developed in the paper are context-complete.

As a result of context-completeness, the body of the successor state axiom for a fluent f would simplify to an expression involving only f. As a consequence, H-expressions simplify to expressions of the form:

$$f(s) = \theta(f(do(a, s)))$$

where $\theta(x)$ is some rigid $\mathcal{L}$-term, possibly an arithmetic expression, with a single free variable x. Analogously, if $A(\vec{t})$ is any ground action, then,

$$\Upsilon_A(\vec{t}) = \delta(\vec{t}, \vec{f}).$$

where $\delta(\vec{x}, \vec{y})$ is some rigid $\mathcal{L}$-term, possibly an arithmetic expression, with free variables $\vec{x} \cup \vec{y}$. So in a progression, every fluent f would be replaced by an arithmetic expression using only f, resulting in a theory that is linear in the size of the *initial theory* $\mathcal{D}_0$ only:

Theorem 20: *Suppose $\mathcal{D}_0 \cup \Sigma$ is any invertible basic action theory that is also context-complete. After the iterative progression of $\mathcal{D}_0 \cup \Sigma$ wrt a sequence σ, the size of the new initial theory is $O(|\mathcal{D}_0| \times |\sigma|)$.*

This can make a substantial difference in the size of the expression for p. Nevertheless, after n actions, we might still end up with an expression consisting of a product of n terms. We now simplify this computation further by appealing to the analytical tractability of *conjugate distributions* (Box and Tiao 1973; Bertsekas and Tsitsiklis 2008), such as Gaussians, where the product of two Gaussians is definable as another Gaussian. We show how this helps with an example.

[7] Strictly speaking, our notion of context-completeness is inspired by, but not the same as the one in (Liu and Levesque 2005). This pertains to the clarification just made: we allow successor state axioms to use the fluent's previous value.

Example 21: Let $\mathcal{D}_0 \cup \Sigma$ be as in Example 16. We noted its progression wrt $mv'(2, g(S_0))$ includes:

$$p(\iota, S_0) = \mathcal{U}(h; 2, 12) \times \mathcal{N}(v; g, 1) \times \mathcal{N}(g; 2, 2)\ [\iota].$$

If we progress this sentence further wrt a second noisy action $mv'(3, g'(S_0))$, we would obtain:

$$p(\iota, S_0) = \mathcal{U}(h; 2, 12) \times \mathcal{N}(v - g'; g, 1) \times \mathcal{N}(g; 2, 2) \times \mathcal{N}(g'; 3, 2)\ [\iota] \quad (6)$$

By letting g^* be a new fluent symbol such that $g^* = g + g'$, it follows that (6) is equivalent to the following:

$$p(\iota, S_0) = \mathcal{U}(h; 2, 12) \times \mathcal{N}(v; g^*, 1) \times \mathcal{N}(g^*; 5, 4)\ [\iota].$$

Analogously, after any number n of noisy moves, we would only need to integrate over 3 variables *vs.* $n + 2$ in (6).

Theorem 22: *Suppose $\mathcal{D}_0 \cup \Sigma$ is any invertible action theory that is context-complete and the likelihood axioms are normal distributions. After the iterative progression of $\mathcal{D}_0 \cup \Sigma$ wrt a sequence σ, the size of the new initial theory is $O(|\mathcal{D}_0|)$.*

That is, the size of the progression does not grow beyond a constant factor. This result is easily adapted to other conjugate distributions. Essentially, like other *filtering* techniques (Thrun, Burgard, and Fox 2005), such as Kalman filters, we conclude that progression can be made very efficient. But unlike some of these other techniques, we are not limited to Gaussians. More significantly, unlike all other probabilistic formalisms, we can allow arbitrary context formulas, and under the reasonable assumption of context-completeness, still remain efficient.

Related Work and Discussion

This work builds on Lin and Reiter's (1997) notion of progression. Other advances on progression have been made since then (Liu and Lakemeyer 2009; Vassos 2009), mainly by appealing to the notion of *forgetting* (Lin and Reiter 1994). We were motivated by concerns about continuity, and this led to the notion of invertible theories. These theories allowed us to perform first-order progression by inverting successor state axioms in a way that, as far as we know, has not been investigated before. Although we restricted $\mathcal{L}$ to nullary real-valued fluents, we suspect that invertibility and its connection to progression may apply more generally. This is left for future investigations.

The progression of categorical knowledge against noise-free effectors and sensors is considered in (Liu and Wen 2011; Lakemeyer and Levesque 2009). The progression of discrete degrees of belief wrt context-completeness is considered in (Belle and Lakemeyer 2011). In the fluent calculus (Thielscher 2001), a dual form of successor state axioms is used, leading naturally to a form of progression. However, continuity is not considered in any of these.

The form of progression considered here follows Lin and Reiter and differs from weaker forms including the one proposed by Liu and Levesque (2005), and the notion of *logical filtering* (Shirazi and Amir 2005; Hajishirzi and Amir 2010),

which is a form of (approximate) progression.[8] Interestingly, logical filtering is inspired by Kalman filters (Thrun, Burgard, and Fox 2005), although the precise connection is not considered. In situation calculus terminology, Kalman filters and its variants are derived using *strongly* context-free (Reiter 2001) noisy actions and sensors, with additive Gaussian noise, over normally distributed fluents. See, for example, (Belle and Levesque 2013b). Indeed, as discussed in BL, the framework used here is significantly more general than probabilistic formalisms used in the robotics and uncertainty literature, including Dynamic Bayesian Networks (Boyen and Koller 1998; Darwiche and Goldszmidt 1994) and hybrid control structures (McIlraith et al. 2000), among others.

See BL for a discussion of how the formalism used here also differs from other logical accounts for reasoning about uncertainty, such as probabilistic logics (Fagin and Halpern 1994), Markov logics (Richardson and Domingos 2006; Choi, Guzman-Rivera, and Amir 2011), probabilistic planning languages (Kushmerick, Hanks, and Weld 1995; Younes and Littman 2004; Sanner 2011), dynamic logics (Van Benthem, Gerbrandy, and Kooi 2009), and previous first-order proposals (Mateus et al. 2001; Thielscher 2001; Martin and Thielscher 2009; Fritz and McIlraith 2009; Poole 1998). In essence, none of these address continuous uncertainty and continuous noise in a general way.

Conclusions

Lin and Reiter developed the notion of progression, with long-lived agents in mind. However, their account does not deal with probabilistic uncertainty nor with noise, as seen in real-world robotic applications. In the work here, we consider semantically correct progression in the presence of continuity. By first identifying what we called invertible basic action theories, we obtained a new way of computing progression. Under the additional restriction of context-completeness, progression is very efficient. Most significantly, by working within a richer language, we have obtained progression machinery that, to the best of our knowledge, has not been discussed elsewhere, and goes beyond existing techniques. The unrestricted nature of the specification of the p fluent, for example, which we inherit from (Bacchus, Halpern, and Levesque 1999), allows for agents whose beliefs are not determined by a unique distribution.

One major topic for future investigation is an implementation. But apart from simply progressing a theory, we would also like to support the numerical calculation of degrees of belief. For this, we may need to appeal to numerical integration techniques such as Monte Carlo sampling (Murphy 2012), and so we would also like to study formal constraints on $\mathcal{D}_0$ which guarantee that sampling can be done in an effective manner.

[8] In (Shirazi and Amir 2005; Hajishirzi and Amir 2010), the notion of a *permuting action* is introduced for computing their form of progression, which bears some similarity to invertible fluents. However, as mentioned above, our work on continuity led itself to invertibility. Neither continuous uncertainty nor continuous noise is considered in (Shirazi and Amir 2005; Hajishirzi and Amir 2010).

Acknowledgements

The authors are grateful to the Natural Sciences and Engineering Research Council of Canada for financial support. The authors also thank the reviewers for their feedback.

References

Bacchus, F.; Halpern, J. Y.; and Levesque, H. J. 1999. Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence* 111(1–2):171 – 208.

Baker, A. B. 1989. A simple solution to the yale shooting problem. In *Proc. KR*, 11–20.

Belle, V., and Lakemeyer, G. 2011. A semantical account of progression in the presence of uncertainty. In *Proc. AAAI*, 165–170.

Belle, V., and Levesque, H. J. 2013a. Reasoning about continuous uncertainty in the situation calculus. In *Proc. IJCAI*.

Belle, V., and Levesque, H. J. 2013b. Reasoning about probabilities in dynamic systems using goal regression. In *Proc. UAI*.

Bertsekas, D., and Tsitsiklis, J. 2008. *Introduction to probability*. Athena Scientific, 2nd edition edition.

Box, G. E. P., and Tiao, G. C. 1973. *Bayesian inference in statistical analysis*. Addison-Wesley.

Boyen, X., and Koller, D. 1998. Tractable inference for complex stochastic processes. In *Proc. UAI*, 33–42.

Choi, J.; Guzman-Rivera, A.; and Amir, E. 2011. Lifted relational kalman filtering. In *Proc. IJCAI*, 2092–2099.

Darwiche, A., and Goldszmidt, M. 1994. Action networks: A framework for reasoning about actions and change under uncertainty. In *Proc. UAI*, 136–144.

Fagin, R., and Halpern, J. Y. 1994. Reasoning about knowledge and probability. *J. ACM* 41(2):340–367.

Fritz, C., and McIlraith, S. A. 2009. Computing robust plans in continuous domains. In *Proc. ICAPS*, 346–349.

Hajishirzi, H., and Amir, E. 2010. Reasoning about deterministic actions with probabilistic prior and application to stochastic filtering. In *Proc. KR*.

Kushmerick, N.; Hanks, S.; and Weld, D. 1995. An algorithm for probabilistic planning. *Artificial Intelligence* 76(1):239–286.

Lakemeyer, G., and Levesque, H. J. 2007. Cognitive robotics. In *Handbook of Knowledge Representation*. Elsevier. 869–886.

Lakemeyer, G., and Levesque, H. 2009. A semantical account of progression in the presence of defaults. In *Conceptual Modeling: Foundations and Applications*. Springer. 82–98.

Levesque, H. J.; Pirri, F.; and Reiter, R. 1998. Foundations for the situation calculus. *Electron. Trans. Artif. Intell.* 2:159–178.

Lin, F., and Reiter, R. 1994. Forget it. In *Working Notes of AAAI Fall Symposium on Relevance*, 154–159.

Lin, F., and Reiter, R. 1997. How to progress a database. *Artificial Intelligence* 92(1-2):131–167.

Liu, Y., and Lakemeyer, G. 2009. On first-order definability and computability of progression for local-effect actions and beyond. In *Proc. IJCAI*, 860–866.

Liu, Y., and Levesque, H. 2005. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proc. IJCAI*, 522–527.

Liu, Y., and Wen, X. 2011. On the progression of knowledge in the situation calculus. In *Proc. IJCAI*, 976–982.

Martin, Y., and Thielscher, M. 2009. Integrating reasoning about actions and Bayesian networks. In *International Conference on Agents and Artificial Intelligence (ICAART)*.

Mateus, P.; Pacheco, A.; Pinto, J.; Sernadas, A.; and Sernadas, C. 2001. Probabilistic situation calculus. *Annals of Math. and Artif. Intell.* 32(1-4):393–431.

McCarthy, J., and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, 463–502.

McIlraith, S.; Biswas, G.; Clancy, D.; and Gupta, V. 2000. Hybrid systems diagnosis. In *Proc. of Workshop on Hybrid Systems: Computation and Control (HSCC 2000)*, LNCS, 282–295.

Murphy, K. P. 2012. *Machine learning: a probabilistic perspective*. The MIT Press.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Poole, D. 1998. Decision theory, the situation calculus and conditional plans. *Electron. Trans. Artif. Intell.* 2:105–158.

Reiter, R. 2001. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT Press.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine learning* 62(1):107–136.

Sanner, S. 2011. Relational dynamic influence diagram language (rddl): Language description. Technical report, Australian National University.

Scherl, R. B., and Levesque, H. J. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence* 144(1-2):1–39.

Shirazi, A., and Amir, E. 2005. First-order logical filtering. In *Proc. IJCAI*, 589–595.

Thielscher, M. 2001. Planning with noisy actions (preliminary report). In *Proc. Australian Joint Conference on Artificial Intelligence*, 27–45.

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics*. MIT Press.

Trench, W. F. 2003. *Introduction to real analysis*. Prentice Hall.

Van Benthem, J.; Gerbrandy, J.; and Kooi, B. 2009. Dynamic update with probabilities. *Studia Logica* 93(1):67–96.

Vassos, S., and Levesque, H. 2008. On the Progression of Situation Calculus Basic Action Theories: Resolving a 10-year-old Conjecture. In *Proc. AAAI*, 1004–1009.

Vassos, S. 2009. *A Reasoning Module for Long-Lived Cognitive Agents*. Ph.D. Dissertation, University of Toronto, Toronto, Canada.

Waldinger, R. 1977. Achieving several goals simultaneously. In *Machine Intelligence*, volume 8. 94–136.

Younes, H., and Littman, M. 2004. PPDDL 1. 0: An extension to pddl for expressing planning domains with probabilistic effects. Technical report, Carnegie Mellon University.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Transforming Situation Calculus Action Theories for Optimised Reasoning

Christopher Ewin
National ICT Australia and
Computing & Information Systems
The University of Melbourne
chris.ewin@nicta.com.au

Adrian R. Pearce
National ICT Australia and
Computing & Information Systems
The University of Melbourne
adrianrp@unimelb.edu.au

Stavros Vassos
DIAG
Sapienza University of Rome
Rome, Italy
vassos@dis.uniroma1.it

Abstract

Among the most frequent reasoning tasks in the situation calculus are projection queries that query the truth of conditions in a future state of affairs. However, in long running action sequences solving the projection problem is complex. The main contribution of this work is a new technique which allows the length of the action sequences to be reduced by reordering independent actions and removing dominated actions; maintaining semantic equivalence with respect to the original action theory. This transformation allows for the removal of actions that are problematic with respect to progression, allowing for periodical update of the action theory to reflect the current state of affairs. We provide the logical framework for the general case and give specific methods for two important classes of action theories. The work provides the basis for handling more expressive cases, such as the reordering of sensing actions in order to delay progression, and forms an important step towards facilitating ongoing planning and reasoning by long-running agents. It provides a mechanism for minimising the need for keeping the action history while appealing to both regression and progression.

Introduction

In this paper we are concerned with reasoning about action and change in the general case where information about the environment is represented as a first-order logical theory. In practical settings two of the most typical reasoning tasks required are: performing *projection* queries that query the truth of conditions in a future state of affairs; and *updating* or *progressing* the world representation to reflect the current state of affairs after a sequence of actions has been performed.

To illustrate optimised reasoning for these tasks we focus on the logical language of the situation calculus (McCarthy and Hayes 1969; Reiter 2001), where the environment is formalized as an action theory including a set of first-order sentences for specifying the initial state of the environment and a set of first-order axioms for specifying the effects of named actions with respect to the changing properties of the environment. The first task amounts to the question of whether a first-order sentence corresponding to the projection query is entailed by the action theory, while the second involves finding a new updated first-order representation of the world which is typically referred to as progression.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

These two tasks are very much connected as a proper solution for progression ensures that the truth value of all possible projection queries is preserved in the updated representation of the environment. However, solving the progression problem in the general case is quite tricky, in fact requiring the use of second-order axioms (Lin and Reiter 1997; Vassos and Levesque 2008; 2013) for the updated representation. The identified problematic cases that require second-order logic are mathematically involved and are rarely encountered, occurring infrequently in practice.

Consequently, syntactically restricted forms of action theories have been investigated in order to ensure that a first-order progression exists and can be effectively computed. Examples for first-order progressable action theories include those with *local-effect actions* (Liu and Levesque 2005; Vassos, Lakemeyer, and Levesque 2008) which restrict actions to affect only ground atoms that are composed exclusively by using arguments of the action. A type of first-order 'surgery' can be performed to the initial theory, removing the truth values of (finitely many) ground atoms—i.e., by *forgetting* in the sense of (Lin and Reiter 1994)—and setting their new values accordingly. A map of progressable action theory classes can be found in (Vassos and Patrizi 2013), however, they are quite limited and a wide range of cases do not fit in the identified classes—for which we have no means of performing a (logically correct) first-order update.

In practice this means that if the desired action theory includes even a single action that does not fit into the identified progressable cases, then updating of the action theory cannot be guaranteed: from the time that a problematic action α_1 is executed no more updating can be performed. Of course, projection queries can still be decided by the original action theory by forming appropriate entailment queries of the form: 'Will ϕ be true after actions $\langle \alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m \rangle$ are performed in the initial state of the environment?', where $\alpha_1, \ldots, \alpha_n$ are the actions that have been already executed (for which the action theory unfortunately cannot be updated due to α_1) and $\beta_1, \ldots, \beta_m$ are the ones with respect to which we want to project in the future.

Although *regression* can be used to answer history-based queries, for many practical cases where the action theory is used by long-living agents in order to reason about their actions, such approaches become infeasible as the history of actions constantly increases.

An important observation is that similar to action α_1, that makes further updates impossible, there can be another action, α_k, that brings the environment to a state of affairs that is easier to represent and reason about. For example, α_k could be a *reseting* action that brings the environment to the initial state of affairs, or one that essentially *subsumes* the effects of α_1 in a way that the new representation is first-order progressable—via setting all problematic atoms to a fixed or sensed truth value. Current approaches do not consider this type of reasoning with respect to the action history, leaving out a wide range of practical cases that can be handled by such a hybrid approach to progression.

We therefore investigate sufficient and necessary conditions for transforming and simplifying the history of actions in a logically correct way by combining two basic types of operations: *swapping independent actions* in the action history; and *eliminating dominated actions* by removing them entirely from the action history. This transformation will often allow to remove actions that are problematic with respect to progression, thus allowing to periodically update the action theory to reflect the current state of affairs.

We provide a mechanism for minimising the need for keeping the action history $\langle \alpha_1, \ldots, \alpha_n \rangle$ of the *actions that have already been executed*, and having a representation of the environment that is *as much up-to-date as possible*. In case reasoning over the past history of events is also required then a separate mechanism that keeps track of the true action history and the original action theory can be employed. Importantly, our proposed mechanism can also be used to reason about and plan for the actions needed in order for a progression to become feasible when the length of the history of actions reaches a certain limit.

Finally, our approach can be also useful in cases where all actions fall in the first-order progressable classes—for the purpose of obtaining computational benefits. Since the updating of the action theory is typically a much more computationally demanding procedure than that of projection, it can be beneficial to keep a history of executed actions up to some fixed (past horizon) length anticipating that a future action may greatly simplify the history. Depending on the characteristics of the domain and the projection queries that are needed, such an approach has the potential to save a lot of computational resources. For instance, in cases where there is a large degree of uncertainty in the form of incomplete or disjunctive information which would, under normal circumstances, lead to an exponential growth of the progressed theory with respect to the number of actions executed, a later sensing action that settles uncertainty would lead to larger computational savings, as compared to those consumed in order to perform projection queries using the longer history of executed actions.

Our contribution is a framework for specifying transformations in a logically correct way in the general case. A promising new class of global-effect actions is introduced, termed *resetting* actions, which facilitate transformation based on dominance principles. We specify practical procedures for realizing the proposed transformations, allowing for swapping and elimination of actions for handling non-progressable instances; or the interleaving of regression and progression as required. We illustrate how such theories correspond to common agent planning tasks, using a slightly modified `Sokoban` domain.

Formal preliminaries

The logical language of situation calculus

The language $\mathcal{L}$ of the situation calculus as presented in (Reiter 2001) is a three-sorted first-order logic language with equality and some limited second-order features. The sorts are: *action*, *situation*, and a catch-all sort *object* for everything else depending on the domain of application.

Similar to a normal one-sorted first-order language, $\mathcal{L}$ includes function and predicate symbols. In this case since there are three sorts, each of the symbols has a type that specifies the sorts for the arguments it takes. The situation calculus includes symbols only of certain types each of which has a special role in the representation of the world and its dynamics.

An action term or simply an *action* represents an atomic action that may be performed in the world. For example consider the action $move(l_1, l_2)$ that may be used to represent that a robot moves from location l_1 to location l_2. A situation term or simply a *situation* represents a world history as a sequence of actions. The constant S_0 is used to denote the *initial situation* where no actions have occurred. Sequences of actions are built using the function symbol *do*, such that $do(\alpha, \sigma)$ represents the successor situation resulting from performing action α in situation σ.

A *relational fluent* is a predicate whose last argument is a situation, and thus whose truth value can change from situation to situation. For example, $RobotAt(l, \sigma)$ may be used to represent that the robot lies at location l in situation σ. In order to simplify the analysis we have restricted the language $\mathcal{L}$ so that there are no functional fluent symbols in $\mathcal{L}$, that is, functions whose last argument is a situation. This is not a restriction on the expressiveness of $\mathcal{L}$ as functional fluents can be represented by relational fluents with a few extra axioms.

Actions need not be executable in all situations, and the predicate atom $Poss(\alpha, \sigma)$ states that action α is executable in situation σ. For example, $Poss(move(l_1, l_2), \sigma)$ is intended to represent that the action $move(l_1, l_2)$ can be executed in situation σ. The language $\mathcal{L}$ also includes the binary predicate symbol $\sqsubset$ which provides an ordering on situations. The atom $s \sqsubset s'$ means that the action sequence s' can be obtained from the sequence s by performing one or more actions in s. We will typically use the notation $\sigma \sqsubseteq \sigma'$ as a macro for $\sigma \sqsubset \sigma' \vee \sigma = \sigma'$.

In this paper, we shall restrict our attention to a language $\mathcal{L}$ with a finite number of relational fluent symbols that only take arguments of sort object (apart their last situation argument), an infinite number of constant symbols of sort object $\mathcal{C} = \{c_1, c_2, \ldots\}$, and a finite number of function symbols of sort action that take arguments of sort object. We adopt the following notation with subscripts and superscripts: α and a for terms and variables of sort action; σ and s for terms and variables of sort situation; t and x, y, z, w for terms and variables of sort object. Also, we use A for action function symbols, F, G for fluent symbols, and b, c, d, e for constants

of sort object. Finally, we will typically write $\phi(\vec{x})$ to state that the free variables of the formula are among $\vec{x}$.

The well-formed first-order formulas of $\mathcal{L}$ are defined inductively similarly to a normal one-sorted language but also respecting that each parameter has a unique sort. As far as the second-order formulas of $\mathcal{L}$ are concerned, only quantification over relations is allowed and the well-formed formulas are defined inductively similarly to a normal second-order language.

Often we will focus on sentences that refer to a particular situation. For this purpose, for any situation term σ, we define the set of *uniform formulas in* σ to be all those (first-order or second-order) formulas in $\mathcal{L}$ that do not mention any other situation terms except for σ, do not mention *Poss*, and where σ is not used by any quantifier (Lin and Reiter 1997). Finally, we use δ to denote a sequence of actions of the form $\langle \alpha_1, \ldots, \alpha_n \rangle$, and $do(\delta, S_0)$ as a shorthand for the situation term $do(\alpha_n, \cdots do(\alpha_1, S_0))$.

Basic action theories

Within the language $\mathcal{L}$, one can formulate action theories that describe how the world changes as the result of the available actions. We focus on a variant of the *basic action theories (BATs)* (Reiter 2001) of the following form:[1]

$$\mathcal{D} = \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_0 \cup \mathcal{D}_{fnd},$$

where:

1. $\mathcal{D}_{ap}$ is the set of action precondition axioms (PAs), one per action symbol A, of the form $Poss(A(\vec{y}), s) \equiv \Pi_A(\vec{y}, s)$, where $\Pi_A(\vec{y}, s)$ is first-order and uniform in s. PAs characterize the conditions under which actions are physically possible.
2. $\mathcal{D}_{ss}$ is the set of successor state axioms (SSAs), one per fluent symbol F, of the form $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$, where $\Phi_F(\vec{x}, a, s)$ is first-order and uniform in s. SSAs describe how fluents change between situations as the result of actions.
3. $\mathcal{D}_{una}$ is the set of unique-names axioms for actions.
4. $\mathcal{D}_0$, the *initial knowledge base (KB)*, is a set of first-order sentences uniform in S_0 describing the initial situation S_0.
5. $\mathcal{D}_{fnd}$ is the set of domain independent axioms of the situation calculus, formally defining the legal situations. A second-order induction axiom is included in $\mathcal{D}_{fnd}$.

Probably the most interesting component of a basic action theory is the set successor state axioms, which together encode the dynamics of the domain being represented. Technically, SSAs are meant to capture the effects and non-effects of actions. To achieve that in a parsimonious way, one typically follows the well-known solution to the frame problem by means of successor state axioms of the following form (Reiter 2001):

$$F(\vec{x}, do(a, s)) \equiv \gamma^+_F(\vec{x}, a, s) \vee F(\vec{x}, s) \wedge \neg\gamma^-_F(\vec{x}, a, s),$$

where both $\gamma^+_F(\vec{x}, a, s)$ and $\gamma^-_F(\vec{x}, a, s)$ are first-order formulas uniform in s encoding the positive and negative effects, respectively, of action a on fluent F at situation s.

[1] For legibility, we typically omit leading universal quantifiers.

Transforming situation calculus histories

In this section we provide the logical foundations for two basic operations that will allow to transform an action history in a way that is logically correct with respect to a basic action theory $\mathcal{D}$.

Swapping consecutive actions

We start with the logical specification of the notion of consecutive actions being *swappable* in an action history of the form $do(\delta, S_0)$, where δ is a sequence of ground actions. We will introduce two versions of the notion, *just-in-time swappable* and *always swappable*. For the following definitions let $\mathcal{D}$ be a basic action theory of the type we introduced in the previous section, and let δ be $\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle$, where k is in $\{1, \ldots, n-1\}$, and each of the elements of δ is a ground action term.

Definition 1. We say that actions α_k and α_{k+1} are *just-in-time swappable* in δ wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:

$$\mathcal{D} \models \forall \vec{x}.\ F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1} \rangle, S_0)) \equiv F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k+1}, \alpha_k \rangle, S_0)).$$

So, we can swap actions α_k and α_{k+1} in δ as long as in all models of $\mathcal{D}$ it happens that the interpretation of all fluent atoms is preserved in the two versions of the action history. The *just-in-time* terminology is used to stress that it may be due to the axioms in this particular instance of $\mathcal{D}_0$ and the action sequence δ that ensure that this condition holds, while in fact actions α_k and α_{k+1} would not necessarily be swappable if another $\mathcal{D}_0$ and δ were assumed.

Example 1. Consider the case of fluent $F(s)$ that is only affected by actions set_F and $unset_F$ such that set_F has a conditional effect that makes $F(s)$ true when $G(s)$ holds, and $unset_F$ has the conditional effect that makes $F(s)$ false when $G(s)$ holds. The SSA for $F(s)$ is then as follows:

$$F(do(a, s)) \equiv (a = set_F \wedge G(s)) \vee F(s) \wedge \neg(a = unset_F \wedge G(s)).$$

Consider now an action sequence δ as above in which action α_k is set_F and α_{k+1} is $unset_F$, assuming there are also other fluents and actions in the language. By looking at the SSA for $F(s)$ we can conclude that these actions may have "conflicting" effects and therefore they should not be considered swappable in general, as the order of applying these two actions may result to a different truth value for $F(s)$.

Nonetheless, if we also take into account $\mathcal{D}_0$ and the particular course of action that takes us to the situation where α_k and α_{k+1} are performed, we may conclude that we can actually swap the actions. For instance, consider a $\mathcal{D}_0$ that includes the fact $\neg G(S_0)$, and the following SSA for $G(s)$:

$$G(do(a, s)) \equiv a = set_G \vee G(s) \wedge \neg(a = unset_G).$$

In the case that none of the actions $\alpha_1, \ldots, \alpha_{k-1}$ is set_G then condition $G(s)$ is in fact not satisfied in the situation when α_k is performed, and also in the resulting situation in which α_{k+1} is performed. As a result, actions α_k and α_{k+1}

have no effect in the truth value of $F(s)$, and actually no effects whatsoever in δ in every model of $\mathcal{D}$. For the same reason, reversing the order of these two actions also results in no effects in the truth value of any fluent atom, and we can conclude that they can be safely swapped, or more precisely that α_k and α_{k+1} are just-in-time swappable in δ wrt $\mathcal{D}$.[2]

Note then that if none of the actions $\alpha_1, \ldots, \alpha_{k-1}$ is $unset_G$ and exactly one is set_G then condition $G(s)$ is satisfied in the situation when α_k is performed, and also in the resulting situation in which α_{k+1} is performed. As a result, actions α_k and α_{k+1} have conflicting effects and are not just-in-time swappable in δ wrt $\mathcal{D}$.

Definition 1 is a precise characterization of when two consecutive actions may be swapped in a particular action sequence without affecting the interpretations of fluents. In order to check this in general one needs to look into the properties of the situation where the actions are performed and verify that no conflict arises. As we saw in Example 1 the same actions for the same $\mathcal{D}$ may be just-in-time swappable in one sequence of actions and not in another.

There can be also cases of actions that the actual situation in which they are performed does not play any role as for example actions that may only affect different fluents. We can capture this notion which we call *always swappable* with a similar definition that removes the dependency on the particular $\mathcal{D}_0$ the action sequence δ as follows.

Definition 2. We say that actions α and α' are *always swappable* wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:

$$\mathcal{D} - \mathcal{D}_0 \models \forall \vec{x}.\, F(\vec{x}, do(\langle \alpha, \alpha' \rangle, S_0)) \equiv F(\vec{x}, do(\langle \alpha', \alpha \rangle, S_0)).$$

Essentially, Definition 2 requires that starting from any possible extension for all fluents, applying the two actions in question will have the same effect regardless of the order the actions are applied.

Example 2. Actions set_F and $unset_F$ are not always swappable wrt $\mathcal{D}$, and also the same holds for all α, α' such that $\alpha \in \{set_F, unset_F\}$ and $\alpha' \in \{set_G, unset_G\}$. On the other hand, if we assume that $\mathcal{D}$ also includes the following successor state axiom for H, $H(do(a, s)) \equiv a = set_H \vee H(s) \wedge \neg(a = unset_H)$, then for all α, α' such that $\alpha \in \{set_F, unset_F, set_G, unset_G\}$ and $\alpha' \in \{set_H, unset_H\}$, α, α' are always swappable wrt $\mathcal{D}$.

As we will see later, this distinction will become important when we specify algorithmic ways to decide whether consecutive actions are swappable. Next, we show some straightforward results about projection that follow from these definitions. The *simple projection* problem is deciding whether a condition about a particular situation in the future holds (Reiter 2001), and the next theorem shows that swapping actions according to our definitions preserves the result for any extension of the action sequence in the question.

Theorem 1. *Let δ be $\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle$, where k is in $\{1, \ldots, n-1\}$, and each of the elements of δ is a ground action term. Let $\phi(s)$ be a (first-order or second-order) formula in $\mathcal{L}$ that is uniform in s. If actions α_k and α_{k+1} are always swappable wrt $\mathcal{D}$ or just-in-time swappable in δ wrt $\mathcal{D}$ then the following holds:*

$$\mathcal{D} \models \phi(do(\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle \cdot \zeta, S_0)) \equiv \phi(do(\langle \alpha_1, \ldots, \alpha_{k+1}, \alpha_k, \ldots, \alpha_n \rangle \cdot \zeta, S_0)),$$

where ζ is any sequence of ground action terms.

Proof. For the case of just-in-time swappable we work as follows. Let M be an arbitrary model of $\mathcal{D}$. We show by induction on the situation terms σ such that $do(\langle \alpha_1, \ldots, \alpha_{k-1} \rangle, S_0) \sqsubseteq \sigma$, that for all fluents F and all sequences of action terms ζ, $M \models F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1} \rangle \cdot \zeta, S_0)$ iff $M \models F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k+1}, \alpha_k \rangle \cdot \zeta, S_0)$. For the base case we use Definition 1 and the induction step is straightforward. The theorem then follows by induction on the construction of the formulas ϕ that are uniform in s.

For the case of always swappable let M be an arbitrary model of $\mathcal{D}$. Since $\mathcal{D} - \mathcal{D}_0$ admits all possible extensions for the fluents in S_0 it follows that there is a model M' of $\mathcal{D} - \mathcal{D}_0$ with the same domain for all sorts as M such that for all μ and for all fluents F, $M, \mu \models F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k-1} \rangle, S_0))$ iff $M', \mu \models F(\vec{x}, S_0)$. By Definition 2 then it follows that $M \models F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1} \rangle, S_0)$ iff $M \models F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k+1}, \alpha_k \rangle, S_0)$, from which point we proceed as in the case of just-in-time swappable actions. □

Theorem 1 shows that swapping actions that are just-in-time or always swappable in δ preserves the entailment of any projection query that refers to some particular situation that comes after $do(\delta, S_0)$. This generalizes to more expressive sentences that refer to situations that come after $do(\delta, S_0)$, not necessary being uniform in one situation.

For example, we can show that swapping actions preserves sentences that may also quantify over future situations such as in the set $\mathcal{L}^F_\sigma$ defined in (Vassos and Levesque 2013), which is a form of the so-called *generalized projection task*. For σ being $do(\alpha, S_0)$, an example of a sentence in $\mathcal{L}^F_\sigma$ is the following: $\forall s(do(\alpha, S_0) \sqsubseteq s \supset \psi(s))$, which states that after executing action α in S_0 then $\psi(s)$ remains true always for all the future situations s from that point on.

Theorem 2. *Let $\mathcal{L}^F_\sigma$ be the set of first-order formulas that refer to the future of σ as defined in (Vassos and Levesque 2013) that allows limited quantification over situations, and $\phi(\sigma)$ be a sentence in $\mathcal{L}^F_\sigma$ for σ being $do(\delta, S_0)$. If actions α_k and α_{k+1} are always or just-in-time swappable in δ wrt $\mathcal{D}$ then the following holds:*

$$\mathcal{D} \models \phi(do(\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle \cdot \zeta, S_0)) \equiv \phi(do(\langle \alpha_1, \ldots, \alpha_{k+1}, \alpha_k, \ldots, \alpha_n \rangle \cdot \zeta, S_0)),$$

where ζ is any sequence of ground action terms.

Proof. Similar to the proof of Theorem 1 except that we do induction on the construction of the formulas ϕ in $\mathcal{L}^F_\sigma$. □

Also it is easy to show that always swappable implies just-in-time swappable but not the opposite.

[2] In fact they could also be eliminated as we will see later.

Corollary 1. *If actions α_k and α_{k+1} are always swappable wrt $\mathcal{D}$ then for any sequence of actions δ in which they appear consecutively they are just-in-time swappable in δ wrt $\mathcal{D}$, but the opposite is not true.*

We now turn our attention to how swappable actions may be detected in practice. Definitions 1 and 2 provide a semantical account of the requirement that characterizes when two actions can be safely swapped in an action sequence. In particular, as we noted earlier, for just-in-time swappable actions we need to be able to project on the situation where the actions are applied and check the effects of actions conditioned on the action history. On the other hand, the notion of actions being always swappable is decoupled from the action history, allowing us to check for a simpler condition. The following theorem shows an equivalent characterization of always swappable actions in terms of the effects of the actions and their successor state axioms.

Theorem 3. *Actions α and α' are always swappable wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:*

$$\begin{aligned}\mathcal{D} - \mathcal{D}_0 \models \forall \vec{x}\{&(\gamma_F^+(\vec{x}, \alpha, S_0) \vee \gamma_F^+(\vec{x}, \alpha', do(\alpha, S_0)) \\ &\equiv \gamma_F^+(\vec{x}, \alpha', S_0) \vee \gamma_F^+(\vec{x}, \alpha, do(\alpha', S_0))) \\ &\wedge (\gamma_F^-(\vec{x}, \alpha, S_0) \vee \gamma_F^-(\vec{x}, \alpha', do(\alpha, S_0)) \\ &\equiv \gamma_F^-(\vec{x}, \alpha', S_0) \vee \gamma_F^-(\vec{x}, \alpha, do(\alpha', S_0)))\} \\ \wedge \neg\exists \vec{x}\{&\gamma_F^+(\vec{x}, \alpha, S_0) \wedge \gamma_F^-(\vec{x}, \alpha', S_0) \\ &\vee \gamma_F^+(\vec{x}, \alpha', S_0) \wedge \gamma_F^-(\vec{x}, \alpha, S_0)\}.\end{aligned}$$

Proof (sketch). Observe that two actions are not always swappable only if in some model the application of one action in S_0 changes the truth value of the effect formulas of the other action or if they have conflicting effects. The first condition is covered by the universally quantified formula that covers all the cases, and the second condition is covered by the negated existentially quantified formula.

In the case of *context-free* actions such that γ_F^+ and γ_F^- do not depend on s, this is simplified as follows.

Corollary 2. *Let $\mathcal{D}$ be a basic action theory such that all successor state axioms are context-free in the sense that they have the following form: $F(\vec{x}, do(a, s)) \equiv \gamma_F^+(\vec{x}, a) \vee F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a)$. Actions α and α' are always swappable wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:*

$$\begin{aligned}\mathcal{D} - \mathcal{D}_0 \models \neg\exists \vec{x}\{&\gamma_F^+(\vec{x}, \alpha, S_0) \wedge \gamma_F^-(\vec{x}, \alpha', S_0) \\ &\vee \gamma_F^+(\vec{x}, \alpha', S_0) \wedge \gamma_F^-(\vec{x}, \alpha, S_0)\}.\end{aligned}$$

These results allow us to reduce the semantic notion expressed in Definition 2 into a more concrete one about the effects of actions as expressed in the positive and negative effect formulas γ^+ and γ^- in the successor state axioms of the fluents in $\mathcal{L}$. Even though this condition is still an entailment question about the effects of actions, as we will see later for special cases of successor state axioms we will be able to further reduce it to simpler tests that we can easily evaluate.

Next we proceed to a similar analysis for actions that can be considered redundant and can be safely omitted.

Eliminating actions

Now we turn our attention to simplifying the action history by *eliminating* an action α_k that is *dominated* by actions $\alpha_{k+1}, \ldots, \alpha_n$ in a sequence of ground action terms δ of the form $\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle$. As in the case of swappable actions, we identify two versions for this notion.

Definition 3. We say that action α_k is *just-in-time dominated* by α_{k+1} in δ wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:

$$\begin{aligned}\mathcal{D} \models \forall \vec{x}.\, F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k-1}, \alpha_k, \alpha_{k+1} \rangle, S_0)) \equiv \\ F(\vec{x}, do(\langle \alpha_1, \ldots, \alpha_{k-1}, \alpha_{k+1} \rangle, S_0)).\end{aligned}$$

Example 3. Consider the fluent $F(x, s)$ with the following successor state axiom:

$$\begin{aligned}F(x, do(a, s)) \equiv\ & a = set_F(x) \vee a = resetTrue_F \wedge G(s) \vee \\ & F(s) \wedge \neg(a = unset_F(x) \vee a = resetFalse_F).\end{aligned}$$

Action $set_F(c)$ sets the truth value of atom $F(c, s)$ to true, $unset_F(x)$ sets it to false, and there are two resetting actions that set all atoms to true and false. Action $resetTrue_F$ is conditional and only has this global effect when $G(s)$ holds, while $resetFalse_F$ is unconditional. Assume that $G(s)$ has the same successor state axiom as in Example 1. Consider the action δ: $\langle unset_G, unset_F(c), set_F(c), resetTrue_F \rangle$. Then, $unset_F(c)$ is just-in-time dominated by $set_F(c)$ in δ wrt to $\mathcal{D}$, but $set_F(c)$ is not just-in-time dominated by $resetTrue_F$ in δ wrt to $\mathcal{D}$.

Similarly to always swappable actions we introduce the notion of *always dominated* actions that removes the dependency on the particular $\mathcal{D}_0$ and the action sequence δ as follows.

Definition 4. We say that action α is *always dominated* by α' wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:

$$\mathcal{D} - \mathcal{D}_0 \models \forall \vec{x}.\, F(\vec{x}, do(\langle \alpha, \alpha' \rangle, S_0) \equiv F(\vec{x}, do(\alpha', S_0)).$$

Example 4. Consider the action sequence δ': $\langle set_G, unset_F(c), set_F(c), resetFalse_F \rangle$. Then $set_F(c)$ is always dominated by $resetFalse_F$ in δ wrt to $\mathcal{D}$.

Similar to the analysis of swappable actions, the notions of just-in-time and always dominated actions preserve the entailment of first-order (and second-order) sentences that refer to situations after δ. The next theorem shows this for sentences of the simple projection problem like Theorem 1, and a similar theorem can be obtained for more expressive sentences for the generalized progression problem such as the one addressed in Theorem 2.

Theorem 4. *Let δ be $\langle \alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle$, where k is in $\{1, \ldots, n-1\}$, and each of the elements of δ is a ground action term. Let $\phi(s)$ be a (first-order or second-order) formula in $\mathcal{L}$ that is uniform in s. If action α_k is always dominated by α_{k+1} wrt $\mathcal{D}$ or just-in-time dominated by α_{k+1} in δ wrt $\mathcal{D}$ then the following holds:*

$$\begin{aligned}\mathcal{D} \models \phi(do(\langle \alpha_1, \ldots, \alpha_{k-1}, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n \rangle \cdot \zeta, S_0)) \equiv \\ \phi(do(\langle \alpha_1, \ldots, \alpha_{k-1}, \alpha_{k+1}, \ldots, \alpha_n \rangle \cdot \zeta, S_0)),\end{aligned}$$

where ζ is any vector of ground action terms.

Also it is easy to show that always domination implies just-in-time domination but not the opposite.

Corollary 3. *If action α_k is always dominated by α_{k+1} wrt $\mathcal{D}$ then for any vector of actions δ in which they appear consecutively and α_{k+1} is after α_k, α_k is just-in-time dominated by α_{k+1} in δ wrt $\mathcal{D}$, but the opposite is not true.*

The following theorem shows an equivalent characterization of always domination of actions in terms of the effects of the actions and their successor state axioms.

Theorem 5. *Action α is always dominated by α' wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:*

$$\begin{aligned}\mathcal{D} - \mathcal{D}_0 \models \forall \vec{x}\{&\gamma_F^+(\vec{x}, \alpha', S_0) \equiv \gamma_F^+(\vec{x}, \alpha', do(\alpha, S_0)) \\ &\wedge \gamma_F^-(\vec{x}, \alpha', S_0) \equiv \gamma_F^-(\vec{x}, \alpha', do(\alpha, S_0))\} \\ \wedge \forall \vec{x}\{&(\gamma_F^+(\vec{x}, \alpha, S_0) \vee \ \gamma_F^-(\vec{x}, \alpha, S_0)) \\ &\supset (\gamma_F^+(\vec{x}, \alpha', S_0) \vee \gamma_F^-(\vec{x}, \alpha', S_0))\}.\end{aligned}$$

Proof (sketch). Observe that α is not always dominated by α' only if in some model the application of α in S_0 changes the truth value of the effect formulas of α' or if some of the effects of α is not reset by α'.

In the case of *context-free* actions such that γ_F^+ and γ_F^- do not depend on s, this is simplified as follows.

Corollary 4. *Let $\mathcal{D}$ be a basic action theory such that all successor state axioms are context-free in the sense that they have the following form: $F(\vec{x}, do(a, s)) \equiv \gamma_F^+(\vec{x}, a) \vee F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a)$. Action α is always dominated by α' wrt $\mathcal{D}$ iff for all fluents $F(\vec{x}, s)$ in $\mathcal{L}$ the following holds:*

$$\begin{aligned}\mathcal{D} - \mathcal{D}_0 \models \forall \vec{x}\{&(\gamma_F^+(\vec{x}, \alpha, S_0) \vee \ \gamma_F^-(\vec{x}, \alpha, S_0)) \\ &\supset (\gamma_F^+(\vec{x}, \alpha', S_0) \vee \gamma_F^-(\vec{x}, \alpha', S_0))\}.\end{aligned}$$

These notions provide the logical specification for transforming and simplifying action histories in a way that preserves the entailment of first-order projection queries. What becomes interesting then is identifying efficient procedures that can identify when sets of actions can be swapped and when an action can be eliminated for certain classes of action theories $\mathcal{D}$.

Action theories with single-value fluents and resetting actions

We now turn our attention to basic action theories with relational fluents that have a function-like behavior in the following sense. For each fluent $F(\vec{y}, s)$, we distinguish the last argument of sort object as the *output* of the fluent and the rest of the arguments of sort object as the *input*. In order to make this explicit we will write $F(\vec{x}, v, s)$ where v is the the output and $\vec{x}$ is input. We will also require that in every model of the action theory for every input $\vec{x}$ there is exactly one object for v (up to equality) such that $F(\vec{x}, v, s)$ holds. We call fluents of this type *single-value fluents*.

Definition 5. *Single-value* fluents are fluents for which the following holds:

$$\mathcal{D} \models \forall \vec{x}, v\{F(\vec{x}, v, s) \supset \neg\exists v'(F(\vec{x}, v', s) \wedge v' \neq v)\}.$$

This is a very common case in many practical domains, such as those involving mobile agents (i.e. a mobile agent can be in only one position in any given situation).[3] The next definition introduces the single-value successor state axioms which preserve this property.

Definition 6. The successor state axiom for $F(\vec{x}, v, s)$ has *resetting-actions* iff $\gamma_F^+(\vec{x}, v, a, s)$ is a disjunction of formulas of the form:

$$\exists \vec{z}(a = A(\vec{y}) \wedge v = y_i \wedge \phi(\vec{y}, s)),$$

and $\gamma_F^-(\vec{x}, v, a, s)$ is a disjunction of formulas of the form:

$$\exists \vec{z}(a = A(\vec{y}) \wedge \neg v = y_i \wedge \phi(\vec{y}, s)),$$

where A is an action symbol, $\vec{y}$ contains $\vec{x}$, $\vec{z}$ corresponds to the remaining variables of $\vec{y}$, y_i is a variable in $\vec{y}$, and $\phi(\vec{y}, s)$ (called a *context formula*) is a first-order formula uniform in s, and γ_F^+,γ_F^- come in pairs in the sense that for each disjunct in γ_F^+ there is one in γ_F^- that is identical except for the atom $v = y_i$ and vice versa. As in the case of functional fluents in (Reiter 2001), we also require the following sentence be entailed by $\mathcal{D}$:

$$\mathcal{D} \models \gamma_F^+(\vec{x}, v, \alpha, s) \supset \neg\exists v'(v' \neq v \wedge \gamma_F^-(\vec{x}, v', \alpha, s)).$$

A basic action theory $\mathcal{D}$ is one of resetting-actions if every SSAs in $\mathcal{D}_{ss}$ is a resetting SSA.

Note that the positive effects of a resetting SSA are in fact local-effect as they have exactly the same form, requiring that any affected atom is built using arguments of action $A(\vec{y})$, while the negative effects are a special form of global effects. Similarly to (Liu and Levesque 2005), the instantiation of a local-effect SSA on a ground action term can be simplified using the unique names axioms for actions as follows.

Lemma 1. *Let α be the ground action term $A(\vec{e})$, where $\vec{e}$ is a vector of constants and suppose that the SSA for F is resetting. Then $\gamma_F^+(\vec{x}, v, \alpha, s)$ is logically equivalent to a formula of the following form:*

$$(\vec{x} = \vec{c}_1 \wedge v = d_1 \wedge \phi_1(s)) \vee \cdots \vee (\vec{x} = \vec{c}_m \wedge v = d_m \wedge \phi_m(s)),$$

where each of the $\vec{c}_1, \ldots, \vec{c}_m$ is a vector of constants contained in $\vec{e}$, and $\phi_1(s), \ldots, \phi_m(s)$ are first-order and uniform in s. Each combination of $\vec{c}_i$, $\vec{d}_i$ must be distinct, and for any $\vec{x} = \vec{c}_i$, at most one $\phi_i(s)$ can hold for any model of s. Similarly, $\gamma_F^-(\vec{x}, \alpha, s)$ is logically equivalent to a formula of the following form:

$$\begin{aligned}&(\vec{x} = \vec{c}_1 \wedge \neg v = d_1 \wedge \phi_1(s)) \vee \cdots \\ &\vee (\vec{x} = \vec{c}_m \wedge \neg v = d_m \wedge \phi_m(s)).\end{aligned}$$

An example follows.

Example 5. Consider a modified instance of the Sokoban problem such that the surface of any cell can be either clean or oily. Consider three actions: $move(x)$, which moves the

[3] We note that a "regular" relational fluent could still be expressed by means of special constants *true* and *false* and with some extra syntactical machinery.

player to position x, $push(b, x, y)$, which pushes block b from position x to position y and $clean(x, t)$, which sets the truth value of x being oily to t. We use three fluents: *PlAt*(x, s), which says the player is in position x in situation s, *BlAt*(b, x, s), which says that block b is in position x in situation s and *Oily*(x, t, s), which says the surface of x contains oil where t is a variable that will take an object representing True or False. There are also appropriate $Poss$ axioms for walls, connectivity, and other necessary restrictions. The successor state axioms for this problem are as follows:

$$\begin{aligned}
&\textit{PlAt}(v, do(a, s)) \equiv \exists x(a = move(x) \wedge v = x)\\
&\quad \vee \exists b, x, y(a = push(b, x, y) \wedge v = x)\\
&\quad \vee \textit{PlAt}(v, s) \wedge \neg(\exists x(a = move(x) \wedge \neg v = x)\\
&\quad \vee \exists b, x, y(a = push(b, x, y) \wedge \neg v = x))\\
&\textit{BlAt}(b, v, do(a, s)) \equiv \exists x, y(a = push(b, x, y) \wedge v = y)\\
&\quad \vee \textit{BlAt}(b, v, s) \wedge \neg\exists x, y(a = push(b, x, y) \wedge \neg v = y)\\
&\textit{Oily}(x, v, do(a, s)) \equiv \exists t(a = clean(x, t) \wedge v = t))\\
&\quad \vee \textit{Oily}(x, v, s) \wedge \neg\exists t(a = clean(x, t) \wedge \neg v = t))
\end{aligned}$$

We begin by introducing the notion of the *effect set* of an action. The idea is to represent a set of situation-suppressed ground fluent atoms that will be affected by an action, as well as the conditions under which the fluent atom will be affected. This is formalised as follows:

Definition 7. Let α_k be a ground resetting action. The *effect set* of α_k is the following set:

$$\begin{aligned}
\Phi_k = \{\phi \supset F(\vec{c}, d) \mid\\
\vec{x} = \vec{c} \wedge \phi \wedge v = d \text{ appears in } \gamma_F^+(\vec{x}, v, \alpha, s)\}.
\end{aligned}$$

Without loss of generality, we assume that the ground versions of the γ^+ and γ^- formulas are simplified according to Lemma 1. The effect sets can then be obtained directly. We observe the following fundamental property of the effect set:

Lemma 2. *Let $F(\vec{x}, v, s)$ be a single-value fluent with positive and negative successor state axioms $\gamma_F^+(\vec{x}, v, \alpha, s)$ and $\gamma_F^-(\vec{x}, v, \alpha, s)$ respectively. Let M be any model of $\mathcal{D}$. Let α_k be a ground resetting action with effect set Φ_k. Then for all $\vec{x} = \vec{c}, v = d$: $M \models \gamma_F^+(\vec{x}, v, \alpha_k, s)$ iff there exists a $\phi \supset F(\vec{c}, d)$ in Φ_k such that $M \models \phi$.*
$M \models \gamma_F^-(\vec{x}, v, \alpha_k, s)$ iff there exists a $\phi \supset F(\vec{c}, e)$ in Φ_k such that $d \neq e$ and $M \models \phi$.

Proof. From the construction of the effect set and the form of the successor state axioms. □

Resetting actions without contexts

We begin by considering a subset of resetting actions where each ϕ_i is $true$. This allows us to reorder or eliminate actions without needing to consider the effects that one action can have on the contexts of the other. We introduce a procedure for determining whether two context-free actions are swappable by directly inspecting their effect sets:

Theorem 6. *Let actions α_1 and α_2 be consecutive resetting actions without contexts with effect sets Φ_1 and Φ_2 respectively. Then α_1 and α_2 are always swappable iff for all elements of Φ_1 of the form $true \supset F(\vec{c}, d_1)$ there is no element of Φ_2 of the form $true \supset F(\vec{c}, d_2)$ such that $d_1 \neq d_2$.*

Proof. From Corollary 2 and Lemma 2. □

While this is a restricted class of actions, it is nonetheless sufficient to cover a number of interesting cases, such as the Sokoban example.

Example 6. Consider the SSAs used in Example 5 and a sample action sequence $\langle move(pos_1), clean(pos_1, F)\rangle$. We first calculate the effect sets for each action: $\Phi_1 = \{true \supset \textit{PlAt}(pos_1)\}$, $\Phi_2 = \{true \supset \textit{Oily}(pos_1, F)\}$. From Theorem 6 we can observe that $move(pos_1)$ and $clean(pos_1, F)$ are always swappable.

We now introduce a procedure for determining whether one resetting action without a context always dominates another by directly inspecting their effect sets.

Theorem 7. *Let actions α_1 and α_2 be consecutive resetting actions without contexts with effect sets Φ_1 and Φ_2 respectively. Then α_2 always dominates α_1 iff for all elements of Φ_1 of the form $true \supset F(\vec{c}, d_1)$ there exists an element of Φ_2 of the form $true \supset F(\vec{c}, d_2)$.*

Proof. From Corollary 4 and Lemma 2 □

Note that determining whether actions can be just-in-time eliminated purely by inspection of the successor state axioms is not possible. This is because of the possibility that some or all of those fluent atoms already held the same value before the action was taken. In the extreme case, for example, that an action occurred twice and identically set all fluent atoms, the second occurrence of that action could be said to be dominated by *any* subsequent action.

Below is an example demonstrating the idea of always-dominating for resetting actions without contexts.

Example 7. Consider the SSAs used in Example 5 and a sample action sequence $\langle move(pos_1), push(b_1, pos_2, pos_3)\rangle$. We first calculate the effect sets for each action: $\Phi_1 = \{true \supset \textit{PlAt}(pos_1)\}$, $\Phi_2 = \{true \supset \textit{PlAt}(pos_2), true \supset \textit{BlAt}(b_1, pos_2)\}$. From Theorem 7 we can observe that $push(b_1, pos_2, pos_3)$ dominates $move(pos_1)$ and thus the action sequence can be simplified to $\langle push(b_1, pos_2, pos_3)\rangle$.

Resetting actions with conjunctive queries as contexts

We now consider a larger class of resetting actions where the context formulas ϕ_i consist of a quantifier-free conjunction of ground fluent atoms. We require that each context formula ϕ_i is satisfiable in the sense that if an element $F(\vec{c}, d)$ is mentioned then neither $\neg F(\vec{c}, d)$ nor $F(\vec{c}, e)$ is mentioned for some $e \neq d$. Reasoning about the properties of these actions is not so straightforward, as determining the value of a context formula for an action requires detailed inspection of the knowledge base. Moreover, the application of one action

can affect the contexts of another. We introduce the following sound but incomplete method for determining whether a pair of conjunctive resetting actions are always swappable:

Theorem 8. *Let actions α_1 and α_2 be consecutive conjunctive resetting actions with effect sets Φ_1 and Φ_2 respectively. Then α_1 and α_2 are always swappable if:*

- *For all elements of Φ_1 of the form $\phi_1 \supset F_1(\vec{c_1}, d_1)$, there does not exist an element of Φ_2 of the form $\phi_2 \supset F_2(\vec{c_2}, d_2)$ such that $F_1 = F_2, \vec{c_1} = \vec{c_2}$, $d_1 \neq d_2$, $\phi_1 \wedge \phi_2$ is satisfiable.*
- *For all elements of Φ_1 of the form $\phi_1 \supset F_1(\vec{c_1}, d_1)$, there does not exist an element of Φ_2 of the form $\phi_2 \supset F_2(\vec{c_2}, d_2)$ such that ϕ_2 mentions $F_1(\vec{c_1}, e)$ for some e.*
- *For all elements of Φ_2 of the form $\phi_2 \supset F_2(\vec{c_2}, d_2)$, there does not exist an element of Φ_1 of the form $\phi_1 \supset F_1(\vec{c_1}, d_1)$ such that ϕ_1 mentions $F_2(\vec{c_2}, e)$ for some e.*

Proof (sketch). The first point above corresponds with the second condition of Theorem 3 and follows from Lemma 2. For the second condition of Theorem 3, we consider a slightly stronger requirement such that $\gamma^+_F(\vec{x}, v, \alpha_1, S_0) \equiv \gamma^+_F(\vec{x}, \alpha_1, do(\alpha_2, S_0))$ and $\gamma^+_F(\vec{x}, v, \alpha_2, S_0) \equiv \gamma^+_F(\vec{x}, \alpha_2, do(\alpha_1, S_0))$ for all $\vec{c}, v$ and similarly for negative SSAs. From Lemma 2, any $F(\vec{c}, d)$ for which there is no $\phi \supset F(\vec{c}, e)$ in Φ_1 for some e cannot affect $\gamma^+_F(\vec{x}, v, \alpha_2, S_0) \equiv \gamma^+_F(\vec{x}, \alpha_2, do(\alpha_1, S_0))$, and similarly for the other conditions.

This result is incomplete due to the possibility that for all models, a pair of context formulas in Φ_1 and Φ_2 referring to the same fluent may be unsatisfiable, or that one may imply the other. We expect such cases to be uncommon in practice, however, as in the majority of cases we expect to reorder actions that do not affect the same fluents. We now consider the complexity of always-swapping for conjunctive queries:

Theorem 9. *The complexity of determining whether two conjunctive resetting actions are always swappable is $O(nm)$ where n is the number of elements in the effect set and m is the number of elements in the context formula ϕ. For resetting actions without contexts, this reduces to $O(n)$.*

Proof. The first item in Theorem 8 requires that for each of the n elements of the effect set, the satisfiability of $\phi_1 \wedge \phi_2$ for any corresponding element in the other set is determined. Since $\phi_1 \wedge \phi_2$ is a conjunctive query, this can be done in $O(nm)$ time. The second and third points require each of the $n \times m$ members of an effect set to be checked for membership in the other effect set, with a complexity of $O(nm)$. For resetting actions without contexts, the second and third points are not required, and the first part does not require a satisfiability check. This reduces to $O(n)$, as the problem is essential checking set membership. □

We also note that the satisfiability check can be omitted while still retaining a sound result. To demonstrate, we consider a version of the Sokoban example used previously, modified to include conjunctive contexts. Imagine now that pushing a box over an oily surface will cause it to move one additional unit further than expected. We modify the formalism as follows:

Example 8. Let the push action now take five arguments, $push(b, x, y, z, t)$ where t is a variable that will take an object representing a truth value. This action will push block b from position x to position y. If the surface of y is oily, the block will be moved to z instead. The successor state axiom for *BlAt* is now as follows:

$$\begin{aligned}
&\textit{BlAt}(b, v, do(a, s)) \equiv \\
&\exists x, y, z(a = push(b, x, y, z, t) \wedge v = y \wedge \neg \textit{Oily}(y, t, s)) \\
&\vee \exists x, y, z(a = push(b, x, y, z, t) \wedge v = z \wedge \textit{Oily}(y, t, s)) \\
&\vee \textit{BlAt}(b, v, s) \wedge \neg\{ \\
&\exists x, y, z(a = push(b, x, y, z, t) \wedge \neg v = y) \wedge \neg \textit{Oily}(y, t, s) \\
&\vee \exists x, y, z(a = push(b, x, y, z, t) \wedge \neg v = z \wedge \textit{Oily}(y, t, s))\}
\end{aligned}$$

Consider now an action sequence $\langle clean(pos_1, F), push(b_1, pos_2, pos_3, pos_4, T)\rangle$. We firstly calculate the effect sets of the two actions: $\Phi_1 = \{true \supset \textit{Oily}(pos_1, F)\}$ $\Phi_2 = \{\neg \textit{Oily}(pos_3, T) \supset \textit{BlAt}(b_1, pos_3), \textit{Oily}(pos_3, T) \supset \textit{BlAt}(b_1, pos_4), true \supset \textit{PlAt}(pos_2)\}$. Now using Theorem 8, we can observe that $clean(pos_1, F)$ and $push(b_1, pos_2, pos_3, pos_4, T)$ are always swappable.

We now propose a method for identifying when one resetting action is always dominated by a subsequent action, and can thus be removed.

Theorem 10. *Let α_1 and α_2 be consecutive conjunctive resetting actions with effect sets Φ_1 and Φ_2 respectively. Let $\phi_1[F(\vec{c})]$ refer to the disjunction of all ϕ_1 for which $\phi_1 \supset F(\vec{c}, d)$ is in Φ_1 for any d and $\phi_2[F(\vec{c})]$ refer to the disjunction of all ϕ_2 for which $\phi_2 \supset F(\vec{c}, d)$ is in Φ_2 for any d. Then α_2 always dominates α_1 iff for all $\vec{c}$, $\phi_1[F(\vec{c})] \supset \phi_2[F(\vec{c})]$ is valid and for all elements of Φ_1 of the form $\phi_1 \supset F_1(\vec{c_1}, d_1)$, there is no element of Φ_2 of the form $\phi_2 \supset F_2(\vec{c_2}, d_2)$ such that ϕ_2 mentions $F_1(\vec{c_1}, e)$ for some e and $\phi_1 \wedge \phi_2$ is satisfiable.*

Proof (sketch). Focus firstly on the second part of Theorem 5. From Lemma 2, $\gamma^-_F(\vec{x}, v, \alpha_1, S_0) \vee \gamma^+_F(\vec{x}, v, \alpha_1, S_0)$ iff there exists a $\phi_1 \supset F(\vec{x}, v')$ in Φ_1 such that ϕ_1 is $true$ (similarly for α_2). Now $(\gamma^-_F(\vec{x}, v, \alpha_1, S_0) \vee \gamma^+_F(\vec{x}, v, \alpha_1, S_0)) \supset (\gamma^-_F(\vec{x}, v, \alpha_2, S_0) \vee \gamma^+_F(\vec{x}, v, \alpha_2, S_0))$ is valid iff $\phi_1[F(\vec{c})] \supset \phi_2[F(\vec{c})]$ is valid for all models. For the first part of Theorem 5, for each $\phi_2 \supset F_2(\vec{c_2}, d_2)$ in Φ_2, consider a model, M such that every $F(\vec{x}, y)$ mentioned in ϕ_2 holds with the exception of some $F(\vec{c}, d)$ for which $\phi_1 \supset F(\vec{c}, d)$ appears in Φ_1. Then $\gamma^+_F(\vec{x}, v, \alpha_2, S_0) \not\equiv \gamma^+_F(\vec{x}, v, \alpha_2, do(\alpha_1, S_0))$ for $x = c_2, v = d_2$ iff ϕ_1 is also satisfiable M. The approach is similar for the case of $F(\vec{c}, e)$ in ϕ_1 as well as for the negative effects.

We now consider the complexity of always-dominating for conjunctive queries:

Theorem 11. *The complexity of determining whether one conjunctive-resetting action dominates another is $O(n.2^{mk})$ where n is the number of elements in the effect set, k is the number of times a $\phi \supset F(\vec{c}, d)$ appears in the effect set for fixed $\vec{c}$ but varying d and m is the number of elements in the context formula ϕ. For resetting actions without contexts, this reduces to $O(n)$.*

Proof. The first condition in Theorem 10 requires that for each of the n elements of the effect set, the validity of $\phi_1[F(\vec{c})] \supset \phi_2[F(\vec{c})]$ is determined. Each $\phi[F(\vec{c})]$ contains $m \times k$ terms, the complexity of determining the validity for each formula of this type is thus $O(2^{mk})$. For the second part, each of the $n \times m$ elements of the contexts in Φ_2 must be checked for membership in Φ_1. If found, the satisfiability of $\phi_1 \wedge \phi_2$ must be determined. Since $\phi_1 \wedge \phi_2$ is a conjunctive query, the satisfiability check can be done in $O(m)$ time. The total complexity is thus $O(n.2^{mk}) + O(nm^2)$. For resetting actions without contexts, the second part is not required, and the first part does not require a validity check. This reduces to $O(n)$, as the problem is essential checking set membership. □

We expect such a method to be feasible in practice, as it is exponential only in the size of the context formulas and the number of times a particular $F(\vec{c})$ appears in the effect set. We now show an example of dominating conjunctive resetting actions.

Example 9. Using the successor state axioms developed in Example 8, consider now an action sequence $\langle push(b_1, pos_1, pos_2, pos_3, T),$
$push(b_1, pos_2, pos_3, pos_4, T)\rangle$. We firstly calculate the effect sets of the two actions: $\Phi_1 = \{\neg \mathit{Oily}(pos_2, T) \supset \mathit{BlAt}(b_1, pos_2), \mathit{Oily}(pos_2, T) \supset \mathit{BlAt}(b_1, pos_3), true \supset \mathit{PlAt}(pos_1)\}$ $\Phi_2 = \{\neg \mathit{Oily}(pos_3, T) \supset \mathit{BlAt}(b_1, pos_3), \mathit{Oily}(pos_3, T) \supset \mathit{BlAt}(b_1, pos_4), true \supset \mathit{PlAt}(pos_2)\}$. Now using Theorem 10, $push(b_1, pos_2, pos_3, pos_4, T)$ always dominates $push(b_1, pos_1, pos_2, pos_3, T)$.

Discussion and Related Work

To the best of our knowledge this is the first approach that looks into formalizing operations over an action history that preserves truth of (simple and generalized) projection queries in the final situation. It helps us handle cases which cannot be easily progressed, or actions which can result in a non-progressible theory, by swapping and eliminating actions—achieving transformed theories which can be later progressed. Note that the work does not extend the known classes of progressable theories (Vassos and Patrizi 2013), in contrast the work contributes to a setting where regression and progression can be invoked and interleaved in sensible ways. It is potentially applicable to a wide range of queries, for different action languages.

A similar approach in spirit is nonetheless employed in the *implementations* of reasoning systems that rely on constraints solvers. In particular, for reasoning about action the `Flux` system for implementing fluent calculus theories (Thielscher 2004) uses (among other things) constraints to represent the effects implied to fluents with arithmetic arguments. These constraints specify how the old value of fluents relates to the new one, and are appended to the constraint store as the action history grows. Periodically, `Flux`'s constraint store invokes heuristic techniques for simplifying the history by eliminating redundant constraints.

There are, nonetheless, parallels to work being done in classical planning. In particular (Chrpa, McCluskey, and Osborne 2012) identifies pairs of actions as being either *independent*, *shared*, *nested* or *interleaved*, and goes on to define conditions under which these different types of actions can be removed from a sequence. Working in the classical planning domain, however, allows Chrpa to consider a clear and finite set of effects for each action, which is not the case in the Situation Calculus. Also, in order to realise the modified plan, Chrpa is required to consider the preconditions of each action, whereas by focusing on the projection problem one is able to bypass these, allowing more actions to be dominated. Finally, Chrpa does not consider the possibility of reordering actions.

Symmetry breaking, and the related issue of planning landmarks, have also been shown to be promising for achieving efficiency in planning domains (Domshlak, Katz, and Shleyfman 2013). Interestingly, dominance relations are known to entail the special case of symmetry in constraint programming (Chu, Garcia del la Banda, and Stuckey 2012). Dominance relations might therefore form a basis of a range of constraint optimisation techniques for action languages. This paper follows these insights by realising re-ordering for action histories for a wide range of rich theories.

There are a number of settings where transformation could be effectively used, including *sensing* actions and reasoning about the domination of *sets of actions*. Sensing actions provide new information about the value of fluents and could dominate earlier actions. It may therefore be possible to delay progression, in the event that progression is not possible in one situation due to missing information about certain fluents, until later when sensing actions can learn fluent values which dominate the effects of earlier actions. If we can delay progression until *after* sensing actions are performed, this could facilitate the progression of the action theory at a later time. The dominance of sets of actions over one another could significantly extend the number of cases under which dominance was possible and facilitate new ways of reordering and elimination on set theoretic grounds.

The generality of our approach has the potential to be utilised for wider classes of action theory. For example, the approach could be extended beyond conjunctive queries towards arbitrary context formulas. The extension of the approach to an even wider class of global effect actions, would also be very useful, as global effect theories are known to be particularly problematic with respect to progression.

Conclusions

This work takes the important step of formalizing the transformation of theories of action in order to optimise frequently executed reasoning tasks. The transformation allows action sequences to be shortened, via the swapping and reordering of independent actions and elimination of dominating actions. This can facilitate easier, more optimised, forms of reasoning. Our technique is potentially applicable to a wide range of rich first order theories.

We have identified a new class of action theory which allows for a restricted form of global effects–resetting actions–which falls outside of any known progressable theory of action. It overcomes problems with progression commonly encountered in the agent reasoning tasks illustrated. Our

technique facilitates a new approach for tackling situations falling outside the known classes of progressable theories, and can also be used for optimising the progression of theories known to be progressable. It also enables more efficient entailment queries to be answered by progressing where possible, only performing regressing where necessary.

Acknowledgements

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. The authors acknowledge support of EU Project FP7-ICT 318338 (OPTIQUE) and Sapienza Award 2013 "Spiritlets" project.

References

Chrpa, L.; McCluskey, T. L.; and Osborne, H. 2012. Determining redundant actions in sequential plans. In *ICTAI*, 484–491.

Chu, G.; Garcia del la Banda, M.; and Stuckey, P. J. 2012. Exploiting subprogem dominance in constraint programming. *Constraints* 1:1–38.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2013. Symmetry breaking: Satisficing planning and landmark heuristics. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 298–302.

Lin, F., and Reiter, R. 1994. Forget it! In Greiner, R., and Subramanian, D., eds., *Working Notes, AAAI Fall Symposium on Relevance*, 154–159. American Association for Artificial Intelligence.

Lin, F., and Reiter, R. 1997. How to progress a database. *Artificial Intelligence* 92(1-2):131–167.

Liu, Y., and Levesque, H. J. 2005. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, 522–527. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

McCarthy, J., and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence* 4:463–502.

Reiter, R. 2001. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*.

Thielscher, M. 2004. FLUX: A logic programming method for reasoning agents. *Theory and Practice of Logic Programing* 5(4-5):533–565.

Vassos, S., and Levesque, H. 2008. On the progression of situation calculus basic action theories: Resolving a 10-year-old conjecture. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, AAAI'08, 1004–1009. Chicago, Illinois, USA: AAAI Press.

Vassos, S., and Levesque, H. J. 2013. How to progress a database III. *Artificial Intelligence* 195:203–221.

Vassos, S., and Patrizi, F. 2013. A classification of first-order progressable action theories in situation calculus. In *Proceedings of the 23rd international joint conference on Artificial intelligence*, IJCAI'13, 1132–1138.

Vassos, S.; Lakemeyer, G.; and Levesque, H. 2008. First-order strong progression for local-effect basic action theories. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning*, KR'08, 662–272. AAAI Press.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

State-Boundedness in Data-Aware Dynamic Systems

Babak Bagheri Hariri
Diego Calvanese **Marco Montali**
Free University of Bozen-Bolzano, Italy
lastname@inf.unibz.it

Alin Deutsch
University of California, San Diego
San Diego, CA USA
deutsch@cs.ucsd.edu

Abstract

Verification of dynamic systems that manipulate data, stored in a database or ontology, has lately received increasing attention. A plethora of recent works has shown that verification of systems working over unboundedly many data is decidable even for very rich temporal properties, provided that the system is *state-bounded*. This condition requires the existence of an overall bound on the amount of data stored in each single state along the system evolution. In general, checking state-boundedness is undecidable. An open question is whether it is possible to isolate significant classes of dynamic systems for which state-boundedness is decidable. In this paper we provide a strong negative answer, by resorting to a novel connection with variants of Petri nets. In particular, we show undecidability for systems whose data component contains unary relations only, and whose action component queries and updates such relations in a very limited way. To contrast this result, we propose interesting relaxations of the sufficient conditions proposed in the concrete setting of Data-Centric Dynamic Systems, building on recent results on chase termination for tuple-generating dependencies.

1 Introduction

The formalization and verification of dynamic systems that manipulate data, stored in a database or ontology, has lately received an increasing attention from both a foundational and a practical point of view (Calvanese, De Giacomo, and Montali 2013). In particular, business process management has progressively shifted its focus from a purely control-flow perspective that under-specifies (if not completely abstracts away) the data component, to a data-aware perspective where data have the same importance as system dynamics (Dumas 2011). This comprehensive view of dynamic systems has given rise to a series of paradigms, such as data-centric workflows (Vianu 2009) and business artifacts (Hull 2008), where the static (i.e., data-related) and dynamic (i.e., process-related) system aspects are jointly modelled and managed.

The verification of a data-aware dynamic system is notoriously challenging: as soon as it can incorporate (possibly unbounded) data from the external environment, e.g., due to interaction with external services or humans, the transition system representing its execution becomes infinite-state, and verification turns out to be undecidable even for propositional reachability properties (Deutsch et al. 2009; Belardinelli, Lomuscio, and Patrizi 2012; Bagheri Hariri et al. 2013b; 2013a). To mitigate this problem, an extensive amount of research has been devoted to find suitable classes of data-aware dynamic systems for which verification of first-order temporal properties becomes decidable. In particular, a plethora of recent works has shown that verification of data-aware dynamic systems working over unboundedly many data is decidable even for very rich temporal logics, provided that the system is *state-bounded*. For example, Belardinelli, Lomuscio, and Patrizi (2012) show decidability of verification of state-bounded (there called b-bounded) artifact centric multiagent systems (ACMASs) for a first-order, epistemic variant of CTL, with active domain quantification that applies across time points. Bagheri Hariri et al. (2013b) give a key decidability result for the verification of state-bounded Data-Centric Dynamic Systems (DCDSs) against a first-order variant of the μ-calculus with a limited form of quantification across time. Within the research line of reasoning about actions, De Giacomo, Lesperance, and Patrizi (2012) show that (state-)bounded Situation Calculus theories can be verified against a first-order variant of the μ-calculus, without quantification across states. Notably, in all these cases state-boundedness guarantees the existence of a faithful (sound and complete) finite-state abstraction of the system, paving the way for the application of standard model checking tools.

Intuitively, state boundedness requires the existence of an overall bound on the amount of data stored in each single state of the system (a database instance or an ABox) along the system evolution. As shown by Bagheri Hariri et al. (2013b), checking state-boundedness is in general undecidable. The proof is given for the specific framework of DCDSs, but can be straightforwardly applied to other similar frameworks, such as ACMASs. Due to undecidability of state-boundedness, the existing works have either: *(i)* assumed that the system is state-bounded (Belardinelli, Lomuscio, and Patrizi 2012); *(ii)* explicitly enforced state-boundedness by blocking those actions that would lead to exceed the bound (De Giacomo, Lesperance, and Patrizi 2012; Solomakhin et al. 2013); *(iii)* studied sufficient conditions that can be checked over the dynamic component of the system (actions and/or processes) to test whether they guarantee state-boundedness (Bagheri Hariri et al. 2013b).

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

An open question is whether it is possible to isolate significant classes of dynamic systems for which state-boundedness is decidable. In this paper we study this problem, devising a novel connection between variants of Petri nets (Transfer/Reset nets) (Dufourd, Finkel, and Schnoebelen 1998; Dufourd, Jancar, and Schnoebelen 1999) and DCDSs, and inter-reducing the problem of checking whether a DCDS is state-bounded to the problem of checking whether a Transfer/Reset/Petri net is bounded (i.e., ensures that the number of tokens present in each place of the net never exceeds a given bound). Observe that DCDSs are expressively equivalent to business artifact formalisms for business process specification deployed in industrial settings (Bagheri Hariri et al. 2013b; Solomakhin et al. 2013). Thanks to the connection between DCDSs and Petri nets, we provide a strong negative answer to the aforementioned open question: we show undecidability of DCDSs with unary relations, queried in a very limited way (i.e., without using joins, and with a very limited usage of inequalities). Furthermore, we show that undecidability is mainly related to the possibility for relations to lose their content during the application of a single action, and that decidability is obtained if this is prevented (though with an intractable complexity lower bound). Also in this case, the provided results can be transferred to other similar frameworks, such as ACMASs.

To contrast this negative result, we improve the sufficient (syntactic) conditions proposed by Bagheri Hariri et al. (2013b) to check whether a DCDS is state-bounded, by defining a hierarchy of sufficient conditions that progressively refine the analysis of how data are manipulated by the action component of the DCDS. To do so, we take inspiration from recent results on chase termination for tuple-generating dependencies, and in particular from the *safety* (Meier, Schmidt, and Lausen 2009) and *stratified* (Deutsch, Nash, and Remmel 2008) extensions of *weak-acyclicity* (Fagin et al. 2005).

It is interesting to observe that all our results can be lifted from the pure relational setting we consider here, to a knowledge-based setting as the one presented in (Bagheri Hariri et al. 2013a). This can be immediately done when the data component is a full-fledged ontology expressed in a Description Logic that allows for rewriting queries posed over the ontology as unions of conjunctive queries posed over the data only (Calvanese et al. 2007).

2 Data-Centric Dynamic Systems

We recall the main aspects of Data-Centric Dynamic Systems (DCDSs). For a comprehensive treatment, the reader is referred to (Bagheri Hariri et al. 2013b).

A DCDS $\mathcal{S}$ is a pair $\langle \mathcal{D}, \mathcal{P} \rangle$, where $\mathcal{D}$ is the data component of $\mathcal{S}$, and $\mathcal{P}$ is its process component. The data component $\mathcal{D}$ is a tuple $\mathcal{D} = \langle \mathcal{C}, \mathcal{R}, \mathcal{E}, I_0 \rangle$, where: *(i)* $\mathcal{C}$ is a countably infinite set of constants; *(ii)* $\mathcal{R}$ is a *database schema*, i.e., a sets of relation schemas; *(iii)* $\mathcal{E}$ is a set of *equality constraints* over $\mathcal{R}$; *(iv)* I_0 is the *initial database instance* of $\mathcal{S}$, i.e., a database instance conforming to $\mathcal{R}$ and $\mathcal{E}$, and made up of values in $\mathcal{C}$. Given a relation $R \in \mathcal{R}$ and a database instance I conforming to $\mathcal{R}$, we denote with $|R|_I$ the number of R-tuples contained in I.

The process layer $\mathcal{P}$ defines the progression mechanism for the DCDS. It is constituted by a process, which queries the current data maintained by $\mathcal{D}$ and determines which actions are executable; actions, in turn, query and update $\mathcal{D}$, possibly introducing new values from the external environment, by issuing service calls. Technically, $\mathcal{P} = \langle \mathcal{F}, A, \varrho \rangle$, where *(i)* $\mathcal{F}$ is a finite set of *functions*, each representing the interface to a (nondeterministic) *external service*; *(ii)* A is a finite set of *actions*, whose execution updates the data layer, and may involve external service calls; *(iii)* ϱ is a finite set of *condition-action rules* that form the specification of the overall *process*, which tells at any moment which actions can be executed. By considering a finite set $\mathfrak{A}$ of action names, an *action* $\alpha \in A$ is an expression $\alpha(p_1, \ldots, p_n) : \{e_1, \ldots, e_m\}$, where: *(i)* $\alpha(p_1, \ldots, p_n)$ is its *signature*, constituted by a name $\alpha \in \mathfrak{A}$ and a sequence $p_1, \ldots, p_n$ of *parameters*, to be substituted with values when the action is invoked, and *(ii)* $\{e_1, \ldots, e_m\}$, also denoted as $\text{EFFECT}(\alpha)$, is a set of *specifications of effects*, which are assumed to take place simultaneously. Each e_i has the form $q_i^+ \wedge Q_i^- \rightsquigarrow E_i$, where:

- $q_i^+ \wedge Q_i^-$ is a query over $\mathcal{R}$ whose terms are variables, action parameters, and constants from I_0, where q_i^+ is a union of conjunctive queries, and Q_i^- is an arbitrary FO formula whose free variables are among those of q_i^+. Intuitively, q_i^+ selects the tuples to instantiate the effect with, and Q_i^- filters away some of them.
- E_i is the effect, i.e., a set of facts for $\mathcal{R}$, which includes as terms: terms in the active domain of I_0, free variables of q_i^+ and Q_i^- (including action parameters), and Skolem terms formed by applying a function $f \in \mathcal{F}$ to one of the previous kinds of terms. Such Skolem terms involving functions represent external (nondeterministic) service calls and are interpreted as the returned value chosen by an external user/environment when executing the action.

The *process* ϱ is a finite set of *condition-action rules*, of the form $Q \mapsto \alpha$, where α is an action in A and Q is a FO query over $\mathcal{R}$ whose free variables are exactly the parameters of α, and whose other terms can be quantified variables or constants mentioned in I_0.

Execution semantics. The execution semantics of a DCDS $\mathcal{S}$ is a possibly infinite transition system $\Upsilon_{\mathcal{S}}$ whose states are labeled by database instances. It represents all possible computations that the process layer can do on the data layer. Specifically, $\Upsilon_{\mathcal{S}} = \langle \mathcal{C}, \mathcal{R}, \Sigma, s_0, db, \Rightarrow \rangle$, where: *(i)* Σ is a set of states; *(ii)* $s_0 \in \Sigma$ is the initial state; *(iii)* db is a function that, given a state $s \in \Sigma$, returns the database of s, which is made up of values in $\mathcal{C}$ and conforms to $\mathcal{R}$; *(iv)* $\Rightarrow \subseteq \Sigma \times \mathfrak{A} \times \Sigma$ is a transition relation over states, labeled by action names.

Given a DCDS $\mathcal{S} = \langle \mathcal{D}, \mathcal{P} \rangle$ with $\mathcal{D} = \langle \mathcal{C}, \mathcal{R}, \mathcal{E}, I_0 \rangle$ and $\mathcal{P} = \langle \mathcal{F}, A, \varrho \rangle$, the transition system $\Upsilon_{\mathcal{S}}$ is intuitively constructed as follows. Starting from I_0, all condition-action rules in ϱ are evaluated, determining which actions are executable, and with which ground parameter assignments. Nondeterministically, one such action with parameter assignment $\alpha\rho$ is selected and executed over I_0. To do so, every effect of α (partially grounded with the parameter assignment ρ)

is evaluated, by calculating all the answers of its left-hand side, and grounding the right-hand side accordingly. If the right-hand side contains service calls, they are issued, receiving back for each of them a value nondeterministically chosen from $\mathcal{C}$. This value is then used to substitute the service call with the actual result. The overall set of ground facts obtained in this way finally constitutes the next database instance. Notice that upon the execution of $\alpha\rho$, the content of a relation is lost unless it is explicitly maintained through dedicated effects of α. The transition system construction then proceeds by constructing all possible successors, each of which is obtained by selecting one of the executable actions with parameters, and one result for each of the involved service calls. The construction then recursively proceeds over this newly generated states. For a formal description of the execution semantics, see (Bagheri Hariri et al. 2013b).

State-Boundedness. Let $\mathcal{S} = \langle \mathcal{D}, \mathcal{P} \rangle$ be a DCDS with $\mathcal{D} = \langle \mathcal{C}, \mathcal{R}, \mathcal{E}, I_0 \rangle$ and $\mathcal{P} = \langle \mathcal{F}, A, \varrho \rangle$. We say that $\mathcal{S}$ is *state-bounded* if every state of $\Upsilon_{\mathcal{S}}$ is labeled with a database instance whose size does not exceed a pre-defined bound. We say that $\mathcal{S}$ is *structurally state-bounded* if, for every database instance I_0' conforming to $\mathcal{R}$ and $\mathcal{E}$, and made up of values in $\mathcal{C}$, $\langle \mathcal{D}', \mathcal{P} \rangle$ is state-bounded, where $\mathcal{D}' = \langle \mathcal{C}, \mathcal{R}, \mathcal{E}, I_0' \rangle$.

3 Relating DCDSs and Reset Transfer Nets

In this section we introduce variants of Petri nets and draw a formal correspondence with some specific classes of DCDSs. In Section 4 we leverage on this connection to provide key results related to (structural) state-boundedness of DCDSs. We assume the reader has knowledge of standard Petri nets.[1]

3.1 Reset Transfer Nets

Reset Transfer nets (RT nets) are a restricted form of Reset Post G-nets, introduced for the first time in (Dufourd, Finkel, and Schnoebelen 1998). Intuitively, a (standard) Petri net can consume and produce a fixed amount of tokens when a transition fires, while all the remaining tokens are left untouched. The "reset" dimension of RT nets extends this behavior by allowing to instantaneously emptying the content of some places when a transition fires. The "transfer" dimension of RT nets supports instead the possibility of instantaneously transferring the entire content of some place to another place when a transition fires. These specific classes exactly correspond to those defined in (Dufourd, Finkel, and Schnoebelen 1998). An RT net combines the capabilities of R and T nets.

A (marked) *RT net* is a tuple $N = \langle P, T, F, m_0 \rangle$, where:

- P is a set of *places*;
- T is a set of *transitions* (with $P \cap T = \emptyset$);
- $F : P \times T \cup T \times P \to P \cup \{0, 1\}$ is a *flow function* mapping each pair place/transition (called *pre-arc*) and transition/place (called *post-arc*) to either a place name or the integers 0 or 1,[2] with the following constraints:
  - for each place $p \in P$ and transition $t \in T$, $F(p, t) \in \{0, 1, p\}$;
  - for each transition $t \in T$ and place $p_2 \in P$, if $F(t, p_2) = p$, then the following two conditions hold: *(i)* $F(p, t) = p$, *(ii)* for each place $p_3 \in P$ with $p_3 \neq p_2$, $F(t, p_3) \neq p$.
- m_0 is the *initial marking*, i.e., a multiset of places $m_0 \in \mathbb{B}(P)$, with $\mathbb{B}(P) : P \to \mathbb{N}$ the set of multisets over the set P of places.

Given a place $p \in P$ and a transition $t \in T$, we say that p is an *input place* for t if $F(p, t) = 1$, and that (p, t) is a *reset arc* if $F(p, t) = p$. Symmetrically, given a place $p_2 \in P$ and a transition $t \in T$, we say that p_2 is an *output place* for t if $F(t, p_2) = 1$, and that (t, p_2) is a *copy arc* if $F(t, p_2) = p$ for some $p \in P$. In this latter case, the pre-arc (p, t) and the post-arc (t, p_2) form together a *transfer arc*[3].

An *RT net* $N = \langle P, T, F, m_0 \rangle$ is a:

- *Reset Petri net (R net)* if for each $t \in T$ and $p \in P$, $F(t, p) \in \{0, 1\}$;
- *Transfer Petri net (T net)* if for each $p \in P$ and $t \in T$ such that $F(p, t) = p$, there exists $p_2 \in P$ such that $F(t, p_2) = p$;
- *Petr net* if for each $p \in P$ and $t \in T$, $F(p, t) \in \{0, 1\}$ and $F(t, p) \in \{0, 1\}$.

Execution Semantics. Given a marking m and a place p, notation $m(p) \in \mathbb{N}$ represents the number of tokens assigned by m to p, i.e., the number of times p appears in m. Instrumental to the definition of the net execution semantics is the notion of "token transfer" induced by an arc of the net for a given marking. This is formally captured by the function TRANSF, defined as follows: given an RT net $N = \langle P, T, F, m_0 \rangle$, for every marking m over P, place $p \in P$, and transition $t \in T$, we have

$$\text{TRANSF}_m(p, t) = \begin{cases} m(p) & \text{if } F(p, t) = p \\ F(p, t) & \text{otherwise} \end{cases}$$

$$\text{TRANSF}_m(t, p) = \begin{cases} m(p') & \text{if } F(t, p) = p' \\ F(t, p) & \text{otherwise} \end{cases}$$

Given a marking m and a transition $t \in T$, we say that t is *enabled* (or *fireable*) in m, written $m[t\rangle$, if, for every place $p \in P$, we have that $m(p) \geq \text{TRANSF}_m(p, t)$. Given a transition t and two markings m and m', we say that t fires in m producing m', written $m[t\rangle m'$, if and only if: *(i)* t is enabled in m, *(ii)* m' is such that for every $p \in P$,

$$m'(p) = m(p) - \text{TRANSF}_m(p, t) + \text{TRANSF}_m(t, p)$$

The execution semantics of an RT net $N = \langle P, T, F, m_0 \rangle$ is defined in terms of a transition system $\text{RG}(N) = \langle M, m_0, \to \rangle$ (called *reachability graph*), where $M \subseteq \mathbb{B}(P)$ is a set of markings, and $\to \subseteq M \times T \times M$ is a transition relation labeled with elements from T. Specifically, M and $\to$ are defined by mutual induction as the smallest sets satisfying the following properties: *(i)* $m_0 \in M$; *(ii)* if $m \in M$, then for every transition $t \in T$ and marking m' such that $m[t\rangle m'$, we have $m' \in M$ and $m \xrightarrow{t} m'$. An RT net N is *bounded* if and only if each marking in $\text{RG}(N)$ is bounded or, equivalently, $\text{RG}(N)$ contains a finite number of markings.

[1] Here Petri net is used to identify a Place Transition (P/T) net.

[2] Admitting arbitrary integers does not increase the expressive power of the model.

[3] Remember that, by definition, in this case we have $F(p, t) = p$.

An RT net $N = \langle P, T, F, m_0 \rangle$ is *structurally bounded* if its boundedness does not depend on the initial marking, i.e., for every $m'_0 \in \mathbb{B}(P)$, $\langle P, T, F, m'_0 \rangle$ is bounded.

3.2 From RT Nets to DCDSs

We formally define a translation from RT Nets to DCDSs, and then provide an informal account for it. Specifically, we introduce a translation function N-TO-D that, given an RT net $N = \langle P, T, F, m_0 \rangle$, produces a DCDS $\text{N-TO-D}(N) = \langle \mathcal{D}_N, \mathcal{P}_N \rangle$ whose execution semantics weakly reproduces that of $\text{RG}(N)$. The data layer is $\mathcal{D}_N = \langle \mathcal{C}, \mathcal{R}_N, \emptyset, I_0^N \rangle$, where

- $P_i/1 \in \mathcal{R}_N$ if and only if $p_i \in P$;
- I_0^N is such that $\{P_i(d_1), \ldots, P_i(d_k) \mid d_j \neq d_h \text{ for } j, h \in \{1, \ldots, k\} \text{ and } j \neq h\} \subseteq I_0^N$ if and only if $p^k \in m_0$.

The process layer of $\text{N-TO-D}(N)$ is $\mathcal{P}_N = \langle \mathcal{F}_N, A_N, \varrho_N \rangle$, where every transition $t \in T$ is translated into a condition-action rule $\xi_t \in \varrho_N$ and a dedicated action $\alpha_t \in A_N$, which respectively encode the enablement and firing semantics of t:

$$\xi_t = \bigwedge_{p_i^{in} \in \{p \mid F(p,t)=1\}} P_i^{in}(x_i) \mapsto \alpha_t(x_1, \ldots, x_{|\{p \mid F(p,t)=1\}|})$$

where $\alpha_t(x_1, \ldots, x_n)$ contains the following effects:

1. *(Lossy maintenance of tokens in input places)* For each $p_i^{in} \in P$ such that $F(p_i^{in}, t) = 1$:
$$P_i^{in}(y) \wedge y \neq x_i \rightsquigarrow P_i^{in}(f_i(y)) \in \text{EFFECT}(\alpha_t) \quad (1)$$
2. *(Lossy maintenance of tokens in other places)* For each $p_l \in P$ such that $F(p_l, t) = 0$:
$$P_l(y) \rightsquigarrow P_l(h_l(y)) \in \text{EFFECT}(\alpha_t) \quad (2)$$
3. *(Lossy token generation)* For each $p_j^{out} \in P$ such that $F(t, p_j^{out}) = 1$:
$$\mathsf{true} \rightsquigarrow \{P_j^{out}(g_j())\} \in \text{EFFECT}(\alpha_t) \quad (3)$$
4. *(Lossy token transfer)* For each $p_k^{tout} \in P$ such that $F(t, p_k^{out}) = p_k^{tin}$:
$$P_k^{tin}(y) \rightsquigarrow \{P_k^{tout}(h_k(y))\} \in \text{EFFECT}(\alpha_t) \quad (4)$$
5. No other effect belongs to $\text{EFFECT}(\alpha_t)$.

The intuition behind the translation is to mimic the firing of a transition by means of service calls. In the resulting DCDS, places become unary relations, and tokens become distinct identifiers from $\mathcal{C}$. The condition-action rule ξ_t checks whether each input place for t has a non-empty extension (i.e., contains at least one token) and, if so, non-deterministically selects one token from each of such places. Effects of type (1) and (2) maintain those tokens that are not affected by the firing, i.e., tokens present in non-input places, or tokens contained in input places but not selected by ξ_t. The maintenance is "lossy" because there is no guarantee that the service calls f_i will return different identifiers when called with different values. This notion of *lossiness* resembles that proposed by (Mayr 2003) for lossy counter machines and lossy Petri nets. Effects of type (3) are used to generate the tokens that t puts in its output places upon firing. Such generation is also lossy, because there is no guarantee that service calls g_j will return fresh identifiers, distinct from those already present in the output places. Effects of type (4) lossily deal with the transfer arcs of t, producing one token to be put in the transfer destination for each token present in the transfer source. Note that those places connected to a transition with a reset arc that does not form a transfer arc, do not lead to any effect in the translation. This reflects the "destructive" nature of DCDSs (data not explicitly copied are lost during the application of an action).

The following theorem states that, despite lossiness, the obtained DCDS preserves boundedness of the original net.

Theorem 1. *For every RT net N, N is bounded if and only if $\text{N-TO-D}(N)$ is state-bounded.*

This is proven in two steps. First we show that $\text{N-TO-D}(N)$ obeys to a sort of *monotonicity*. Given two database instances I_1 and I_2 over $\mathcal{R}$, we say that I_1 is *cardinality-contained* in I_2, written $I_1 \trianglelefteq I_2$ if for every relation $R \in \mathcal{R}$, $|R|_{I_1} \leq |R|_{I_2}$. Monotonicity is then defined as follows: for every two states s_1 and s_2 in Σ such that $db(s_1) \trianglelefteq db(s_2)$, and for each state s'_1 such that $s_1 \overset{\alpha_t}{\Longrightarrow} s'_1$, there exists s'_2 such that $s_2 \overset{\alpha_t}{\Longrightarrow} s'_2$ and $db(s'_1) \trianglelefteq db(s'_2)$. Monotonicity implies that, as long as state boundedness is concerned, for every state s in $\Upsilon_{\text{N-TO-D}(N)}$ it is sufficient to consider only its "maximal successors" $\text{MAX-SUCC}_{\alpha_t}(s) = \{s' \in \Sigma \mid s \overset{\alpha_t}{\Longrightarrow} s' \text{ and } \nexists s''.s \overset{\alpha_t}{\Longrightarrow} s'' \text{ and } s' \trianglelefteq s''\}$.

The second step consists in showing that the portion of $\Upsilon_{\text{N-TO-D}(N)}$ that just considers maximal successors faithfully reproduces $\text{RG}(N)$. To compare the states of $\Upsilon_{\text{N-TO-D}(N)}$ with those of $\text{RG}(N)$, we define the following *cardinality-equivalence* relation: given a marking $m \in M$ and a state $s \in \Sigma$, we say that m is cardinality-equivalent to s, written $m \approx s$, if, for each place $p_i \in P$, $m(p) = n$ if and only if $|P_i|_{db(s)} = n$. By definition, $m_0 \approx s_0$. It can then be shown, inductively, that, given $m \in M$ and $s \in \Sigma$ such that $m \approx s$, for every transition $t \in T$ (and corresponding $\alpha_t \in A_N$):

- $m[t\rangle$ if and only if α_t is executable in s;
- whenever $m[t\rangle$, given $m' \in M$ such that $m \xrightarrow{t} m'$, we have $m' \approx s'$ for every $s' \in \text{MAX-SUCC}_s(\alpha_t)$.

This implies the result, since $\text{RG}(N)$ has an unbounded run if and only if $\Upsilon_{\text{N-TO-D}(N)}$ has a state-unbounded run.

As a direct consequence, we obtain:

Corollary 2. *For every RT net N, N is structurally bounded if and only if $\text{N-TO-D}(N)$ is structurally state-bounded.*

3.3 Lossy Reset Transfer DCDSs

We now introduce a class of DCDSs that resembles those obtained from RT nets through the application of the N-TO-D translation function. For this reason, we call such a class *Lossy Reset Transfer (LRT)* DCDSs.

An *LRT DCDS* is a DCDS $\mathcal{S} = \langle \mathcal{D}, \mathcal{P} \rangle$, with $\mathcal{D} = \langle \mathcal{C}, \mathcal{R}, \mathcal{E}, I_0 \rangle$ and $\mathcal{P} = \langle \mathcal{F}, A, \varrho \rangle$, that obeys to the following requirements:

- $\mathcal{R} = \{P_1/1, \ldots, P_{|\mathcal{R}|}/1\}$ (only unary relations are used).
- $\mathcal{E} = \emptyset$.
- For each action $\alpha(\vec{x}) \in A$, ϱ contains a single rule of the form $Q(\vec{x}) \mapsto \alpha(\vec{x})$, where Q is a (non-self) cartesian

product of a set of unary relations:

$$Q(x_1, \ldots, x_n) = \bigwedge_{i \in \{1,\ldots,n\},\ P_i \neq P_j \text{ for } i \neq j} P_i(x_i)$$

The set of relations in Q is denoted by $\text{RELS}(Q)$.

- Every action $\alpha(\vec{x}) \in A$, with $Q(\vec{x}) \mapsto \alpha(\vec{x}) \in \varrho$, has the following shape:
  1. for each $P_i \in \text{RELS}(Q)$, α *must* contain a *lossy maintain-rest* effect of the form
  $$P_i(y) \wedge y \neq x_i \rightsquigarrow P_i(f_i(y)) \quad (5)$$
  2. for each $P_l \in \mathcal{R} \setminus \text{RELS}(Q)$, α *may* contain a *lossy maintain* effect of the form
  $$P_l(y) \rightsquigarrow P_l(h_l(y)) \quad (6)$$
  3. for each $P_j \in \mathcal{R} \setminus \text{RELS}(Q)$, α *may* contain *either* a *lossy create* effect of the form
  $$\mathsf{true} \rightsquigarrow P_j(g_j()) \quad (7)$$
  or a *lossy copy* effect of the form
  $$P'_j(y) \rightsquigarrow P_j(h_j(y)) \quad (8)$$
  for some $P'_j \in \mathcal{R} \setminus (\text{RELS}(Q) \cup P_j)$;
  4. no other effect belongs to α.

An LRT DCDS $\mathcal{S} = \langle \mathcal{D}, \mathcal{P} \rangle$ with $\mathcal{D} = \langle \mathcal{C}, \mathcal{R}, \emptyset, I_0 \rangle$ and $\mathcal{P} = \langle \mathcal{F}, A, \varrho \rangle$ is:

- A *Lossy Reset (LR) DCDS* if no action $\alpha \in A$ contains lossy-copy effects of the form (8); the "reset" nature comes from the fact that those relations that are not mentioned in the left-hand side of any effect in $\text{EFFECT}(\alpha)$ loose their current content when α is executed.
- A *Lossy Transfer (LT) DCDS* if, for every action $\alpha \in A$ with $Q(\vec{x}) \mapsto \alpha(\vec{x}) \in \varrho$, and every relation $P_i \in \mathcal{R} \setminus \text{RELS}(Q)$, P_i appears in the left-hand side of at least an effect of α (which must be either a lossy maintain effect of the form (6), or a lossy copy effect of the form (8)); in other words, every relation contributes to determine the content of the state obtained by the application of α.
- A *Lossy Petri (LP) DCDS* if, for every action $\alpha \in A$ with $Q(\vec{x}) \mapsto \alpha(\vec{x}) \in \varrho$, and every relation $P_i \in \mathcal{R} \setminus \text{RELS}(Q)$, P_i appears only in the left-hand side of a lossy maintain effect of the form (6) in α (and possibly in the right-hand side of a lossy create effect).

The following proposition highlights the connection induced by the N-TO-D translation function between the different classes of nets and DCDSs studied in this paper.

Proposition 3. *Given an RT/R/T/Petri net N,* $\text{N-TO-D}(N)$ *produces an LRT/LR/LT/LP DCDS.*

The proof of this proposition can be obtained by comparing the definitions for such different classes, and checking that N-TO-D preserves their properties in the translation.

We conclude this section by pointing out that LRT DCDSs are very weak in terms of their ability of querying and manipulating data. In fact, they only contain unary relations, which are queried without using any joins in the condition-action rules, and only by means of atomic queries in the effects. Furthermore, no negative components are used in such queries, except for a very limited use of inequalities (always employed to select all tuples of a relation except the one matching with one of the action parameters). Also, the content of relations is not guaranteed to be preserved when executing an action: copy and maintain effects always employ service calls to transfer data from one state to the other, which implies that the extension of a relation may change in content, and decrease in size. Finally, the overall size of the entire database instance may maximally increase only of a fixed amount of tuples, determined by the number of lossy create effects present in the applied action.

3.4 From LRT DCDSs to RT Nets

We define a translation function D-TO-N that, given an LRT DCDS $\mathcal{S} = \langle \mathcal{D}, \mathcal{P} \rangle$ with $\mathcal{D} = \langle \mathcal{C}, \mathcal{R}, \emptyset, I_0 \rangle$ and $\mathcal{P} = \langle \mathcal{F}, A, \varrho \rangle$, produces an RT net $\text{D-TO-N}(\mathcal{S}) = \langle P_\mathcal{S}, T_\mathcal{S}, F_\mathcal{S}, m_0^\mathcal{S} \rangle$ defined as follows:

- $p_i \in P_\mathcal{S}$ if and only if $P_i/1 \in \mathcal{R}$;
- $p_i^k \in m_0^\mathcal{S}$ if and only if $|P_i|_{I_0} = k$;
- $t_\alpha \in T$ if and only if $\alpha \in A$;
- As for pre-arcs, for each $p_i \in P_\mathcal{S}$ and each $t_\alpha \in T$:
  1. if, given $Q(\vec{x}) \mapsto \alpha(\vec{x})$ in ϱ, $P_i \in \text{RELS}(Q)$, then $F_\mathcal{S}(p_i, t_\alpha) = 1$;
  2. else if α contains an effect of the form $P_i(y) \rightsquigarrow P_i(h_i(y))$,[4] then $F_\mathcal{S}(p_i, t_\alpha) = 0$;
  3. else $F_\mathcal{S}(p_i, t_\alpha) = p_i$.
- As for post-arcs, for each $t_\alpha \in T$ and $p_i \in P_\mathcal{S}$:
  1. if α contains an effect of the form $\mathsf{true} \rightsquigarrow P_i(g_i())$, then $F(t_\alpha, p_i) = 1$;
  2. if α contains an effect of the form $P'_i(y) \rightsquigarrow P_i(h_i(y))$ with $P'_i \neq P_i$, then $F(t_\alpha, p_i) = p'_i$;
  3. $F(t_\alpha, p_i) = 0$ otherwise.

Intuitively, the translation works as follows. Every relation mentioned in the condition-action rule for α becomes an input place for t_α, and every relation targeted by a lossy create effect becomes an output place for t_α. Relations which are lossily maintained by α do not contribute as input for t_α, attesting that t will not consume tokens from them. Relations that do not fall in any of these two categories are consumed completely by t; this can happen either because their content is lossily copied, or because they do not appear in the left-hand side of any effect in α (and hence will loose their content when α is applied). Finally, lossy copy effects are translated into a transfer arc passing through t (notice that, in this case, the definition of LRT DCDSs does not allow the target relation to be also targeted by a lossy create effect).

Theorem 4. *The translation functions* N-TO-D *and* D-TO-N *are one the inverse of the other, i.e., (i) for every RT net N, $N = \text{D-TO-N}(\text{N-TO-D}(N))$; (ii) for every LRT DCDS $\mathcal{S}$, $\mathcal{S} = \text{N-TO-D}(\text{D-TO-N}(\mathcal{S}))$.*

The proof of this theorem is obtained by: *(i)* considering each component of an RT net, and showing that the same component is obtained by applying D-TO-N $\circ$ N-TO-D on it, *(ii)* considering each component of an LRT DCDS, and showing that the same component is obtained by applying N-TO-D $\circ$ D-TO-N on it.

Theorems 4 and 1, together with Proposition 3, imply:

[4] Note that the same relation P_i is used in both sides of the effect.

Corollary 5. *Given an LRT/LR/LT/LP DCDS $\mathcal{S}$,* D-TO-N($\mathcal{S}$) *produces an RT/R/T/Petri net.*

Theorem 6. *For every LRT DCDS $\mathcal{S}$, $\mathcal{S}$ is state-bounded if and only if* D-TO-N$\mathcal{S}$ *is bounded.*

As a direct consequence of Theorem 6, we then have:

Corollary 7. *For every LRT DCDS $\mathcal{S}$, $\mathcal{S}$ is structurally state-bounded if and only if* D-TO-N($\mathcal{S}$) *is structurally bounded.*

4 State-Boundedness is Difficult

We now exploit the connection between RT nets (and their subclasses) and LRT DCDSs (and their subclasses) to show that state-boundedness is intrinsically difficult to check for data-aware dynamic systems: either highly-undecidable[5] or intractable even for the low expressive systems expressible with LRT DCDSs. In particular, we have that: *(i)* checking boundedness of R nets reduces to checking state-boundedness of LR DCDSs (cf. Proposition 3 and Theorem 1); *(ii)* checking state-boundedness of LP DCDSs reduces to checking boundedness of Petri nets (cf. Corollary 5 and Theorem 6); *(iii)* checking state-boundedness of LT DCDSs reduces to checking boundedness of T nets (cf. Corollary 5 and Theorem 6). Similar reductions hold for structural state-boundedness, thanks to Proposition 3 and Corollaries 2, 5, and 7.

Since checking boundedness for R nets is Σ^1_1-complete (Dufourd, Finkel, and Schnoebelen 1998; Dufourd, Jancar, and Schnoebelen 1999), we thus get:

Theorem 8. *Checking state-boundedness is highly undecidable for LR (and hence LRT) DCDSs.*

Thanks to the fact that checking boundedness is EXPSPACE-complete for Petri nets (Esparza and Nielsen 1994), and decidable for T nets (Dufourd, Jancar, and Schnoebelen 1999), decidability is interestingly obtained for those LRT DCDSs in which all relations are used in the left-hand side of effects, i.e., all relations contribute to determine the new database instance obtained after the application of an action. The sub-class of LRT DCDSs that enjoy this property is that of LT (which includes LP) DCDSs:

Theorem 9. *Checking state-boundedness is* EXPSPACE*-complete for LP DCDSs.*

Theorem 10. *Checking state-boundedness is decidable and* EXPSPACE*-hard for LT DCDSs.*

Structural state-boundedness pushes the undecidability frontier even further, but can be efficiently checked for the very simple case of LP DCDSs. Specifically, since checking structural boundedness is Π^1_2-complete for R nets, Π^1_1-complete for T nets (Dufourd, Jancar, and Schnoebelen 1999), and decidable in PTIME for Petri nets, (Esparza and Nielsen 1994; Dufourd, Jancar, and Schnoebelen 1999) we get:

Theorem 11. *Checking structural state-boundedness is highly undecidable for LT and LR (and hence LRT) DCDSs.*

Theorem 12. *Checking structural state-boundedness is decidable in* PTIME *for LP DCDSs.*

[5]I.e., at the first level or above in the Analytical Hierarchy (Rogers 1967).

We conclude by observing that a line of reasoning analogus to that of DCDSs can be repeated for similar frameworks, e.g., ACMASs (Belardinelli, Lomuscio, and Patrizi 2012).

5 Sufficient State-Boundedness Conditions

In Section 3.1, we have proved that checking (structural) state-boundedness even for very limited DCDSs is already undecidable. We introduce now sufficient syntactic conditions that guarantee structural state-boundedness. In particular, we discuss different dimensions for relaxation of the GR^+-acyclicity condition originally introduced by Bagheri Hariri et al. (2013b), and based on them we develop our most general sufficient condition, called stratified-$\exists GR^{++}$-acyclicity.

5.1 GR^{++}-Acyclicity

The GR^+-acyclicity condition for structural state-boundedness (Bagheri Hariri et al. 2013b) is a syntactic restriction over the dataflow graph of a DCDS, in which nodes of the dataflow graph represent relation names that occur in the schema of the system, and edges represent the flow of the data between these relations. Here we introduce GR^{++}-acyclicity as a relaxation of GR^+-acyclicity in which the dataflow graph represents the system behavior at finer granularity by using as nodes positions of relations, instead of relations themselves.

We call *refined dataflow graph* of a set A of actions the directed edge-labeled graph $\langle N, G\rangle$ where: *(i)* $N \subseteq \mathcal{R} \times \mathbb{N}^+$ is a set of nodes such that for every R/n occurring in A, $\langle R, i\rangle \in N$ for every $i \in \{1, \ldots, n\}$; *(ii)* G is a 5-tuple $(R_1, \mathsf{id}, R_2, b, a)$, where R_1 and R_2 are two nodes in N, id is a (unique) edge identifier, b is a Boolean flag used to mark *special* edges, and a is flag used to denote the action of A to which the edge belongs. Formally, G is the minimal set such that, for each action α in A, each effect e in α of the form $q^+ \wedge Q^- \rightsquigarrow E$, each $R(r_1, \ldots, r_m)$ in q^+, each $i \in \{1, \ldots, m\}$, each $Q(t_1, \ldots, t_n)$ in E, and each $j \in \{1, \ldots, n\}$, we have that:

- if t_i is a free variable and $r_i = t_j$, then $(\langle R, i\rangle, \mathsf{id}, \langle Q, j\rangle, \mathsf{false}, \alpha) \in G$.
- if t_i is a service call $f(s_1, \ldots, s_l)$, then for each $k \in \{1, \ldots, l\}$, if $r_i = s_k$ then $(\langle R, i\rangle, \mathsf{id}, \langle Q, j\rangle, \mathsf{true}) \in G$.
- if t_i is a nullary service call $f()$, then $(\langle R, i\rangle, \mathsf{id}, \langle Q, j\rangle, \mathsf{true}) \in G$.

in which id is a fresh edge identifier.

Let π_1, π_2, and π_3 be paths in the dataflow graph D of A. We say that $\langle \pi_1, \pi_2, \pi_3\rangle$ is a *generate-recall triple* if

(i) $\pi_1\pi_2\pi_3$ forms a path in D,
(ii) π_1, π_3 are simple cycles, and
(iii) π_2 is a path containing a special edge, with $edges(\pi_2) \setminus edges(\pi_1) \neq \emptyset$.

The generate-recall triple $\langle \pi_1, \pi_2, \pi_3\rangle$ is called *non-accumulating* if there is an edge $e \in edges(\pi_2)$ that does not belong to the same action as any of the edges following e in $\pi_2\pi_3$. Finally, we say that A is GR^{++}*-acyclic* if all its generate-recall triples are non-accumulating.

We say a process layer $\mathcal{P} = \langle \mathcal{F}, A, \varrho\rangle$ is GR^{++}*-acyclic*, if A is GR^{++}-acyclic. A DCDS is GR^{++}-acyclic if its process layer is GR^{++}-acyclic. The GR^+-acyclicity definition by

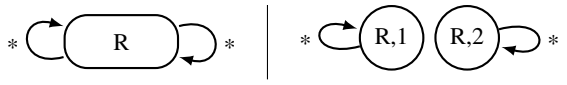


Figure 1: Dataflow and refined dataflow graphs of Example 1

Bagheri Hariri et al. (2013b) differs only in that the dataflow graph's nodes represent entire relations, not their positions (the dataflow graph by Bagheri Hariri et al. corresponds to collapsing all refined dataflow graph nodes modeling positions of relation R into a single node, for every R.)

Example 1. Consider a DCDS D with one action $\alpha = \{R(x,y) \rightsquigarrow \{R(f(x),\ h(y))\}$. It is easy to see that D is *(i)* structural state-bounded, *(ii)* not GR^{+}-acyclic, but *(iii)* GR^{++}-acyclic. For *(i)*, observe that the cardinality of R after executing α is at most the cardinality before execution (potentially strictly lower if the service calls behave non-injectively). For *(ii)*, consult the data flow graph on the left side of Figure 1. The simple cycle π_1 can be the left self-loop on node R, π_2 and π_3 can coincide with the right self-loop. For *(iii)*, consult the refined data flow graph on the right side of Figure 1. Notice now that this graph contains no generate-recall cycle, since in each connected component π_1 and π_2 must coincide. ■

Theorem 13. *GR^{++}-acyclicity is a strict relaxation of GR^{+}-acyclicity that guarantees structural state-boundedness.*

5.2 Safe-GR^{++}-Acyclicity

The main idea of this relaxation is to distinguish different types of special edges in the dataflow graph to define a more refined condition for structural state-boundedness. We borrowed the notion of affected positions, which are an overestimation of the positions in which a new value can be introduced, from Calì, Gottlob, and Kifer (2013), Meier, Schmidt, and Lausen (2009), to distinguish unbounded edges, which is an overestimation of the edges of the dataflow graph, in which unboundedly many different values can flow during the life of a DCDS.

More formally, we call *affected positions* of a set A of actions, denoted with $\text{AFF}(A)$, the smallest set of positions π occurring in A satisfying following conditions:

- If a service call appears in π, then $\pi \in \text{AFF}(A)$.
- If in the head of an effect a variable x appears in π, and occurrences of x in the body of this effect is only limited to the affected positions.

We use affected positions to determine an overestimation of unbounded edges of dataflow graph. Given an effect f, we say that an edge e from a position p in the body of e to a position in the head of e is (potentially) *unbounded* if all the positions in the body of e that using the same variable as p are affected positions. Otherwise, we call e *bounded*. For instance, consider a DCDS with one action as follows: $\{A(x) \wedge B(x) \wedge C(y) \rightsquigarrow A(f(x)), C(h(y))\}$. $\langle A,1\rangle$ and $\langle C,1\rangle$ are the only affected positions. The edges from the position $\langle A,1\rangle$ and $\langle B,1\rangle$ to $\langle A,1\rangle$ are bounded edges, since for each of them there is an unaffected position $\langle B,1\rangle$ in the body of the effect, while the special edge from $\langle C,1\rangle$ to $\langle C,1\rangle$ is an unbounded edge.

We call *safe-refined dataflow graph* of a set A of actions the directed edge-labeled graph $\langle N, G\rangle$ where: *(i)* $N \subseteq \mathcal{R} \times \mathbb{N}^{+}$ is a set of nodes such that for every R/n occurring in A, $\langle R,i\rangle \in N$ for every $i \in \{1,\ldots,n\}$; *(ii)* G is a 6-tuple $(R_1, \mathsf{id}, R_2, f_s, f_b, a)$, where R_1 and R_2 are two nodes in N, id is a (unique) edge identifier, f_s is a flag that specify if the edge is normal with "$-$", or special with "$*$", f_b is a flag that specify if an edge is bounded with "b", or unbounded with "u", and a is flag used to denote the action of A to which the edge belongs. Formally, G is the minimal set satisfying the following condition. For each action α in A, each effect e in α of the form $q^{+} \wedge Q^{-} \rightsquigarrow E$, each $R(r_1,\ldots,r_m)$ in q^{+}, each $i \in \{1,\ldots,m\}$ each $Q(t_1,\ldots,t_n)$ in E, and each $j \in \{1,\ldots,n\}$:

- If t_i is a free variable and $r_i = t_j$,
  - if for each position p in the body of e in which the variable r_i occurs, it holds that $p \in \text{AFF}(A)$, then $(\langle R,i\rangle, \mathsf{id}, \langle Q,j\rangle, \text{-}, u, a) \in G$;
  - otherwise, $(\langle R,i\rangle, \mathsf{id}, \langle Q,j\rangle, \text{-}, b, a) \in G$.
- If t_i is a service call $f(s_1,\ldots,s_l)$ with $l \geq 1$, then for each $k \in \{1,\ldots,l\}$, if $r_i = s_k$, then
  - If for each position p in the body of e in which the variable r_i happens it holds that $p \in \text{AFF}(A)$, then $(\langle R,i\rangle, \mathsf{id}, \langle Q,j\rangle, *, u, a) \in G$;
  - otherwise, $(\langle R,i\rangle, \mathsf{id}, \langle Q,j\rangle, *, b, a) \in G$.
- If t_i is a nullary service call $f()$, then
  - if for each position p in the body of e it holds that $p \in \text{AFF}(A)$ then $(\langle R,i\rangle, \mathsf{id}, \langle Q,j\rangle, *, u) \in G$;
  - otherwise, $(\langle R,i\rangle, \mathsf{id}, \langle Q,j\rangle, *, b) \in G$.

  in which id is a fresh edge identifier.

We define the notion of *generate-recall triple* for a safe-refined dataflow graph exactly as in the case of GR^{++}-acyclicity. Moreover, a generate-recall triple $\langle \pi_1, \pi_2, \pi_3\rangle$ is called *safe* if either there is an edge $e \in edges(\pi_2)$ that does not belong to the same action as any of the edges following e in $\pi_2\pi_3$, or π_3 contains a bounded edge. Finally, we say that A is *safe-GR^{++}-acyclic* if all its generate-recall triples are safe.

Example 2. Consider a DCDS D with one action

$$\alpha = \begin{cases} A(x) \rightsquigarrow \{A(f(x))\} \\ B(x) \rightsquigarrow \{B(x)\} \\ A(x) \rightsquigarrow \{R(x,f(x))\} \\ R(x,y) \wedge B(x) \rightsquigarrow \{R(y,x)\} \end{cases}$$

D is *(i)* structural state-bounded, *(ii)* not GR^{++}-acyclic, but *(iii)* safe-GR^{++}-acyclic.

Given a state I of a DCDS, let's denote with $|R_i|_I$ the number of different values that occurs in I in the position i of the relation R. Consider the states I and I' of D, in which I' is the result of the execution of α over I.

For *(i)* observe that $|A_1|_{I'} \leq |A_1|_I$, $|B_1|_{I'} = |B_1|_I$, $|R_2|_{I'} \leq |A_1|_I + |B_1|_I$, and $|R_1|_{I'} \leq (|A_1|_I + |R_2|_I) \leq (2.|A_1|_I + |B_1|_I)$. Given the fact that the cardinality of $\langle A,1\rangle$ and $\langle B,1\rangle$ is not increasing, and the cardinality of all the positions is linear in the cardinality of these two positions, the size of the resulting instance by application of α cannot

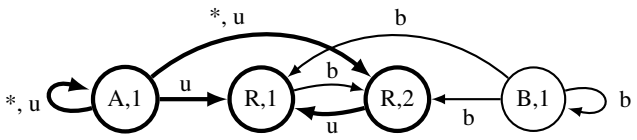


Figure 2: Safe-refined dataflow graph of Example 2. Affected positions and unbounded edges are shown with thicker lines

increase unboundedly. For *(ii)* consult the refined dataflow graph of D given in Figure 2, ignoring the labels for boundedness of edges. The path π_1 can be the self loop in $\langle A, 1\rangle$, π_2 can be the edge from $\langle A, 1\rangle$ to $\langle R, 1\rangle$, and π_3 can be the simple cycle between $\langle R, 1\rangle$ and $\langle R, 1\rangle$. For *(iii)*, consult the safe-refined dataflow graph of Figure 2. $\langle A, 1\rangle$ and $\langle R_2\rangle$ are affected positions because of occurrences of the service call $f(x)$ in them. Moreover, because of the third effect, $\langle R, 1\rangle$ is also an affected position, but $\langle B, 1\rangle$ is not an affected position. Consequently all the outgoing edges from $\langle B, 1\rangle$ are bounded edges. Moreover, because of the occurrences of $\langle B_1\rangle$ with the same variable as the $\langle R, 1\rangle$ in the body of the fourth effect, the outgoing edge from $\langle R, 1\rangle$ to $\langle R, 2\rangle$ is also a bounded edge. Notice now that this graph contains no generate-recall cycle $\pi_1\pi_2\pi_3$ such that all the edges of the recall cycle are unbounded edges. ■

Theorem 14. *Safe-GR^{++}-acyclicity is a strict relaxation of GR^{++}-acyclicity that guarantees structural state-boundedness.*

5.3 Stratified-GR^{++}-Acyclicity

The syntactic restrictions that we have studied until now are defined over the structure of the effects of the actions, and ignore the conditions under which the actions can be executed, i.e., the process of the system. Here, we exploit process to single out larger classes of structural state-bounded DCDSs. Getting insight from stratified relaxation of weak-acyclicity in data exchange (Deutsch, Nash, and Remmel 2008), we specify the potential order of execution of actions of a system, through what we call the control-flow graph of the system, and use this graph to determine maximal set of actions that can violate structural state-boundedness. Remember that actions of DCDSs are not inflationary, and the next state is only determined by applying the effects of the action over the current state. Consequently, by looking to the head of the effects of an action α, we can find a superset of the actions that can be fired after α. We use this idea to relax safe-GR^{++}-acyclicity.

Given an action α and a set of condition-action rules ρ, the set of requirements for firing of α wrt ρ is a set of set of relation names defined as follows: $\text{INPUT}(\rho, \alpha) = \{\{\text{RELATIONS}(Q^+)\} \mid (Q^+ \wedge Q \mapsto \alpha) \in \rho\}$, in which Q^+ is a conjunctive query, and $\text{RELATIONS}(Q)$ is the set of relation names occurring in Q. We denote with $\text{OUTPUT}(\alpha)$ the set of relation names occurring in the head of effects of α. Given a set of condition-action rules ϱ, and two actions α and β, we say that β can (potentially) be fired after α wrt ϱ, written $\alpha <_\varrho \beta$ (or simply $\alpha < \beta$), if there exists a set of relation names $S \in \text{INPUT}(\varrho, \beta)$ such that $S \subseteq \text{OUTPUT}(\alpha)$.

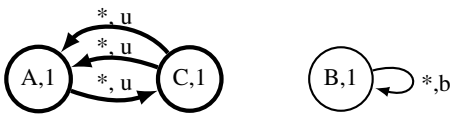


Figure 3: Safe-refined dataflow graph of Example 3. Affected positions and unbounded edges are shown with thicker lines

Given a set of condition-action rules ϱ and a set of actions A, the *controlflow graph* of A is the directed edge-labeled graph $\langle A, E\rangle$ in which for two actions $\alpha, \beta \in \Sigma$ we have that $(\alpha, \beta) \in E$ iff $\alpha < \beta$. We say a DCDS with the set of actions A is *stratified-GR^{++}-acyclic* if for every set of actions S that correspond to a maximal strongly connected component in the controlflow graph of A, the restriction of A to S is safe-GR^{++}-acyclic.

Example 3. Consider a DCDS D with two actions $\alpha(x) = \{A(x) \wedge B(y) \rightsquigarrow \{C(x), B(y)\}$ and $\beta(x, y) = \{C(x) \wedge C(y) \rightsquigarrow \{A(g(x)), A(h(y))\}$ and the set of condition-action rules $\{A(x) \wedge B(x) \mapsto \alpha(x), C(x) \wedge C(y) \mapsto \beta(x, y)\}$. The refined dataflow graph of D is shown in Figure 3. It is easy to see that D is *(i)* structural state-bounded, *(ii)* not safe-GR^{++}-acyclic, but *(iii)* stratified-GR^{++}-acyclic.

For *(i)*, observe that after execution of action α it is only possible to execute action β since the resulting state will not have any atom with the relation name A, which is needed for execution of α. Moreover, after executing β, none of the actions can be fired, since the resulting state will not have any atom with the relation name B, which is needed for execution of α, and not any atom with the relation name of C which is needed for execution of β. For *(ii)*, consult the safe-refined dataflow graph of Figure 3. As a witness of the generate-recall cycle, it is possible to select π_1 as the cycle generated by the top edge from $\langle C, 1\rangle$ and $\langle A, 1\rangle$ and the edge from $\langle A, 1\rangle$ to $\langle C, 1\rangle$, π_2 as the second edge from $\langle C, 1\rangle$ to $\langle A, 1\rangle$, and π_3 as the cycle generated by the edge from $\langle A, 1\rangle$ to $\langle C, 1\rangle$ and the second edge from $\langle C, 1\rangle$ to $\langle A, 1\rangle$.

For *(iii)*, $\text{INPUT}(\varrho, \alpha) = \{\{A, B\}\}$, $\text{INPUT}(\varrho, \beta) = \{\{C\}\}$, $\text{OUTPUT}(\alpha) = \{B, C\}$, $\text{OUTPUT}(\beta) = \{A\}$. Consequently, $\alpha \not< \alpha$, $\alpha < \beta$, $\beta \not< \alpha$, and $\beta \not< \beta$. The controlflow graph has only one edge from α to β. Since there is no strongly connected component in the controlflow graph, D trivially satisfies the condition of stratified-GR^{++}-acyclicity. ■

Theorem 15. *Stratified-GR^{++}-acyclicity is a strict relaxation of safe-GR^{++}-acyclicity that guarantees structural state-boundedness.*

5.4 $\exists$- Acyclicity Relaxations

All the syntactic restrictions presented so far are based on finding cycles in the corresponding dataflow graphs, as part of generate-recall pairs. The graph edges are constructed conservatively, to reflect possible flow of data between relations of the data layer. In this section, we observe that edge construction can be over-conservative, and we show a technique to safely consider fewer edges. This is clearly beneficial, as removing edges reduces the opportunity for forming cycles.

The technique is based on a counterintuitive observation regarding overestimations of an original DCDS D. These are rewritings O of D whose effects generate a superset of the data generated by the effects of D. It turns out that it is possible for the data flow graph of overestimate O to have a subset of the edges of the data flow graph of D. Since structural state-boundedness of O guarantees state-boundedness of D, checking the data flow graph of O leads to a relaxed syntactic restriction for state-boundedness.

One situation in which overestimation of the system results in smaller dataflow graphs is when a nullary service call occurs in heads of the DCDS. In the definition of GR^{++}-acyclicity in Section 5.1, for each nullary service call in the head of an effect we add a special edge from all the positions of the body of the effect to the position in which the service call occurs. Consider an effect $e : A(x) \wedge B(y) \rightsquigarrow C(f())$. In the definition of GR^{++}-acyclicity there are special edges from $\langle A, 1\rangle$ and $\langle B, 1\rangle$ to $\langle C, 1\rangle$ in the dataflow graph. However, since we are looking for sufficient conditions for state-boundedness, we can overestimate the result of e by replacing the effect e with $e_1 : A(x) \wedge B(y) \rightsquigarrow C(f(x))$ or $e_2 : A(x) \wedge B(y) \rightsquigarrow C(f(y))$. Notice although the resulting system is an overestimation of the original system, the resulting dataflow graph has less edges, and consequently, it is more likely to satisfy the condition of GR^{++}-acyclicity.

A similar situation holds for repeated positions with the same variables in the body of the effect. Lets assume to have an effect $f : A(x) \wedge B(x) \rightsquigarrow C(x)$. It is possible to over estimate f with $f1 : A(x) \wedge \exists y B(y) \rightsquigarrow C(x)$ or $f2 : \exists y A(y) \wedge B(x) \rightsquigarrow C(x)$. The resulting dataflow graphs will have less edges, and consequently, it is more likely to satisfy the condition of GR^{++}-acyclicity.

More formally, given a DCDS D, $\text{OVEREST}(D)$ is a set of DCDSs obtained from D by applying the following rules for each of the possible options:

- Replace the nullary service calls in the head of the effects with a unary service call using as variable, one of the variables of the body of the effect. Notice that for each choice of variables for unary service calls, there will be one DCDS in $\text{OVEREST}(D)$.
- Among multiple occurrences of one variable in different positions of the body of an effect, keep one of them, and rename and existentially quantify the others. Also notice that for each possible selection option, there will be a DCDS in $\text{OVEREST}(D)$.

We say that D is $\exists GR^{++}$-acyclic, if there exists a DCDS $D' \in \text{OVEREST}(D)$, such that D' is GR^{++}-acyclic.

Example 4. Consider a DCDS D with one actions $\alpha = \{A(x) \wedge B(x) \wedge B(y) \rightsquigarrow \{A(f()), B(g())\}$. The dataflow graph of D is depicted in Figure 4. It is easy to see that D is *(i)* state-bounded, and *(ii)* not stratified-GR^{++}-acyclic, but *(iii)* $\exists GR^{++}$-acyclic. For *(i)*, observe that $|A_1|_{I'} \leq |A_1|_I$, and $|B_1|_{I'} \leq |B_1|_I$. For *(ii)*, consult the safe-refined dataflow graph of Figure 4. As a witness of the generate-recall cycle consider π_1 as the self loop of $\langle A, 1\rangle$, π_2 as the edge from $\langle A, 1\rangle$ to $\langle B, 1\rangle$, and π_3 as the self loop of $\langle B, 1\rangle$.

For *(iii)*, considering the two options for selecting a position among the positions with the same variable in body of the only effect of α, and four options for overestimating

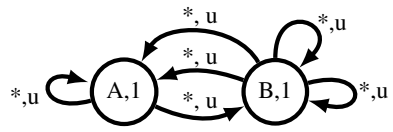


Figure 4: Safe-refined dataflow graph of Example 4

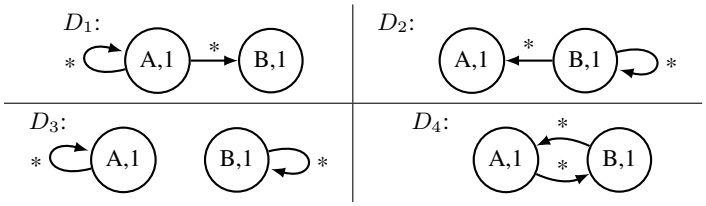


Figure 5: Refined dataflow graphs regarding $\exists GR^{++}$-acyclicity of Example 4

the two nullary service calls in the head of the effect of α to unary service calls using the variables x and y of the body, $\text{OVEREST}(D)$ contains eight different overestimations. Here we represent four of these overestimations $D_1, \ldots, D_4$, each of them with one action, namely, $\alpha_1, \ldots, \alpha_4$ as follows:

$$\alpha_1 : \{A(x) \wedge \exists z\ B(z) \wedge B(y) \rightsquigarrow \{A(f(x), B(g(x))\}\}$$
$$\alpha_2 : \{\exists z\ A(z) \wedge B(x) \wedge B(y) \rightsquigarrow \{A(f(x), B(g(x))\}\}$$
$$\alpha_3 : \{A(x) \wedge \exists z\ B(z) \wedge B(y) \rightsquigarrow \{A(f(x), B(g(y))\}\}$$
$$\alpha_4 : \{A(x) \wedge \exists z\ B(z) \wedge B(y) \rightsquigarrow \{A(f(y), B(g(x))\}\}$$

For *(iii)*, consult the refined dataflow graphs of $D_1, \ldots, D_4$ in Figure 5. For $\exists GR^{++}$-acyclicity, it is enough to have one of the overestimations GR^{++}-acyclic. In this example, all four represented overestimations are GR^{++}-acyclic. ■

Theorem 16. *$\exists GR^{++}$-acyclicity is a strict relaxation of GR^{++}-acyclicity that guarantees state-boundedness.*

A Hierarchy of Acyclicity Classes. The idea for $\exists GR^{++}$-acyclicity applies to generate existential relaxations of safe-GR^{++}-acyclicity and stratified-GR^{++}-acyclicity, called safe-$\exists GR^{++}$-acyclicity and stratified-$\exists GR^{++}$-acyclicity. Figure 6 summarizes our results. Note that stratified-$\exists GR^{++}$-acyclicity is the most relaxed condition, and GR^{+} is the most over-conservative. Moreover, we can prove that safe-GR^{++}-acyclicity and $\exists GR^{++}$-acyclicity are incomparable.

All acyclicity variants we have introduced are purely syntactic notions. The non-existential ones can be checked in coNP since the various versions of dataflow graphs have size polynomial in the size of the process layer specification, and, if there is a violation of acyclicity, then we can show that there is a polynomial-sized one. For existential relaxations, complexity jumps to Σ_2^p, due to the need of guessing the member of $\text{OVEREST}(D)$ to which the coNP check applies.

6 Conclusion

State-boundedness has recently gained a central place in verification of data-aware dynamic systems. This motivated us to study whether there are interesting decidable classes of such systems for which state-boundedness is decidable. Thanks to

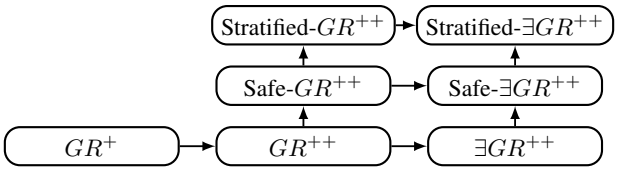


Figure 6: Syntactic restrictions for structural state-boundedness. Arrows show strict containment of classes. Unconnected classes are incomparable.

novel connections with Petri nets, we answer the question in the negative, but at the same time paving the way towards an entire field dedicated to studying sufficient, checkable conditions for state-boundedness. Inspired from sufficient conditions for chase termination in database theory, we show that there are many non-trivial classes of DCDSs that guarantee state-boundedness. The flourishing literature on fine-grained syntactic conditions for chase termination, not considered in this work, points to promising future research.

Acknowledgments. This research has been partially supported by the EU IP project Optique (Scalable End-user Access to Big Data), grant agreement n. FP7-318338.

References

Bagheri Hariri, B.; Calvanese, D.; Montali, M.; De Giacomo, G.; De Masellis, R.; and Felli, P. 2013a. Description logic Knowledge and Action Bases. *Journal of Artificial Intelligence Research* 46.

Bagheri Hariri, B.; Calvanese, D.; Montali, M.; De Giacomo, G.; and Deutsch, A. 2013b. Verification of relational data-centric dynamic systems with external services. In *Proc. of the 32th Symp. on Principles of Database Systems (PODS)*.

Belardinelli, F.; Lomuscio, A.; and Patrizi, F. 2012. An abstraction technique for the verification of artifact-centric systems. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*. AAAI Press.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *Journal of Artificial Intelligence Research* 48:115–174.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning* 39(3):385–429.

Calvanese, D.; De Giacomo, G.; and Montali, M. 2013. Foundations of data-aware process analysis: A database theory perspective. In *Proc. of the 32th Symp. on Principles of Database Systems (PODS)*, 1–12. ACM Press.

De Giacomo, G.; Lesperance, Y.; and Patrizi, F. 2012. Bounded situation calculus action theories and decidable verification. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, 467–477.

Deutsch, A.; Hull, R.; Patrizi, F.; and Vianu, V. 2009. Automatic verification of data-centric business processes. In *Proc. of the 12th Int. Conf. on Database Theory (ICDT)*, 252–267.

Deutsch, A.; Nash, A.; and Remmel, J. B. 2008. The chase revisited. In *Proc. of the 27th Symp. on Principles of Database Systems (PODS)*.

Dufourd, C.; Finkel, A.; and Schnoebelen, P. 1998. Reset nets between decidability and undecidability. In *Proc. of the 25th Coll. on Automata, Languages and Programming (ICALP)*, volume 1443, 103–115.

Dufourd, C.; Jancar, P.; and Schnoebelen, P. 1999. Boundedness of reset p/t nets. In *Proc. of the 26th Coll. on Automata, Languages and Programming (ICALP)*, volume 1644 of *Lecture Notes in Computer Science*, 301–310. Springer.

Dumas, M. 2011. On the convergence of data and process engineering. In *Proc. of ADBIS*, volume 6909 of *Lecture Notes in Computer Science*, 19–26. Springer.

Esparza, J., and Nielsen, M. 1994. Decidability issues for petri nets - a survey. *Bulletin of the EATCS* 52:244–262.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: Semantics and query answering. *Theoretical Computer Science* 336(1):89–124.

Hull, R. 2008. Artifact-centric business process models: Brief survey of research results and challenges. In *Proc. of the On the Move Confederated Int. Conf. (OTM 2008)*, volume 5332 of *Lecture Notes in Computer Science*, 1152–1163. Springer.

Mayr, R. 2003. Undecidable problems in unreliable computations. *Theoretical Computer Science* 297(1-3):337–354.

Meier, M.; Schmidt, M.; and Lausen, G. 2009. On chase termination beyond stratification. *PVLDB* 2(1).

Rogers, H. 1967. *Theory of recursive functions and effective computability*. McGraw-Hill.

Solomakhin, D.; Montali, M.; Tessaris, S.; and De Masellis, R. 2013. Verification of artifact-centric systems: Decidability and modeling issues. In *Proc. of the 11th Int. Conf. on Service Oriented Computing (ICSOC)*.

Vianu, V. 2009. Automatic verification of database-driven systems: a new frontier. In *Proc. of the 12th Int. Conf. on Database Theory (ICDT)*, 1–13.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Decidable Reasoning in a Fragment of the Epistemic Situation Calculus

Gerhard Lakemeyer*
Dept. of Computer Science
RWTH Aachen University
52056 Aachen
Germany
gerhard@cs.rwth-aachen.de

Hector J. Levesque
Dept. of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S 3A6
hector@cs.toronto.edu

Abstract

The situation calculus is a popular formalism for reasoning about actions and change. Since the language is first-order, reasoning in the situation calculus is undecidable in general. An important question then is how to weaken reasoning in a principled way to guarantee decidability. Existing approaches either drastically limit the representation of the action theory or neglect important aspects such as sensing. In this paper we propose a model of limited belief for the epistemic situation calculus, which allows very expressive knowledge bases and handles both physical and sensing actions. The model builds on an existing approach to limited belief in the static case. We show that the resulting form of limited reasoning is sound with respect to the original epistemic situation calculus and eventually complete for a large class of formulas. Moreover, reasoning is decidable.

Introduction

The situation calculus (McCarthy 1963; Reiter 2001) is a popular formalism for reasoning about action and change. Since the language is first-order, one of the advantages of the situation calculus is its high degree of expressiveness. However, this comes at a price: reasoning in the situation calculus is undecidable in general.

There are various ways of addressing this issue in a principled way. One would be to drastically limit the expressiveness, either by moving to a propositional action language such as the language $\mathcal{A}$ (Gelfond and Lifschitz 1993) or its descendants, or by integrating actions into description logics (Baader *et al.* 2005). Another would be to stay within the situation calculus but limit its representational or inferential capabilities (or both) without giving up its first-order nature. In (Gu and Soutchanski 2007), the authors use the two-variable fragment of first-order logic, which is known to be decidable, as the base logic of the situation calculus. In (De Giacomo *et al.* 2012; 2013), decidability is achieved by bounding the extension of fluent predicates by some constant. Going beyond two variables or boundedness requires further restrictions on the inference mechanism to retain decidability. The approach by Liu and Levesque (2005) restricts the representation of what holds in the current situation to literals. Such knowledge bases can then be queried efficiently using a sound but incomplete evaluation-based reasoner proposed in (Levesque 1998). When actions have only local effects, the result of an action can be represented by progressing the current knowledge base to a new set of literals. In (Claßen and Lakemeyer 2009), the restriction to literals is relaxed by allowing disjunctions in the representation of the initial situation. Queries about what holds after actions have been performed are evaluated by first regressing them to a query about the initial situation and then appealing to a logic of limited belief for static knowledge bases with disjunctions developed by Liu et al. (2004). Here a major limitation is that sensing actions cannot be dealt with.

The last two approaches have in common that they separate the reasoning task into two parts: one which addresses the dynamics of actions using progression and regression, respectively, and another which deals with querying a static knowledge base using existing techniques. A downside of this separation is that certain desirable features seem to get lost. For example, it is not at all clear how to handle sensing actions in the case of (Claßen and Lakemeyer 2009).

In order to avoid such limitations, we propose a new model of limited belief, which avoids the above separation by incorporating actions directly into the model. The starting point is the model of limited belief for the static case proposed in (Lakemeyer and Levesque 2013), which itself is based on (Liu et al. 2004). The general idea behind logics of limited belief is to capture, in a semantically perspicuous way, weaker forms of logical entailment. Other examples of such approaches are (Levesque 1984b; Konolige 1986; Vardi 1986; Fagin and Halpern 1988; Fagin *et al.* 1990; Lakemeyer 1996; Cadoli and Schaerf 1992; Delgrande 1995).

In the case of (Liu et al. 2004; Lakemeyer and Levesque 2013), the semantic primitive is a *setup*, a possibly infinite set of ground clauses closed under unit propagation. Roughly, the clauses in a setup can be viewed as those which the agent believes explicitly. Setups are used to give meaning to a sequence of modalities $\boldsymbol{B}_k$, for $k \geq 0$, where $\boldsymbol{B}_k\phi$ should be read as "ϕ is believed at level k." For example, given a clause c, $\boldsymbol{B}_0 c$ is satisfied by a setup s just in case

*Gerhard Lakemeyer is also affiliated with the Dept. of Computer Science at the University of Toronto.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

there is a clause in s contained in c. In other words, at level 0, belief essentially reduces to retrieval wrt. s. At level 1, the agent believes everything that is believed at level 0 and, in addition, the agent is able to reason by cases allowing to split a single clause in s. At belief level 2, the number of possible case splits increases to 2, and so on.

In order to extend this framework to a dynamic setting, we borrow from a modal variant of the epistemic situation calculus called $\mathcal{ES}$ (Lakemeyer and Levesque 2011), which features formulas of the form $[n]\alpha$ meaning that α holds after action n. Setups are then allowed to not only mention literals l but also extended literals of the form $[n_1]\cdots[n_m]l$. The new setups are thus able to talk about what is true initially and after any number of actions have been performed. As we will see, this allows us to model Reiter's basic action theories (Reiter 2001) as infinite setups. Limited belief will be defined in a way analogous to what we sketched in the previous paragraph.[1]

We then specify a reasoning service in terms of valid sentences of the form $\boldsymbol{B}_0 KB \supset [n_1]\cdots[n_m]\boldsymbol{B}_k\phi$, which may be read as "if *KB* is explicitly believed initially, then after actions n_1 to n_m ϕ is believed at level k." Here ϕ itself may mention actions and *KB* represents a basic action theory in the sense of Reiter. We will show that this reasoning service is sound wrt. the situation calculus variant $\mathcal{ES}$. We also obtain an eventual completeness result for a large class of queries in the following sense: If $KB \supset \phi$ is valid in $\mathcal{ES}$, then there is a k such that $\boldsymbol{B}_0 KB \supset \boldsymbol{B}_k\phi$ is valid in the new logic. Moreover, and perhaps most importantly, we obtain a decidability result for so-called bounded queries.

The rest of the paper is organized as follows. In the next section we briefly review the epistemic situation calculus $\mathcal{ES}$. We then introduce our new logic $\mathcal{ESL}$, which is like $\mathcal{ES}$ except that the classical belief operator $\boldsymbol{K}$ is replaced by its limited counterparts $\boldsymbol{B}_k$. We then define our reasoning service and show soundness and eventual completeness. After that we turn to decidability and then conclude.

The Logic $\mathcal{ES}$

The language of $\mathcal{ES}$ is a modal first-order dialect with equality. It comes equipped with a countably infinite set of *standard names*, which can be thought of as special constants that satisfy the unique name assumption and an infinitary version of domain closure. In this sense, the standard names can be identified with the domain of discourse. As we will see, this greatly simplifies the semantic definitions below like interpreting quantification substitutionally (existing criticism of substitutional quantification notwithstanding (Kripke 1976)).

First-order variables and standard names come in two sorts, *action* (like *repair* and *bestAction*) and *object* (like *block5* and *location*). We let $\mathcal{N}$ denote the set of all standard names and $\mathcal{Z}$ denote the set of all sequences of standard names for actions, including $\langle\,\rangle$, the empty sequence.

A term is either a variable or a standard name. The *well-formed formulas* of the language form the least set such that

1. If $t_1, \ldots, t_k$ are terms, and P is a k-ary predicate symbol then $P(t_1, \ldots, t_k)$ is an (atomic) formula;
2. If t_1 and t_2 are terms, then $(t_1 = t_2)$ is a formula;
3. If t is an action term and α is a formula, then $[t]\alpha$ is a formula;
4. If α and β are formulas, and v is a first-order variable, then the following are also formulas: $(\alpha \vee \beta)$, $\neg\alpha$, $\exists v.\,\alpha$, $\Box\alpha$, $\boldsymbol{K}\alpha$.

We read $[n]\alpha$ as "α holds after action n", and $\Box\alpha$ as "α holds after any sequence of actions." We will freely use $\wedge, \supset, \equiv$, and $\forall$ as the usual abbreviations. There are two distinguished predicates *SF* and *Poss*: $Poss(n)$ says that action n can be executed, and $SF(n)$ says that action n returns a binary sensing result of 1 or 0.

Given a sequence of actions $z = \langle z_1, z_2, \ldots, z_n\rangle$ and a formula α, we usually write $[z]\alpha$ instead of $[z_1][z_2]\cdots[z_n]\alpha$. When z is the empty sequence, then $[z]\alpha = \alpha$. For action sequences z and z' we also write $z \cdot z'$ to mean the action sequence consisting of z followed by z'. We write $z \sqsubseteq z'$ when z is a prefix of z'. For a set of action sequences Z, we write $z \sqsubseteq Z$ to mean that there is some $z' \in Z$ such that $z \sqsubseteq z'$.

We call a formula without free variables a *sentence*. By a *primitive sentence* we mean a formula of the form $P(n_1, \ldots, n_m)$ where P is a predicate symbol and all of the n_i are standard names. For simplicity all predicates are considered to be fluent, that is, they may vary as a result of actions and their value may be unknown. It is convenient to distinguish between certain types of formulas using the following terminology:

- a formula with no $\Box$ operators is called *bounded*;
- a formula with no $\Box$ or $[t]$ operators is called *static;*
- a formula with no $\boldsymbol{K}$ operators is called *objective;*
- a formula with no fluent, $\Box$, or $[t]$ operators outside the scope of a $\boldsymbol{K}$ is called *subjective;*
- a formula with no $\boldsymbol{K}$, $\Box$, $[t]$, *Poss*, or *SF* is called a *fluent* formula;

Intuitively, to determine whether or not a sentence α is true after a sequence of actions z has been performed, we need to specify two things: a world w and an epistemic state e. We write $e, w, z \models \alpha$. A world determines truth values for the primitive sentences after any sequence of actions. An epistemic state is defined by a set of worlds, as in possible-world semantics. More precisely:

- a world $w \in W$ is any function from the primitive sentences and $\mathcal{Z}$ to $\{0, 1\}$.
- an epistemic state $e \subseteq W$ is any set of worlds.

To interpret what is known after a sequence of actions possibly including sensing has taken place, we define $w' \simeq_z w$ (read: w' and w agree on the sensing throughout action sequence z) inductively by the following:

1. $w' \simeq_{\langle\,\rangle} w$ for all w';
2. $w' \simeq_{z\cdot n} w$ iff
   $w' \simeq_z w$ and $w'[SF(n), z] = w[SF(n), z]$.

[1] A preliminary version of our model was developed by Capes (2010).

Putting all these together, here is the semantic definition of truth. Given $e \subseteq W$ and $w \in W$, we define $e, w \models \alpha$ (read: α is true) as $e, w, \langle\,\rangle \models \alpha$, where for any $z \in \mathcal{Z}$:

1. $e, w, z \models P(n_1, \ldots, n_m)$ iff $w[P(n_1, \ldots, n_m), z] = 1$, where $P(n_1, \ldots, n_m)$ is a primitive sentence;
2. $e, w, z \models (n_1 = n_2)$ iff n_1 and n_2 are identical standard names;
3. $e, w, z \models (\alpha \vee \beta)$ iff $e, w, z \models \alpha$ or $e, w, z \models \beta$;
4. $e, w, z \models \neg\alpha$ iff $e, w, z \not\models \alpha$;
5. $e, w, z \models \exists v.\, \alpha$ iff $e, w, z \models \alpha^x_n$, for some standard name n (of the same sort as v);
6. $e, w, z \models [n]\alpha$ iff $e, w, z \cdot n \models \alpha$;
7. $e, w, z \models \Box\alpha$ iff $e, w, z \cdot z' \models \alpha$, for every $z' \in \mathcal{Z}$;
8. $e, w, z \models \boldsymbol{K}\alpha$ iff $e, w', z \models \alpha$, for every $w' \in e$ such that $w' \simeq_z w$.

When α is *objective* (has no $\boldsymbol{K}$ operators), we can leave out the e and write $w \models \alpha$. When Σ is a set of sentences and α is a sentence, we write $\Sigma \models \alpha$ (read: Σ logically entails α) to mean that for every e and w, if $e, w \models \alpha'$ for every $\alpha' \in \Sigma$, then $e, w \models \alpha$. Finally, we write $\models \alpha$ (read: α is valid) to mean $\{\} \models \alpha$.

We will not go into details of the logical properties of $\mathcal{ES}$ except to note that $\boldsymbol{K}$ behaves like an ordinary *K45* modal operator (Chellas 1980), after any number of actions.

Basic Action Theories

While $\mathcal{ES}$ in general is less expressive than the classical situation calculus, we showed in (Lakemeyer and Levesque 2011) that it suffices to fully capture Reiter's *basic action theories (BATs)*.

Definition 1 *Given a set of fluents $\mathcal{F}$, a set $\Sigma \subseteq \mathcal{ES}$ of sentences is called a basic action theory over $\mathcal{F}$ iff $\Sigma = \Sigma_0 \cup \Sigma_{\text{pre}} \cup \Sigma_{\text{post}} \cup \Sigma_{\text{sense}}$ where Σ mentions only fluents in $\mathcal{F}$ and*

1. *Σ_0 is any set of fluent sentences;*
2. *Σ_{pre} is a singleton sentence of the form $\Box$Poss$(a) \equiv \pi$, where π is a fluent formula;*[2]
3. *Σ_{post} is a set of sentences of the form $\Box[a]F(\vec{v}) \equiv \gamma_F$, one for each relational fluent F, where γ_F is a fluent formula.*[3]
4. *Σ_{sense} is a sentence (parallel to the one for Poss) of the form $\Box$SF$(a) \equiv \varphi$, where φ is a fluent formula.*

Σ_0 is intended to express what is true initially; Σ_{pre} characterizes the preconditions of actions in a single axiom; Σ_{post} contains the *successor state axioms*, one per fluent, which incorporate Reiter's solution to the frame problem (Reiter 2001); following (Scherl and Levesque 2003), Σ_{sense} expresses the sensing outcomes of every action. (In case an action does not return a meaningful sensing results, the value of *SF* can be defined to be vacuously true.)

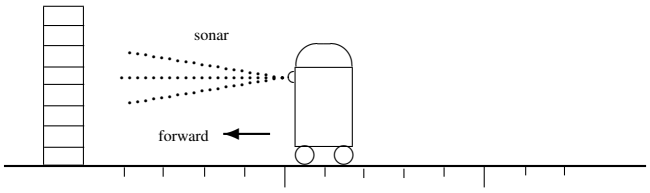


Figure 1: A simple robot

To illustrate BATs, let us consider the example depicted in Figure 1. Here a robot has two actions: *forward* moves it one unit forward towards the wall, and *sonar* is a sensing action, which returns 1 if the robot is close to the wall. Locations are denoted by propositional fluents d_i, which may be read as "the robot is located i units from the wall."[4]

$$\Box\,\textit{Poss}(a) \equiv a = \textit{forward} \wedge \neg d_0 \vee a = \textit{sonar} \wedge \text{TRUE}.$$

In other words, *sonar* is always possible, and *forward* is executable only when the robot is not already at the wall. Next, we define the sensing results for the actions:

$$\Box\,\textit{SF}(a) \equiv a = \textit{forward} \wedge \text{TRUE} \vee a = \textit{sonar} \wedge (d_0 \vee d_1).$$

While *forward* always returns *true* as a sensing result, $\textit{SF}(\textit{sonar})$ is 1 in case the robot is either at location d_0 or d_1. Finally, here are the successor state axioms for the d_i:

$$\Box\,[a]d_i \equiv a = \textit{forward} \wedge d_{i+1} \vee a \neq \textit{forward} \wedge d_i \quad \text{for } i > 0.$$

$$\Box\,[a]d_0 \equiv a = \textit{forward} \wedge d_1 \vee d_0$$

In other words, after *forward*, d_i becomes true if d_{i+1} was true before, and false otherwise, except in the case of d_0, which remains true after *forward* was executed in d_1. After *sonar* the value of d_i does not change.

For a given basic action theory Σ, projection queries, which are bounded sentences, are of particular importance, as they are a basic ingredient of action languages such as Golog (Levesque et al. 1997), among other things.

Let Σ consist of the above axioms together with $(d_2 \vee d_3), \neg d_1$, and $\neg d_0$. Then it follows from $\boldsymbol{K}\Sigma$ that the robot believes that it is initially either 2 or 3 units away from the wall. Suppose the robot is actually 2 units away from the wall. Abbreviating *forward* as f and *sonar* as s, respectively, let $S = \textit{SF}(f) \wedge [f]\textit{SF}(s)$ represent the actual sensing values obtained when executing f and s after f. We write *Close* instead of $(d_0 \vee d_1)$. Then we have:

[2] We assume that $\Box$ has lower syntactic precedence than the logical connectives, so that $\Box\textit{Poss}(a) \equiv \pi$ stands for $\forall a.\Box(\textit{Poss}(a) \equiv \pi)$.

[3] The $[t]$ construct has higher precedence than the logical connectives. So $\Box[a]F(\vec{x}) \equiv \gamma_F$ abbreviates $\forall a.\Box([a]F(\vec{x})) \equiv \gamma_F$.

[4] This way of handling distances is needed because we are not assuming any sort of built-in arithmetic in $\mathcal{ESL}$.

1. $\models \boldsymbol{K}\Sigma \supset \boldsymbol{K}\neg \mathit{Close}$;
2. $\not\models S \wedge \boldsymbol{K}\Sigma \supset [f]\boldsymbol{K}\mathit{Close}$;
3. $\models S \wedge \boldsymbol{K}\Sigma \supset [f][s]\boldsymbol{K}\mathit{Close}$;
4. $\models S \wedge \boldsymbol{K}\Sigma \supset \boldsymbol{K}[f][f]\mathit{Close}$;

In other words: initially the robot knows that it is not close to the wall (1); it does not know that it is close after executing a forward action (2); it knows that it is close after a forward and sensing the distance to the wall (3); it knows that it will be close if it were to perform two forward actions (4).

Since $\mathcal{ES}$ is a full first-order language, answering projection queries is clearly undecidable, even in the case without any actions at all. So let us now consider a variant of $\mathcal{ES}$ with a limited model of belief as the basis for deciding projection queries.

The Logic $\mathcal{ESL}$

The language of $\mathcal{ESL}$ is that of $\mathcal{ES}$ except that the operator $\boldsymbol{K}$ is replaced by operators $\boldsymbol{B}_k$ for $k = 0, 1, 2, \ldots$ For the purposes of this paper, we also assume that the $\boldsymbol{B}_k$ do not occur nested. The definitions of bounded, static, objective, subjective, and fluent formulas carry over in the obvious way, with $\boldsymbol{B}_k$ replacing $\boldsymbol{K}$.

As extended forms of literals and clauses play an important role in the semantics, we introduce them here together with some additional terminology. An *extended literal* is a sequence of actions (possibly empty) followed by a literal, where a literal is either an atomic formula not mentioning $=$ or its negation. $\neg P(n)$ and $[m_1][m_2]P(n)$ are examples of extended literals. The complement of an extended literal l, denoted as $\bar{l}$, is obtained by complementing the literal it contains. For example, the complement of $[m_1][m_2]P(n)$ is $[m_1][m_2]\neg P(n)$. An *extended clause* is a disjunction of extended literals. Often an extended clause is identified with the set of extended literals it contains. In the following we will often refer to extended clauses simply as clauses.

The empty clause is denoted as $[]$. A unit clause is a clause with a single extended literal. A primitive clause is a clause whose atomic formulas are primitive sentences, and similar for primitive literals. When z is a sequence of actions and c a clause we write $[z]c$ to mean c with every extended literal l replaced by $[z]l$.

A *setup* is a (possibly infinite) set of primitive clauses. For any setup s, the closure of s under unit propagation, which we denote as $\mathsf{UP}(s)$, is defined as the least set s' which contains s, and if unit clause $l \in s'$ and $\{\bar{l}\} \cup c \in s'$, then $c \in s'$. $\mathsf{VP}(s)$ is defined as the set $\{c \mid c \text{ is a primitive clause and there exists a } c' \in \mathsf{UP}(s) \text{ such that } c' \subseteq c\}$.

Similar to $\mathcal{ES}$ the semantics of $\mathcal{ESL}$ is defined wrt. a world, an agent's epistemic state, and a sequence of actions. While worlds and action sequences are the same as in $\mathcal{ES}$, the epistemic state is now characterized by a setup. Intuitively, the clauses in a setup determine what the agent believes explicitly, or at level 0. Beliefs at higher levels are then obtained by reasoning by cases.

Given a world w, a setup s, and an action sequence z, the truth of a sentence α, written as $s, w, z \models \alpha$, is inductively defined as follows:

1. $s, w, z \models P(n_1, \ldots, n_m)$ iff $w[P(n_1, \ldots, n_m), z] = 1$, where $P(n_1, \ldots, n_m)$ is a primitive formula;
2. $s, w, z \models (n_1 = n_2)$ iff n_1 and n_2 are identical;
3. $s, w, z \models (\alpha \vee \beta)$ iff $s, w, z \models \alpha$ or $s, w, z \models \beta$;
4. $s, w, z \models \neg\alpha$ iff $s, w, z \not\models \alpha$;
5. $s, w, z \models \exists x.\, \alpha$ iff $s, w, z \models \alpha^x_n$, for some standard name n (of the same sort as x);
6. $s, w, z \models [n]\alpha$ iff $s \cup \{[z] \pm \mathit{SF}(n)\}, w, z \cdot n \models \alpha$;[5]
7. $s, w, z \models \Box\alpha$ iff $s, w, z \models [z']\alpha$, for all $z' \in \mathcal{Z}$;
8. $s, w, z \models \boldsymbol{B}_k\alpha$ iff $s, z, k \models \alpha$.

Note that Rules 1–5 are essentially identical to those of $\mathcal{ES}$. Rule 6 differs in that we explicitly remember the sensing result of the action in s. Roughly, this has the same effect as the compatibility relation $\simeq_z$ of $\mathcal{ES}$ when it comes to interpreting belief. Rule 7 is again essentially the same as in $\mathcal{ES}$. Let us now turn to the semantics of limited belief (Rule 8). Let α be objective. $s, z, k \models \alpha$ is the least relation such that

9. $s, z, k \models \alpha$ if $[] \in \mathsf{UP}(s)$;
10. $s, z, k \models c$ if $k = 0$, c is a clause and $[z]c \in \mathsf{VP}(s)$;
11. $s, z, k \models \alpha$ if $k > 0$ and there is an extended primitive literal l such that $s \cup \{l\}, z, k-1 \models \alpha$ and $s \cup \{\bar{l}\}, z, k-1 \models \alpha$;
12. $s, z, k \models (n = m)$ if n, m are identical std. names; $s, z, k \models \neg(n = m)$ if n, m are distinct std. names;
13. $s, z, k \models \neg\neg\alpha$ if $s, z, k \models \alpha$;
14. $s, z, k \models (\alpha \vee \beta)$ if $s, z, k \models \alpha$ or $s, z, k \models \beta$; $s, z, k \models \neg(\alpha \vee \beta)$ if $s, z, k \models \neg\alpha$ and $s, z, k \models \neg\beta$;
15. $s, z, k \models \exists x.\alpha$ if $s, z, k \models \alpha^x_n$ for some n; $s, z, k \models \neg\exists x.\alpha$ if $s, z, k \models \neg\alpha^x_n$ for all n;
16. $s, z, k \models [n]\alpha$ if $s \cup \{[z]\mathit{SF}(n)\}, z \cdot n, k \models \alpha$ and $s \cup \{[z]\neg\mathit{SF}(n)\}, z \cdot n, k \models \alpha$; $s, z, k \models \neg[n]\alpha$ if $s \cup \{[z]\mathit{SF}(n)\}, z \cdot n, k \models \neg\alpha$ and $s \cup \{[z]\neg\mathit{SF}(n)\}, z \cdot n, k \models \neg\alpha$;
17. $s, z, k \models \Box\alpha$ if $s, z, k \models [z']\alpha$ for all $z' \in \mathcal{Z}$; $s, z, k \models \neg\Box\alpha$ if $s, z, k \models [z']\neg\alpha$ for some $z' \in \mathcal{Z}$.

Rules 9–15 are similar to those proposed for limited belief in the static case (Lakemeyer and Levesque 2013). Rule 9 says that whenever the empty clause can be derived by unit propagation everything is believed. Rule 10, which we will also refer to as the subsumption rule, says that a clause c is believed at level 0 after z just in case $[z]c$ is subsumed by a clause in $\mathsf{UP}(s)$. Rule 11, which we will also refer to as the split rule, says that α is believed at level $k > 0$ if the belief holds at level $k-1$ after splitting on an extended literal.[6] Rule 12 deals with equality in a classical way, independent of any setup. In other words, we assume perfect reasoning

[5] Here $\pm\mathit{SF}(n)$ stands for $\mathit{SF}(n)$ if $w[\mathit{SF}(n), z] = 1$ and $\neg\mathit{SF}(n)$ if $w[\mathit{SF}(n), z] = 0$, respectively.

[6] In (Lakemeyer and Levesque 2013) we considered splitting a clause in s instead. As we will see below, splitting an arbitrary literal has certain advantages.

when it comes to equality. Rules 13–15 make sure that certain "obvious" beliefs are obtained. For example, to believe $(\alpha \vee \beta)$ at level k it suffices to believe either α or β. Rule 16 considers the case of believing that α (or its negation) holds after an action n. Note that in thinking about what will hold after action n, the agent has as yet no access to the sensing information that will result from performing this action. Therefore it needs to entertain both possibilities ($SF(n)$ and $\neg SF(n)$) when evaluating whether α is believed after n. Finally, $\Box$ is handled in the obvious way by considering all action sequences. Note that Rules 7 and 17 are well-founded when including the number of $\Box$ operators occurring in α in the measure for α.[7]

When a formula α is objective, we sometimes write $w, z \models \alpha$ instead of $s, w, z \models \alpha$. For subjective α, we write $s, z \models \alpha$. If, in addition, z is empty, we simply write $w \models \alpha$ and $s \models \alpha$, respectively.

To conclude the semantic definition, we say that a set of sentences Σ logically entails a sentence α (written $\Sigma \models \alpha$) if for every setup s and world w, if $s, w, \langle\,\rangle \models \alpha'$ for every $\alpha' \in \Sigma$, then $s, w, \langle\,\rangle \models \alpha$. Finally, α is valid (written $\models \alpha$) if $\{\} \models \alpha$. (So far we have used $\models$ for entailment in both $\mathcal{ES}$ and $\mathcal{ESL}$. In the following, we will write $\models_{\mathcal{ES}}$ when referring to entailment in $\mathcal{ES}$.)

Some Properties

Here we will not go into a detailed discussion of the properties of $\mathcal{ESL}$ except to note the following. The logic of limited belief presented in (Lakemeyer and Levesque 2013) is very similar to $\mathcal{ESL}$ when restricted to formulas without nested beliefs. This is because, in the case of static formulas, the semantics of both logics is the same except for the split rule. The same applies to a slight variant of the logic presented in (Liu et al. 2004), and the properties discussed there for static formulas generally carry over in a straightforward way to the dynamic case. For example, a simple induction proof establishes that for any k

$\models \Box(\boldsymbol{B}_k \neg\neg\alpha \equiv \boldsymbol{B}_k\alpha)$
$\models \Box(\boldsymbol{B}_k(\alpha \wedge \beta) \equiv \boldsymbol{B}_k\alpha \wedge \boldsymbol{B}_k\beta)$
$\models \Box(\boldsymbol{B}_k \forall x.\alpha \equiv \forall x.\boldsymbol{B}_k\alpha)$

While most of the equivalence preserving transformations of formulas such as removing double negations are preserved at all belief levels, there are exceptions. For example,

$$\boldsymbol{B}_k(p \wedge (q \vee r)) \supset \boldsymbol{B}_k((p \wedge q) \vee (p \wedge r))$$

is not valid. See (Liu et al. 2004) for a detailed discussion of why this is so.

Intuitively, one would expect belief to be monotonic in the sense that the beliefs held at a certain setup and level of belief continue to hold when moving to larger setups or increasing the level of belief. We now show that this is indeed the case.

Theorem 1 *Let s and s' be setups such that $s \subseteq s'$. Then for any z, k, and objective ϕ, if $s, z, k \models \phi$ then $s', z, k \models \phi$.*

[7] We remark that there are equivalent versions of Rule 7 and 17, where the sequence z' is only mentioned on the left-hand side of the rules, but at the expense of a more complicated definition.

Proof: The proof is by induction on k and the structure of ϕ. Let $k = 0$. The only interesting case is when ϕ is a clause c. It is easy to show that $\mathsf{VP}(s) \subseteq \mathsf{VP}(s')$ given that $s \subseteq s'$. Hence, if $[z]c \in \mathsf{VP}(s)$ then $[z]c \in \mathsf{VP}(s')$. Note also that, in the case of actions (Rule 16+17), the same *SF*-formulas are added to s and s' so that the subset relation between the augmented setups continues to hold.

Suppose the lemma holds for some k and let $s, z, k+1 \models \phi$. Again, the proof is by induction on the structure of ϕ. Here the only interesting case is the split rule. So suppose, there is some extended literal l such that $s \cup \{l\}, z, k \models \phi$ and $s \cup \{\bar{l}\}, z, k \models \phi$. In both cases we can apply the induction hypothesis for k and obtain $s', z, k+1 \models \phi$. ∎

Corollary 1 *If $\mathsf{VP}(s) \subseteq \mathsf{VP}(s')$ then $s, z, k \models \phi$ implies $s', z, k \models \phi$.*

Theorem 2 *If $s, z, k \models \phi$ then $s, z, k+1 \models \phi$.*

Proof: Let $s, z, k \models \phi$ and let l be any extended literal. Then, by Theorem 1, $s \cup \{l\}, z, k \models \phi$ and $s \cup \{\bar{l}\}, z, k \models \phi$. Hence, by the split rule, $s, z, k+1 \models \phi$. ∎

As an immediate corollary we obtain

Corollary 2 $\models \Box(\boldsymbol{B}_k\phi \supset \boldsymbol{B}_{k+1}\phi)$.

Eventual completeness

When considering sentences e that only mention $=$, limited belief behaves exactly like $\boldsymbol{K}$, even at level 0. A simple induction shows that $\models_{\mathcal{ES}} \Box(\boldsymbol{K}e \equiv e)$ and $\models \Box(\boldsymbol{B}_0 e \equiv e)$, that is, e is believed just in case e is true, after any number of actions. In general, of course, $\boldsymbol{B}_k$ is much weaker than $\boldsymbol{K}$, especially when k is small. For example, while $\boldsymbol{K}(p \vee \neg p)$ is valid, $\boldsymbol{B}_0(p \vee \neg p)$ is not. For example, if s is the empty setup, then $s, \langle\,\rangle \not\models \boldsymbol{B}_0(p \vee \neg p)$ because neither Rule 10 nor Rule 14 applies. On the other hand, $\boldsymbol{B}_1(p \vee \neg p)$ is valid, since we can apply the split rule using p. This is not accidental. We will now show that for any objective valid bounded propositional formula ϕ of $\mathcal{ESL}$ we can always find a k such $\boldsymbol{B}_k\phi$ is valid. By a propositional formula we mean a formula which does not contain quantifiers and which does not mention $=$.

We begin with a useful lemma that shows the connection between worlds and their corresponding setups for arbitrary objective formulas.

Definition 2 *Given a world w let*

$$s(w) = \{l \mid l \text{ is an extended literal and } w \models l\}.$$

In other words, $s(w)$ is the setup consisting of all extended literals true at w. Note that $[z]l \in s(w)$ iff $w, z \models l$ for any literal l.

Lemma 1 *For any objective ϕ and sequence of actions z, $s(w), z, k \models \phi$ iff $w, z \models \phi$ for all k.*

Proof: We first prove the lemma for $k = 0$ by induction on the structure of ϕ. For a clause c we have $s(w), z, 0 \models c$ iff $[z]l \in s(w)$ for some $l \in c$ (by the subsumption rule) iff $w, z \models l$ for some $l \in c$ (by the construction of $s(w)$) iff $w, z \models c$. The lemma clearly holds for $(n = m)$ and $\neg(n =$

m) and also (by induction) for $\neg\neg\phi$, $(\phi \wedge \psi)$, $\neg(\phi \wedge \psi)$, $\forall x.\phi$, and $\neg\forall x.\phi$.

$s(w), z, 0 \models [n]\phi$ iff $s(w) \cup \{[z]SF(n)\}, z \cdot n, 0 \models \phi$ and $s(w) \cup \{[z]\neg SF(n)\}, z \cdot n, 0 \models \phi$ iff $s(w), z \cdot n, 0 \models \phi$ (because $s(w)$ already contains either $[z]SF(n)$ or $[z]\neg SF(n)$ and hence either does not change by adding the extended literal or it becomes inconsistent so that Rule 9 applies) iff $w, z \cdot n \models \phi$ (by induction) iff $w, z \models [n]\phi$. (The case for $\neg[n]\phi$ is similar.) $s(w), z, 0 \models \Box\phi$ iff $s(w) \cup \sigma, z, 0 \models [z']\phi$ for all z' iff $s(w), z \cdot z', 0 \models \phi$ for all z' (since, similar to the case of a single action, adding all combinations of sensing results for z' to $s(w)$ leads either to inconsistency or leaves $s(w)$ as is) iff $w, z \cdot z' \models \phi$ for all z' iff $w, z \models \Box\phi$.

We now show that $s(w), z, k \models \phi$ iff $s(w), z, k+1 \models \phi$, from which the lemma follows. The only-if direction follows from Lemma 2. For the if direction, suppose $s(w), z, k+1 \models \phi$. The proof is by induction on the structure of ϕ. For a clause c and since $k + 1 > 0$, the only applicable rule is the split rule. Hence there is an extended literal l such that $s(w) \cup \{l\}, z, k \models \phi$ and $s(w) \cup \{\bar{l}\}, z, k \models \phi$. Since $s(w)$ is a complete set of literals, either l or $\bar{l}$ is in $s(w)$. Suppose $l \in s(w)$. Then $s(w) \cup \{l\} = s(w)$ and, hence, $s(w), z, k \models \phi$. The other case is symmetric. The case of $=$-expression is obvious. For the induction step, if any of the rules (12)–(17) apply, then the induction hypothesis applies immediately. In case of the split rule the argument is the same as above. ∎

For the rest of this subsection, we only consider propositional bounded formulas. The following definition characterizes the positive extended literals contained in a formula, or PEL(ϕ) for short.

Definition 3 *Let ϕ be bounded and objective. Then PEL(ϕ) is defined inductively as follows:*

PEL(p) = $\{p\}$ when p is a primitive sentence;
PEL($\neg\phi$) = PEL(ϕ);
PEL($\phi \vee \psi$) = PEL(ϕ) $\cup$ PEL(ψ);
PEL($[n]\phi$) = $\{[n \cdot z]p \mid [z]p \in$ PEL(ϕ)$\}$.

For example, if $\phi = [n](p \vee [m](\neg q \vee [l]r))$ then PEL(ϕ) = $\{[n]p, [n \cdot m]q, [n \cdot m \cdot l]r\}$. Note that PEL ignores negations. Let PEL(ϕ)$_z$ = $\{[z \cdot z']p \mid [z']p \in$ PEL(ϕ)$\}$ for any action sequence z.

Definition 4 (Worlds compatible with a setup) *Let s be a setup. Then*

$$C(s) = \{w \mid w \text{ is a world and } w \models c \text{ for all } c \in s\}$$

Lemma 2 *Let $A = \{p_1, \ldots, p_n\}$ be a set of positive extended literals and $L = \{l_1, \ldots, l_n\}$ with $l_i = p_i$ or $l_i = \overline{p}_i$. Let s be a setup such that $L \subseteq s$. Then for all sequences of actions z, all ϕ with PEL$_z(\phi) \subseteq A$, and all $w \in C(s)$, $w, z \models \phi$ iff $s, z, 0 \models \phi$.*

Proof: First note that in case s is inconsistent the lemma holds vacuously, since there are no compatible worlds in this case. Now suppose s is consistent. Then s does not contain $\bar{l}$ for any $l \in L$ as s already contains l.

The proof is by induction on the structure of ϕ. For a clause c suppose $w, z \models c$. Then $w, z \models l$ for some extended literal $l \in c$. By assumption, either $[z]l \in A$ or $[z]\bar{l} \in A$. Since w is compatible with s and, in particular, with L, $[z]l \in L$ and hence $[z]l \in s$ by assumption. Therefore, by the subsumption rule, $s, z, 0 \models c$.

Conversely, let $s, z, 0 \models c$. Then $[z]c \in \mathsf{VP}(s)$ by subsumption. Since $L \subseteq s$ and since s is consistent, there must be some extended literal $l \in c$ such that $[z]l \in s$ and $[z]l \in L$. Thus $w, z \models l$ follows and, hence, $w, z \models c$.

The cases for $\neg\neg\phi$, $(\phi \vee \psi)$, and $\neg(\phi \vee \psi)$ follow trivially by induction. Let $w, z \models [n]\phi$. Then $w, z \cdot n \models \phi$ and, by induction, $s, z \cdot n, 0 \models \phi$. (Note that PEL$_z([n]\phi)$ = PEL$_{z \cdot n}(\phi)$. By monotonicity, $s \cup \{[z]SF(n)\}, z \cdot n, 0 \models \phi$ and $s \cup \{[z]\neg SF(n)\}, z \cdot n, 0 \models \phi$. Hence, $s, z, 0 \models [n]\phi$. Conversely, let $s, z, 0 \models [n]\phi$. Then $s \cup \{[z]SF(n)\}, z \cdot n, 0 \models \phi$ and $s \cup \{[z]\neg SF(n)\}, z \cdot n, 0 \models \phi$. Suppose $w, z \models SF(n)$ (the other case is symmetric). Then $w \in$ C$(s \cup \{[z]SF(n)\})$. Hence, by induction, $w, z \cdot n \models \phi$ and, therefore $w, z \models [n]\phi$. ∎

We are now ready to prove our eventual-completeness result. The interesting part is that any valid propositional, bounded ϕ is believed at some level k. Roughly, this is because we can always split on every extended literal contained in ϕ so that the resulting setups are essentially complete valuations of all the extended literals mentioned.

Theorem 3 *For any propositional, bounded, objective ϕ, $\models \phi$ iff $\models \boldsymbol{B}_k\phi$ for some k.*

Proof: For the if direction, suppose $\models \boldsymbol{B}_k\phi$ for some k and consider any world w. Then $s(w), \langle\,\rangle, k \models \phi$ and, by Lemma 1, $w, \langle\,\rangle \models \phi$.

Conversely, let $\models \phi$. Let n be the size of PEL(ϕ), and for any setup s, let $\nu(s)$ be the number of elements p of PEL(ϕ) such that either p or $\overline{p}$ appears in s. We will first show that for any s and $k = n - \nu(s)$, $s \models \boldsymbol{B}_k\phi$. The proof is by induction on k.

For $k = 0$ we need to show that $s, \langle\,\rangle, 0 \models \phi$. If $[] \in \mathsf{UP}(s)$ then clearly $s, \langle\,\rangle, 0 \models \phi$ because of Rule 9. Otherwise, since $\nu(s) = n$, we have that for every atom p in ϕ, exactly one of p or $\overline{p}$ is in s. Let $L = \{l_1, \ldots, l_n\}$ be the set of all such literals. Let $w \in$ C(L). Since $w \models \phi$ by assumption, $L \models \boldsymbol{B}_0\phi$ holds by Lemma 2. Since $L \subseteq s$, by Theorem 1, $s \models \boldsymbol{B}_0\phi$.

Suppose for all s with $k = n - \nu(s)$ and $0 \leq k < n$, $s \models \boldsymbol{B}_k\phi$. Now consider a setup s' with $n - \nu(s') = k + 1$. Let p be an atom occurring in ϕ such that neither p nor $\overline{p}$ is in s'. Such p must exist since $n \neq \nu(s')$ by assumption. Since $n - \nu(s' \cup \{p\}) = n - \nu(s' \cup \{\overline{p}\}) = k$, by induction, $s' \cup \{p\} \models \boldsymbol{B}_k\phi$ and $s' \cup \{\neg p\} \models \boldsymbol{B}_k\phi$. Since ϕ is objective, it follows that $s' \cup \{p\}, \langle\,\rangle, k \models \phi$ and $s' \cup \{\neg p\}, \langle\,\rangle, k \models \phi$. Hence, using the split rule, we obtain, $s', \langle\,\rangle, k+1 \models \phi$, that is, $s' \models \boldsymbol{B}_{k+1}\phi$.

To complete the proof of the only-if direction, let s be any setup and let $k = n$. By the above argument, $s \models \boldsymbol{B}_{k'}\phi$ for $k' = n - \nu(s)$. Since $0 \leq k' \leq n$, $s \models \boldsymbol{B}_k\phi$ (using $\models \boldsymbol{B}_{k'}\phi \supset \boldsymbol{B}_{k'+1}\phi$ if needed). ∎

We remark that this significantly strengthens an earlier result by Liu et al. (2004), who showed that eventual

completeness in the static case obtains only when querying knowledge bases and only when the query belongs to a restricted class of formulas in a normal form defined in (Levesque 1998). The reason for our stronger result is our modified split rule. While Liu et al. restricted splitting to clauses in the setup, we allow splitting arbitrary literals, which is strictly more general. In the next section we will revisit the issue of eventual completeness also in the context of querying knowledge bases.

A Reasoning Service Based on $\mathcal{ESL}$

We now turn to the use of the logic as a specification of a reasoning service for an agent with limited beliefs. In particular, we phrase it in terms of the valid sentences of the following form:

$$\boldsymbol{B}_0\phi \supset [z]\boldsymbol{B}_k\psi, \tag{1}$$

where ϕ typically encodes a basic action theory of the kind introduced earlier, and ψ is a bounded formula, that is, it may contain actions but not $\Box$. We will refer to sentences (1) as *belief implications*, and they can be read as "if ϕ is explicitly believed initially then after actions z have occurred ψ is believed at level k." Belief implications specify a reasoning service in the sense that determining beliefs at level k after some actions requires computing whether the corresponding belief implication is valid.

We begin our investigations by showing that belief implications are sound with respect to the original epistemic situation calculus $\mathcal{ES}$.

Theorem 4 *If* $\models \boldsymbol{B}_0\alpha \supset [z]\boldsymbol{B}_k\beta$ *then* $\models_{\mathcal{ES}} \boldsymbol{K}\alpha \supset [z]\boldsymbol{K}\beta$.

Proof: Here we prove it for the case of a single action n. Generalizing it to an arbitrary z is straightforward. Suppose $\models \boldsymbol{B}_0\alpha \supset [n]\boldsymbol{B}_k\beta$ and let $e, w \models \boldsymbol{K}\alpha$. Then $w' \models \alpha$ for all $w' \in e$. Then $s(w'), \langle\,\rangle, 0 \models \alpha$ by Lemma 1. Let $w' \in e$ and suppose $w' \simeq_n w$. Thus $s(w'), w \models \boldsymbol{B}_0\alpha$. By assumption, $s(w') \cup \{\pm SF(n)\}, w, \langle n\rangle \models \boldsymbol{B}_k\beta$ and, therefore, $s(w') \cup \{\pm SF(n)\}, \langle n\rangle, k \models \beta$. Since $w' \simeq_z w$ we have $s(w') \cup \{\pm SF(n)\} = s(w')$ and hence, by Lemma 1, $w', \langle n\rangle \models \beta$. As this holds for all $w' \in e$ such that $w' \simeq_n w$, $e, w \models [n]\boldsymbol{K}\beta$ follows. ∎

Liu et al. (2004) considered so-called proper$^+$ knowledge bases, for which they showed that reasoning under limited belief is decidable in the static case. We will now extend this idea to knowledge bases representing basic action theories. In the following we let e range over *ewffs*, which are quantifier-free fluent formulas that mention only $=$ and no other predicate symbols. Let θ be a mapping from variables to standard names (of the appropriate sort). Then we write $e\theta$ to mean e with each variable substituted by the standard name according to θ. For a formula α we write $\forall\alpha$ to denote its universal closure.

Definition 5 (proper$^+$ Knowledge Bases) *Let e be an ewff and c an extended clause. Then we call a formula of the form $\forall(e \supset c)$ a $\forall$-clause and a formula of the form $\Box\forall(e \supset c)$ a $\Box$-clause. A KB is called proper$^+$ if it is a finite set of $\forall$- and $\Box$-clauses.*

Given a proper$^+$ KB, we let gnd(KB) be the infinite setup which is the union of $\{c\theta \mid \forall(e \supset c) \in KB \text{ and } \models e\theta\}$ and $\{[z]c\theta \mid \Box\forall(e \supset c) \in KB, \models e\theta, \text{ and } z \in \mathcal{Z}\}$.

In the following we will not consider proper$^+$ *KB*s in their full generality but only those which are the translation of a basic action theory as defined earlier. We call such *KB*s *proper$^+$ BATs*. It will also be necessary to consider proper$^+$ BATs augmented with a finite set of extended literals. We call such *KB*s augmented proper$^+$ BATs.

Note that a proper$^+$ BAT has two kinds of $\Box$-formulas: Those $\Box\forall(e \supset c)$ which are the result of the translation of a successor state axiom are such that c mentions a single extended literal of the form $[a]l$, where l is a literal and a is an action variable. All other $\Box\forall(e \supset c)$ like the ones resulting from the translation of the *SF*-axiom are such that c mentions only literals and no actions, that is, c is static.

As an example, let us consider the BAT for our robot example. It is easy to translate the original axioms into a proper$^+$ KB. Here we consider only the translations of the sensing axiom and the successor state axiom for d_i with $i > 0$. The *SF*-axiom can be represented by the following five $\Box$-clauses:

$\Box\forall(a \neq \mathit{forward} \supset (\neg SF(a) \vee d_0 \vee d_1))$;
$\Box\forall(a \neq \mathit{forward} \wedge a \neq \mathit{sonar} \supset \neg SF(a))$;
$\Box\forall(a = \mathit{forward} \supset SF(a))$;
$\Box\forall(a = \mathit{sonar} \supset (\neg d_0 \vee SF(a)))$;
$\Box\forall(a = \mathit{sonar} \supset (\neg d_1 \vee SF(a)))$.

The successor state axiom for d_i with $i > 0$ is captured by these $\Box$-clauses:

$\Box\forall(a = \mathit{forward} \supset (\neg d_{i+1} \vee [a]d_i))$;
$\Box\forall(a \neq \mathit{forward} \supset (\neg d_i \vee [a]d_i))$;
$\Box\forall(a \neq \mathit{forward} \supset ([a]\neg d_i \vee d_i))$;
$\Box\forall(a = \mathit{forward} \supset ([a]\neg d_i \vee d_{i+1}))$;

When considering belief implications, the following result is very useful as it reduces belief implications for proper$^+$ *KB*s to truth in the setup gnd(*KB*).

Theorem 5 $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_k\phi$ *iff* $gnd(KB) \models \boldsymbol{B}_k\phi$.

Proof: For the only-if direction, let $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_k\phi$. It suffices to show that gnd(*KB*) $\models \boldsymbol{B}_0 KB$. It is easy to see that this is the case. For consider $\Box\forall(e \supset c) \in KB$. We need to show that gnd(*KB*), $z, 0 \models (\neg e\theta \vee c\theta)$ for any z and substitution θ. Note that gnd(*KB*), $\langle\,\rangle, 0 \models [z]e\theta$ iff $\models e\theta$. If $\not\models e\theta$, then the claim holds by Rule 14. Otherwise, the subsumption rule applies. (The case of a $\forall$-clause is proved the same way except that $z = \langle\,\rangle$.)

For the if direction, suppose that gnd(*KB*) $\models \boldsymbol{B}_k\phi$. Let $s \models \boldsymbol{B}_0 KB$. Then $\mathsf{VP}(\text{gnd}(KB)) \subseteq \mathsf{VP}(s)$. By Corollary 1, $s \models \boldsymbol{B}_k\phi$. ∎

Corollary 3
$\models \boldsymbol{B}_0 KB \supset [n]\boldsymbol{B}_k\phi$ *iff* $gnd(KB) \cup \{SF(n)\}, n, k \models \phi$ *and* $gnd(KB) \cup \{\neg SF(n)\}, n, k \models \phi$.

Let *KB* be the proper$^+$ BAT of our robot example, including clauses $(d_2 \vee d_3)$, $\neg d_1$, and $\neg d_0$ for the initial situation. As before, suppose the robot is 2 units away from the wall and let $S = SF(f) \wedge [f]SF(s)$. Then

1. $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_0(\neg Close)$;
2. $\not\models S \wedge \boldsymbol{B}_0 KB \supset [f]\boldsymbol{B}_0(d_1 \vee d_2)$;
3. $\models S \wedge \boldsymbol{B}_0 KB \supset [f]\boldsymbol{B}_1(d_1 \vee d_2)$;
4. $\models S \wedge \boldsymbol{B}_0 KB \supset [f][s]\boldsymbol{B}_1 Close$;

The robot believes that it is not close initially at level 0 by subsumption, using the clauses $\neg d_0$ and $\neg d_1$ from gnd(*KB*); after *forward*, it believes that it is at d_1 or d_2 only at level 1, because it needs to split a literal, for example, d_2, and then use unit propagation on the $(\neg d_i \vee [a]d_i)$ for $i = 2$ and $i = 3$; After a *forward* and then sensing the distance to the wall, we can show that the robot believes at level 1 that it is close using the following argument: since the *KB* is augmented by $[f]SF(s)$, $[f](d_0 \vee d_1)$ obtains by unit propagation from $[f](\neg SF(s) \vee d_0 \vee d_1)) \in$ gnd(*KB*). Then splitting $[f]d_0$ and using $[f](\neg d_i \vee [s]d_i)$ the result obtains.

Decidability

We will now turn to the problem of deciding belief implications for proper$^+$ BATs and strictly bounded queries, which are bounded objective formulas not mentioning any action variables. In other words, all actions mentioned in such queries are standard names, as in the examples above.

We will mostly consider belief implications of the form $\boldsymbol{B}_0 KB \supset \boldsymbol{B}_k\phi$. Once we have established decidability for those, the extension to $\boldsymbol{B}_0 KB \supset [z]\boldsymbol{B}_k\phi$-formulas will be straightforward.

The following definition shows how to extract the action sequences mentioned in such queries in a way very similar to PEL (Definition 3).

Definition 6 *Let ϕ be strictly bounded and objective. Then AS(ϕ) is defined inductively as follows:*

$AS(p) = \{\langle\rangle\}$ *when p is an atomic formula;*
$AS((t = t')) = \{\langle\rangle\}$;
$AS(\neg\phi) = AS(\phi)$;
$AS(\phi \vee \psi) = AS(\phi) \cup AS(\psi)$;
$AS(\exists x.\phi) = AS(\phi)$;
$AS([n]\phi) = \{n \cdot z \mid z \in AS(\phi)\}$.

Let $AS_z(\phi) = \{z \cdot z' \mid z' \in AS(\phi)\}$, where z is any action sequence.

The next result shows that, when considering a strictly bounded query ϕ, we only need to consider those instances of $\Box$-formulas in the *KB* which mention action sequences that are subsequences of those in AS(ϕ). We begin by defining such instantiated *KB*s:

Definition 7 *Let KB be proper$^+$ and Z a set of action sequences. The Z-instantiation $KB{\downarrow_Z}$ is then defined as the union of the following sets:*

- $\{[z]\forall(e \supset c) \mid z \sqsubseteq Z, \Box\forall(e \supset c) \in KB, \text{ and } c \text{ static}\}$
- $\{[z]\forall(e \supset c_n^a) \mid z{\cdot}n \sqsubseteq Z, \Box\forall(e \supset c) \in KB, \ c \text{ not static}\}$
- $\{\forall(e \supset c) \mid \forall(e \supset c) \in KB\}$

Theorem 6 *Let KB be a proper$^+$ BAT, ϕ a strictly bounded objective formula, and $Z = AS(\phi)$. Then $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_k\phi$ iff $\models \boldsymbol{B}_0 KB{\downarrow_Z} \supset \boldsymbol{B}_k\phi$.*

The proof is not hard. It involves showing that instances of $\Box$-clauses that are not subsequences of action sequences in AS(ϕ) are irrelevant in establishing beliefs about formulas whose action sequences are included in AS(ϕ). We remark that the analogous theorem holds in $\mathcal{ES}$ as well, that is, $\models \boldsymbol{K}KB \supset \boldsymbol{K}\phi$ iff $\models \boldsymbol{K}KB{\downarrow_Z} \supset \boldsymbol{K}\phi$. It is crucial, though, that *KB* is a proper$^+$ BAT, that is, the $\Box$-clauses in *KB* correspond to the dynamic part of a basic action theories. Otherwise, it is easy to construct a counterexample: for instance, there is a basic action theory *KB* such that $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_0[n_1]F$ and $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_0[n_1][n_2]F$, yet $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_0[n_1][n_2][n_3]\neg F$. Then adding $\Box F$ to *KB* would lead to an inconsistency by unit propagation at level 0, that is, $[] \in \mathsf{UP}(\text{gnd}(KB \cup \{\Box F\}))$, and we obtain that $\boldsymbol{B}_0(KB \cup \{\Box F\}) \supset \boldsymbol{B}_0 p$ is valid for any atomic proposition p. The problem is that we would need to instantiate $KB \cup \{\Box F\}$ using all subsequences of $n_1 \cdot n_2 \cdot n_3$ to discover this, yet AS(p) = {}.

Given Theorem 6, we can restrict our attention to AS(ϕ)-instantiated proper$^+$*KB*s for any given query ϕ. In order to arrive at a decision procedure for belief implications we need two more things: restrict the set of literals to choose from when splitting and, most importantly, restrict the possible substitutions of standard names to consider to a finite set. To achieve the former we let

$$\text{PEL}(KB, \phi, Z) = \{[z]p \mid p \text{ is an atom in } KB \text{ or } \phi, z \sqsubseteq Z\}.$$

The idea is that only literals that mention atoms from the *KB* or the query together with their respective action sequences need to be considered. To limit substitutions of standard names, we follow (Liu et al. 2004) and define a sufficiently large but finite set of standard names for a given *KB* and query ϕ:

For any $m \geq 0$ let H_m^+ be the set of standard names mentioned in *KB* and ϕ plus m new standard names of type object. Let n be the maximum number of (object) variables appearing in a $\forall$-clause appearing in *KB*. Then $\text{gnd}(KB)|_{H_n^+}$ denotes gnd(*KB*) with substitutions θ restricted to names in H_n^+. Similarly, $\text{PEL}(KB, \phi, Z)|_{H_n^+}$ denotes the set of extended literals in $\text{PEL}(KB, \phi, Z)$ with all variables replaced by names in H_n^+.

With these abbreviations in hand, we now define a computable function V, which, as will be shown below, returns 1 just in case $\text{gnd}(KB), z, k \models \phi$, that is, V decides belief implications.

Definition 8 *Let ϕ be a strictly bounded objective sentence, z an action sequence, KB an AS(ϕ)$_z$-instantiated augmented BAT, and $k \geq 0$.*

$$V[KB, \phi, z, k] = \begin{cases} 1 & \text{if any of the following} \\ & \text{conditions (1)–(12) holds} \\ 0 & \text{otherwise} \end{cases}$$

1. $[] \in \mathsf{UP}(gnd(KB)|H_n^+)$;
2. $k = 0$, *ϕ is a clause c, and there exists a* $c' \in \mathsf{UP}(gnd(KB)|H_n^+)$ *such that* $c' \subseteq [z]c$;
3. $k > 0$ *and*
   for some extended literal $l \in PEL(KB, \phi, AS(\phi)_z)|_{H_n^+}$,
   $V[KB \cup \{l\}, \phi, z, k-1] = 1$ *and* $V[KB \cup \{\bar{l}\}, \phi, z, k-1] = 1$

4. $\phi = (n = m)$, *and* n, m *are the same standard names;*
5. $\phi = \neg(n = m)$, *and* n, m *are distinct standard names;*
6. $\phi = \neg\neg\psi$, *and* $V[KB, \psi, z, k] = 1$*;*
7. $\phi = (\psi \vee \chi)$, $V[KB, \psi, z, k] = 1$ *or* $V[KB, \chi, z, k] = 1$*;*
8. $\phi = \neg(\psi \vee \chi)$, $V[KB, \neg\psi, z, k] = 1$ *and* $V[KB, \neg\chi, z, k] = 1$*;*
9. $\phi = \exists x.\psi$, *and* $V[KB, \psi^x_n, z, k] = 1$ *for some* $n \in H^+_1$*;*
10. $\phi = \neg\exists x.\psi$, *and* $V[KB, \neg\psi^x_n, z, k] = 1$ *for all* $n \in H^+_1$*;*
11. $\phi = [n]\psi$, $V[KB \cup \{[z]SF(n)\}, \psi, z \cdot n, k] = 1$ *and* $V[KB \cup \{[z]\neg SF(n)\}, \psi, z \cdot n, k] = 1$*;*
12. $\phi = \neg[n]\psi$, $V[KB \cup \{[z]SF(n)\}, \neg\psi, z \cdot n, k] = 1$ *and* $V[KB \cup \{[z]\neg SF(n)\}, \neg\psi, z \cdot n, k] = 1$.

We will now show that the above restrictions, that is, considering only a finite number of substitution instances of the *KB* and the query, and only finitely many split literals, are indeed sufficient to correctly compute belief implications. As in (Liu et al. 2004), the main idea is that names not mentioned in the *KB* or the query are, in a sense, indistinguishable and hence interchangeable. The following definition of a bijection over names makes this precise:

Let * be a bijection over the set of names (preserving sorts). For any formula α, let α^* be α with every name n replaced by n^*. We extend this notion to sets of formulas, including setups, and substitutions θ in the obvious way. The following properties of * are easy to prove:

Lemma 3

- $\models [z]e$ *iff* $\models [z^*]e^*$ *iff* $\models e^*$*;*
- $c \in \mathsf{UP}(s)$ *iff* $c^* \in \mathsf{UP}(s^*)$*;*
- $s, z, k \models \phi$ *iff* $s^*, z^*, k \models \phi^*$*;*
- *gnd*$(KB)^* =$ *gnd*(KB^*).

Lemma 4 *Let* $m_1, \ldots, m_n$ *be the names in* H^+_n *that do not occur in KB and* ϕ*. Let* $c \in \mathsf{UP}($*gnd*$(KB))$ *contain names* $m'_1, \ldots, m'_l$ $(l \leq n)$ *not mentioned in* H^+_n*. Let* * *be the bijection which swaps* m_i *and* m'_i *for all* $1 \leq i \leq l$ *and is the identity otherwise. Then* $c^* \in \mathsf{UP}($*gnd*$(KB)|_{H^+_n})$.

The proof essentially follows the same argument as Lemma 4 in (Liu et al. 2004). Note that any $c \in \mathsf{UP}(\text{gnd}(KB))$ mentions at most n names not in H^+_n. Also note that the lemma covers the special case that, if $[] \in \mathsf{UP}(\text{gnd}(KB))$ then $[] \in \mathsf{UP}(\text{gnd}(KB)|_{H^+_n})$.

Lemma 5 *Let* ϕ *be a strictly bounded objective formula with a single object variable* x*. Let* n, m *be object names not occurring in KB or* ϕ*.*
Then *gnd*$(KB), z, k \models \phi^x_n$ *iff* *gnd*$(KB), z, k \models \phi^x_m$.

The lemma is easily proved using Lemma 3. Note that this justifies that restricting substitutions of names for quantified variables (Rule 9+10) to elements of H^+_1 is correct.

Lemma 6
Suppose that *gnd*$(KB), z, k \models \phi$ *by splitting an extended primitive literal* l*. Then* *gnd*$(KB), z, k \models \phi$ *by splitting an extended primitive literal* l *in PEL*$(KB, \phi, AS(\phi)_z)|_{H^+_n}$.

Proof: Clearly, only elements of $\text{PEL}(KB, \phi, \text{AS}(\phi)_z)$ are relevant when choosing a literal for splitting. The fact that only substitution instances from H^+_n are needed can again be shown using Lemma 3. ∎

Finally, we are able to show that the beliefs of a proper knowledge base at any level can be decided by V:

Theorem 7 *gnd*$(KB), z, k \models \phi$ *iff* $V[KB, \phi, z, k] = 1$.

Proof: Given Lemma 4–6, the proof is a simple induction on k and the structure of ϕ. ∎

Given Theorem 5, we immediately obtain:

Corollary 4 *The validity problem for sentences of the form* $\boldsymbol{B}_0 KB \supset \boldsymbol{B}_k\phi$*, where KB is a proper*$^+$ *BAT,* ϕ *a strictly bounded objective sentence, and* $k \geq 0$*, is decidable.*

The result can easily be extended to the case of beliefs after a sequence of actions. For simplicity, consider a single action n. Then $\boldsymbol{B}_0 KB \supset [n]\boldsymbol{B}_k\phi$ iff $\text{gnd}(KB) \cup \{SF(n)\}, n, k \models \phi$ and $\text{gnd}(KB) \cup \{\neg SF(n)\}, n, k \models \phi$ (Corollary 3). In both cases, we obtain decidability because Theorem 7 applies.

Corollary 5 *The validity problem for sentences of the form* $\boldsymbol{B}_0 KB \supset [z]\boldsymbol{B}_k\phi$ *is decidable.*

Given the result on eventual completeness in the propositional case (Theorem 3), it is not hard to obtain a similar result for belief implications when the query is propositional.

Theorem 8 *Let KB be a proper*$^+$ *BAT and* ϕ *a propositional strictly bounded objective sentence. Then* $\models KB \supset \phi$ *iff* $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_k\phi$ *for some* k.

Proof: (Sketch) Note that $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_k\phi$ iff $\text{gnd}(KB{\downarrow_Z})|_{H^+_n} \models \boldsymbol{B}_k\phi$ for $Z = \text{AS}(\phi)$ (by Theorem 6,7). Moreover, $\text{gnd}(KB{\downarrow_Z})|_{H^+_n}$ is finite. By an argument very similar to the proof of Theorem 3 one can show that, provided $\models KB \supset \phi$, $\text{gnd}(KB \downarrow_Z)|_{H^+_n} \models \boldsymbol{B}_k\phi$ holds where k is the size of $\text{PEL}(\text{gnd}(KB{\downarrow_Z})|_{H^+_n}, \phi, Z)$. ∎
We can even generalize the theorem to universal queries of the form $\forall\vec{x}.\phi$ where ϕ mentions no quantifiers. Here we only consider the case of a single universal quantifier:

Corollary 6
$\models KB \supset \forall x.\phi$ *iff* $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_k \forall x.\phi$ *for some* k.

Proof: $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_k \forall x\phi$ iff $\models \boldsymbol{B}_0 KB \supset \forall x.\boldsymbol{B}_k\phi$ iff $\models \boldsymbol{B}_0 KB \supset \boldsymbol{B}_k\phi^x_n$ for all n mentioned in *KB* and ϕ plus one new name (using Lemma 5). Now apply the theorem for each such n and choose k to be the largest among all of them. ∎

Conclusions

In this paper we extended ideas of limited belief, which were first explored by Liu et al. (2004) for static knowledge bases, to the dynamic case. By modifying the split rule considered by Liu et al. we were able to substantially extend results on eventual completeness. Moreover, for proper$^+$ knowledge bases which encode Reiter-style basic action theories and strictly bounded queries we showed decidability.

In (Lakemeyer and Levesque 2013), we considered decidable reasoning with nested beliefs and unknown individuals, but only in the static case. We believe that these results carry over to the dynamic setting without much difficulty.

Furthermore, it is not hard to generalize our results to bounded queries that may contain action variables. For example, we could then ask whether $\exists a_1 \exists a_2 [a_1][a_2]\phi$ is believed at some level k. Hence, with bounded queries we are able to express a form of bounded planning problems. Allowing $\Box$ within a query is much more problematic, as $\Box\phi$ expresses a state constraint about ϕ holding after any sequence of actions. Handling $\Box$ as part of a query is problematic for reasons similar to those that make it essential that proper$^+$ *KB*s are restricted to those representing basic action theories. In fact, we conjecture that answering queries involving $\Box$ is undecidable even for limited belief.

Finally, a possible extension we would like to consider in the future is to allow for action functions like *pickup*(x). These are clearly desirable when constructing a basic action theory for a given domain. The main question will be how to achieve this without losing decidability.

References

F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter, Integrating Description Logics and Action Formalisms: First Results. *The Twentieth National Conference on Artificial Intelligence (AAAI)*, 572–577, 2005.

M. Cadoli and M. Schaerf, Approximate Reasoning and Non-Omniscient Agents. In *Proc. of the 4th Conference on Theoretical Aspects of Reasoning about Knowledge (TARK-92)*, pages 159–183, 1992.

T. Capes, *A Logic for the Progression of Knowledge Bases as a Consequence of Unit Propagation* M.Sc. Thesis, University of Toronto, 2010.

B. Chellas, *Modal Logic: An Introduction*. Cambridge University Press, Cambridge, 1980.

J. Claßen and and G. Lakemeyer, Tractable First-Order Golog with Disjunctive Knowledge Bases, Proceedings of the Ninth International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense 2009), Toronto, Canada, UTSePress, pp. 27–33, 2009.

G. De Giacomo, Y. Lesprance, and F. Patrizi. Bounded Situation Calculus Action Theories and Decidable Verification, KR 2012.

G. De Giacomo, Y. Lesprance, and F. Patrizi. Bounded Epistemic Situation Calculus Theories. IJCAI 2013.

J.P. Delgrande, A framework for logics of explicit belief. *Computational Intelligence*, 11(1):47–88, 1995.

R. Fagin and J.Y. Halpern, Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39–76, 1988.

R. Fagin, J.Y. Halpern, and M.Y. Vardi, A nonstandard approach to the logical omniscience problem. In *Proc. of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge (TARK-90)*, pages 41–55, 1990.

M. Gelfond and V. Lifschitz, Representing Actions and Change by Logic Programs. *Journal of Logic Programming,* 17(2–4), 301–323, 1993.

Y. Gu and M. Soutchanski, Decidable Reasoning in a Modified Situation Calculus. *In Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI),* 1891–1897, 2007.

K. Konolige, *A Deduction Model of Belief*. Brown University Press, 1986.

S. Kripke, Is there a problem with substitutional quantification? In G. Evans and J. McDowell, editors, *Truth and Meaning*, pages 325–419. Clarendon Press, Oxford, 1976.

G. Lakemeyer, *Limited Reasoning in First-Order Knowledge Bases with Full Introspection Artif. Intell.* 84(1-2): 209–255, 1996.

G. Lakemeyer, H.J. Levesque, A Semantic Characterization of a Useful Fragment of the Situation Calculus with Knowledge. *Artificial Intelligence,* 175, pages 142–164, 2011.

G. Lakemeyer, H.J. Levesque, Decidable Reasoning in a Logic of Limited Belief with Introspection and Unknown Individuals. *23rd International Joint Conference on Artificial Intelligence (IJCAI),* 2013.

H.J. Levesque, A logic of implicit and explicit belief. In *Proc. of AAAI-84*, 198–202, 1984.

H.J. Levesque, A completeness result for reasoning with incomplete first-order knowledge bases. In *Proc. KR-98*, pages 14–23, 1998.

H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl, Golog: A logic programming language for dynamic domains. *Journal of Logic Programming,* **31**, 59–84, 1997.

Y. Liu, G. Lakemeyer, H.J. Levesque, A Logic of Limited Belief for Reasoning with Disjunctive Information. *Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR),* 587–597, 2004.

Y. Liu and H.J. Levesque, Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In Proc. of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05),522–527, 2005.

J. McCarthy, *Situations, Actions and Causal Laws.* Technical Report, Stanford University, 1963. Also in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, 410–417, 1968.

R. Reiter, *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press, 2001.

R. B. Scherl and H. J. Levesque, Knowledge, action, and the frame problem. *Artificial Intelligence* 144(1-2), 1-39, 2003.

M.Y. Vardi, On epistemic logic and logical omniscience. In *Proc. of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge (TARK-86)*, pages 293–305, 1986.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

A First-Order Semantics for Golog and ConGolog under a Second-Order Induction Axiom for Situations

Fangzhen Lin
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

Golog and ConGolog are languages defined in the situation calculus for cognitive robotics. Given a Golog program δ, its semantics is defined by a macro $Do(\delta, s, s')$ that expands to a logical sentence that captures the conditions under which performing δ in s can terminate in s'. A similar macro is defined for ConGolog programs. In general, the logical sentences that these macros expand to are second-order, and in the case of ConGolog, may involve quantification over programs. In this paper, we show that by making use of the foundational axioms in the situation calculus, in particular, the second-order closure axiom about the space of situations, these macro expressions can actually be defined using first-order sentences.

Introduction

Golog (Levesque et al. 1997) and ConGolog (De Giacomo, Lespérance, and Levesque 2000) are Algol-like programming languages defined in the situation calculus (McCarthy 1968; Reiter 2001). They are intended for high-level control of agents, including both physical robots and virtual agents (Reiter 2001; Burgard et al. 1999; Funge 2000; Lespérance, Levesque, and Ruman 1997; McIlraith and Son 2002). They include non-determinism, and in the case of ConGolog, concurrency.

The semantics of Golog programs is axiomatized in logic by macro expansions (Levesque et al. 1997): given a Golog program δ and two situation terms s and s', the macro expression $Do(\delta, s, s')$ is used to denote that s' is a terminating situation of performing δ in s. There may be more than one possible terminating situation as the program may be indeterminate. In general, the macro expression $Do(\delta, s, s')$ expands into a second-order sentence because δ may have loops. In the case of ConGolog programs, which are extensions of Golog programs with some concurrency operators, De Giacomo, Lespérance and Levesque (2000) introduced a one-step transition predicate $Trans(\delta, s, \delta', s')$ (performing δ in s for one step may end up in situation s' with δ' as the remaining program to be performed) and a final predicate $Final(\delta, s)$ (program δ terminates in s). The Do macros are then defined using these predicates, and in general expand to sentences that are not only second-order, but also involve quantification over programs.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The second-order formulas that are needed in Golog and ConGolog macro expansions are fundamentally the same as the ones for defining transitive closures. One of the foundational axioms of the situation calculus is a second-order one that defines the space of situations as the transitive closure of $do(a, s)$ starting at the initial situation S_0. In this paper, we show that this second-order foundational axiom can be used to capture loops and recursions so that for Golog and ConGolog programs, the macro expression $Do(\delta, s, s')$ can in fact be defined using first-order sentences.

The situation calculus

We first briefly review the situation calculus used for Golog and ConGolog. For more details, see (Reiter 2001; Lin 2007). The language of the situation calculus is a many-sorted one with sorts $action$ for actions, $situation$ for situations, and other domain dependent ones. There is a constant S_0 of sort $situation$, and a function $do : action \times situation \rightarrow situation$.

The foundational axioms say that the set of situations consists exactly of those that can be constructed from S_0 using function $do(a, s)$ (we assume that variables in displayed formulas are universally quantified from outside):

$$S_0 \neq do(a, s), \tag{1}$$

$$do(a_1, s_1) = do(a_2, s_2) \supset a_1 = a_2 \wedge s_1 = s_2, \tag{2}$$

$$\forall P.[P(S_0) \wedge \forall a, s(P(s) \supset P(do(a, s)))] \supset \forall s P(s), \tag{3}$$

The first two axioms are unique names axioms, and the third one a second-order induction axiom. Given a sequence $[a_1, ..., a_n]$ of actions, we use $do([a_1, ..., a_n], s)$ to denote the situation resulting from doing the sequence of actions in s: $do(a_n, ..., do(a_1, s)...)$. Using this notation, we can understand the foundational axioms as saying that s is a situation iff there is a sequence α of actions such that $s = do(\alpha, S_0)$.

With these axioms, one can define other relations inductively on situations (Reiter 1993; Lin and Reiter 1994). The following axioms define a partial order $\leq$ on situations:

$$\neg s < S_0,$$
$$s < do(a, s') \equiv s = s' \vee s < s',$$
$$s \leq s' \equiv s < s' \vee s = s'.$$

Again in terms of sequences of actions, we can see that $s \leq s'$ iff there is a sequence α of actions such that $s' = do(\alpha, s)$.

In the following, we denote by Σ the set of foundational axioms as well as the above axioms for $<$.

Golog

In this section, we briefly review the definitions of Golog programs and their semantics as defined in (Levesque et al. 1997). Golog programs are complex actions and procedures. Complex actions are defined inductively as follows:

$$\begin{array}{rcl}\delta & ::= & A \mid \varphi? \mid \delta;\delta \mid \\ & & (\delta \mid \delta) \mid (\pi x)\delta \mid \delta^*\end{array}$$

where A is a primitive action, φ is a pseudo-situation calculus formula with all situation arguments suppressed. In (Levesque et al. 1997), the semantics of a complex action δ is defined by a macro $Do(\delta, s, s')$ that expands to a situation calculus formula that says that s' is a terminating situation of performing δ in s. Formally, this macro expansion is defined inductively as follows:

$$\begin{array}{l}Do(A, s, s') \stackrel{def}{=} Poss(A, s) \wedge s' = do(A, s),\\ Do(\varphi?, s, s') \stackrel{def}{=} s' = s \wedge \varphi[s],\\ Do(\delta_1;\delta_2, s, s') \stackrel{def}{=} \exists s''.Do(\delta_1, s, s'') \wedge Do(\delta_2, s'', s'),\\ Do(\delta_1 \mid \delta_2, s, s') \stackrel{def}{=} Do(\delta_1, s, s') \vee Do(\delta_2, s, s'),\\ Do((\pi x)\delta(x), s, s') \stackrel{def}{=} \exists x.Do(\delta(x), s, s'),\\ Do(\delta^*, s, s') \stackrel{def}{=} \forall P.[\forall s_1 P(s_1, s_1) \wedge\\ \quad \forall s_1, s_2, s_3(Do(\delta, s_1, s_2) \wedge P(s_2, s_3) \supset P(s_1, s_3))]\\ \qquad \supset P(s, s'),\end{array}$$

where A is a primitive action, $Poss(A, s)$ the action precondition predicate, and $\varphi[s]$ the result of restoring s as the situation argument to the fluents in φ.

As we mentioned, the macro $Do(\delta, s, s')$ expands into a formula that is in general second-order. With nested iterations, the formula can get rather complicated. Consider the following complex action $g = (\pi x)(A(x) \mid (\pi x)B(x)^*)^*$. Let $g_1 = (A(x) \mid (\pi x)B(x)^*)^*$, $g_2 = A(x) \mid (\pi x)B(x)^*$, $g_3 = (\pi x)B(x)^*$, and $g_4 = B(x)^*$. Then

$$\begin{array}{l}Do(g, s, s') \stackrel{def}{=} \exists x.Do(g_1, s, s'),\\ Do(g_1, s, s') \stackrel{def}{=} \forall P.\{\forall s_1 P(s_1, s_1) \wedge\\ \quad \forall s_1, s_2, s_3(P(s_1, s_2) \wedge Do(g_2, s_2, s_3) \supset P(s_1, s_3))\}\\ \qquad \supset P(s, s'),\\ Do(g_2, s, s') \stackrel{def}{=} Do(A(x), s, s') \vee Do(g_3, s, s'),\\ Do(A(x), s, s') \stackrel{def}{=} Poss(A(x), s) \wedge s' = do(A(x), s),\\ Do(g_3, s, s') \stackrel{def}{=} \exists x.Do(g_4, s, s'),\\ Do(g_4, s, s') \stackrel{def}{=} \forall P.\{\forall s_1 P(s_1, s_1) \wedge\\ \quad \forall s_1, s_2, s_3(P(s_1, s_2) \wedge Do(B(x), s_2, s_3) \supset P(s_1, s_3))\}\\ \qquad \supset P(s, s'),\\ Do(B(x), s, s') \stackrel{def}{=} Poss(B(x), s) \wedge s' = do(B(x), s).\end{array}$$

As one can see, when fully expanded, $Do(g, s, s')$ is a second-order sentence with nested second-order quantification.

Golog Do macro - a first-order definition

We now show that given that we already have a second-order induction axiom for situations, the macros can actually be defined in first-order terms.

Obviously, the question is how to expand $Do(\delta^*, s, s')$. The second-order sentence in the previous section defines it to be the transitive closure of $Do(\delta, s, s')$. If we introduce a new predicate denoting this transitive closure, then it can be defined inductively just like $<$:

$$Do(\delta^*, s, s') \stackrel{def}{=} P_\delta(\vec{x}, s, s'), \quad (4)$$

$$\begin{array}{l}P_\delta(\vec{x}, s, s') \equiv s = s' \vee\\ \quad \exists s''.s < s'' \leq s' \wedge Do(\delta, s, s'') \wedge P_\delta(\vec{x}, s'', s'),\end{array} \quad (5)$$

where P_δ a new predicate, and $\vec{x}$ the tuple of free variables in δ. Notice that $P_\delta(\vec{x}, s, s')$ is recursive on $P_\delta(\vec{x}, s'', s')$ for s'' such that $s < s'' \leq s'$, making it well-defined according to the "distance" between s and s'.

Our proposal is then to replace the macro definition of $Do(\delta^*, s, s')$ in the previous section by (4) and add the axiom (5) whenever the macro is expanded (see Theorem 1 below).

As an example, consider again the complex action g given above. We have

$$\begin{array}{l}Do(g, s, s') \stackrel{def}{=} \exists x.Do(g_1, s, s'),\\ Do(g_1, s, s') \stackrel{def}{=} P_{g_2}(x, s, s'),\\ P_{g_2}(x, s, s') \equiv s = s' \vee\\ \quad \exists s''.s < s'' \leq s' \wedge Do(g_2, s, s'') \wedge P_{g_2}(x, s'', s'),\\ Do(g_2, s, s') \stackrel{def}{=} Do(A(x), s, s') \vee Do(g_3, s, s'),\\ Do(A(x), s, s') \stackrel{def}{=} Poss(A(x), s) \wedge s' = do(A(x), s),\\ Do(g_3, s, s') \stackrel{def}{=} \exists x.Do(g_4, s, s'),\\ Do(g_4, s, s') \stackrel{def}{=} P_{B(x)}(x, s, s'),\\ P_{B(x)}(x, s, s') \equiv s = s' \vee\\ \quad \exists s''.s < s'' \leq s' \wedge Do(B(x), s, s'') \wedge P_{B(x)}(x, s'', s'),\\ Do(B(x), s, s') \stackrel{def}{=} Poss(B(x), s) \wedge s' = do(B(x), s).\end{array}$$

Thus the axioms about the two extra predicates are

$$\begin{array}{l}P_{g_2}(x, s, s') \equiv s = s' \vee\\ \quad \exists s''\{s < s'' \leq s' \wedge P_{g_2}(x, s'', s') \wedge\\ \quad [Poss(A(x), s) \wedge s'' = do(A(x), s)] \vee \exists x.P_{B(x)}(x, s, s'')\},\\ P_{B(x)}(x, s, s') \equiv s = s' \vee\\ \quad \exists s''.s < s'' \leq s' \wedge P_{B(x)}(x, s'', s') \wedge\\ \quad [Poss(B(x), s) \wedge s'' = do(B(x), s)].\end{array}$$

To illustrate how to reason with these axioms, suppose there are two objects a and b, and that $Poss(A(a), S_0)$ and $Poss(B(b), do(A(a), S_0))$ hold. We show that

$$Do(g, S_0, do(B(b), do(A(a), S_0)))$$

holds. It is easy to see that this follows from

$$\exists x.P_{g_2}(x, S_0, S_2),$$

where $S_2 = do(B(b), S_1)$ and $S_1 = do(A(a), S_0)$. This follows from

$$Poss(A(a), S_0) \wedge P_{g_2}(a, S_1, S_2).$$

The first conjunct is assumed to be true. The second conjunct follows from

$$\exists x.P_{B(x)}(x, S_1, S_2).$$

This is proved by showing $P_{B(x)}(b, S_1, S_2)$, which follows from

$$Do(B(b), S_1, S_2),$$

which is true because $Poss(B(b), S_1)$ is assumed to be true.

Now suppose that there is another primitive action $C(x)$. Furthermore, assume the unique names axioms for actions, $C(x) \neq A(y) \neq B(z)$. Then we can show that $Do(g, S_0, do(C(a), S_0))$ does not hold:

$$\neg Do(g, S_0, S_3),$$

where $S_3 = do(C(a), S_0)$, follows from

$$\forall x.\neg P_{g_2}(x, S_0, S_3),$$

which is equivalent to

$$\forall x \forall s.\neg\{S_0 < s \leq S_3 \wedge P_{g_2}(x, s, S_3) \wedge$$
$$[Poss(A(x), S_0) \wedge s = do(A(x), S_0)] \vee \exists x.P_{B(x)}(x, S_0, s)\}$$

which is equivalent to

$$\forall x \forall s.S_0 < s \leq S_3 \supset$$
$$[\neg[Poss(A(x), S_0) \wedge s = do(A(x), S_0)] \wedge$$
$$\neg\exists x.P_{B(x)}(x, S_0, s)]\}.$$

Because S_3 is the only s that can make $S_0 < s \leq S_3$ true, thus the above assertion is equivalent to

$$\forall x.\neg[Poss(A(x), S_0) \wedge S_3 = do(A(x), S_0)] \wedge$$
$$\neg\exists x.P_{B(x)}(x, S_0, S_3).$$

But $S_3 \neq do(A(x), S_0)$ for any x, thus the above assertion is equivalent to

$$\forall x.\neg P_{B(x)}(x, S_0, S_3),$$

which is equivalent to

$$\forall x.\neg[Poss(B(x), S_0) \wedge S_3 = do(B(x), S_0)],$$

which is equivalent to

$$\forall x.S_3 \neq do(B(x), S_0),$$

which is always true as $S_3 = do(C(a), S_0)$.

The following result is not hard to prove.

Theorem 1 *Let $Do1(\delta, s, s')$ be the macro operator defined in (Levesque et al. 1997) (given in the previous section), and $Do2(\delta, s, s')$ the modification to $Do1$ as described above. For any Golog complex action δ, we have that*

$$\Sigma \cup T_\delta \models \forall \vec{x}, s, s'.Do1(\delta, s, s') \equiv Do2(\delta, s, s'),$$

where $\vec{x}$ is the tuple of free variables in δ, and T_δ is the set of axioms about P_{δ_1} for any δ_1 such that δ_1^* occurs in δ.

Notice that for a program like

$$\delta = [A(a)]^*; [(\pi x)A(x)]^*; (\pi x)A(x)^*,$$

T_δ would contain axioms about $P_{A(a)}(s, s')$, $P_{(\pi x)A(x)}(s, s')$, and $P_{A(x)}(x, s, s')$.

Procedures

Golog programs can have recursively defined procedures in the following form (Levesque et al. 1997):

proc $P_1(\vec{v}_1)\delta_1$ **endProc** ; ...; **proc** $P_n(\vec{v}_n)\delta_n$ **endProc** ; δ_0.

Here P_i, $1 \leq i \leq n$, are procedures, and δ_0 is the main program. The procedure bodies and the main program δ_i, $0 \leq i \leq n$, are complex actions, extended by procedure calls. In the main program δ_0, a procedure call $P_i(\vec{t}_i)$ is treated like an action:

$$Do(P_i(\vec{t}_i), s, s') \stackrel{def}{=} P_i(\vec{t}_i[s], s, s'),$$

where $P_i(\vec{v}_i, s, s')$ is a new predicate for the procedure P_i, and $\vec{t}_i[s]$, like $\varphi[s]$, is the tuple of terms obtained by restoring s as the situation arguments in the functional fluents occurring in $\vec{t}_i$. As discussed in (Levesque et al. 1997), this amounts to call by value. These new predicates are defined by second-order sentences in (Levesque et al. 1997) to capture least fixed-point semantics of recursive procedures. Here we axiomatize these predicates using first-order axioms, with the help of some additional new predicates: For each $P_i(\vec{v}_i, s, s')$, we introduce a new one with the same name but with one more situation argument $P_i(\vec{v}_i, s, s', s'')$. Intuitively, the extra argument s'' records the number of times that the procedures have been activated. More precisely, $P_i(\vec{v}_i, s, s', s'')$ holds if in the initial situation s, the procedure P_i can return and yields the new situation s' with at most $|s''|$ number of activations for all procedures, where $|s''|$ is the number of actions in the sequence α such that $s'' = do(\alpha, S_0)$. Thus we have the following axioms:

$$P_i(\vec{v}_i, s, s') \equiv \exists s''.P_i(\vec{v}_i, s, s', s''), \quad (6)$$
$$\neg P_i(\vec{v}_i, s, s', S_0), \quad (7)$$
$$P_i(\vec{v}_i, s, s', do(a, s'')) \equiv Do(\delta_i[s''], s, s'), \quad (8)$$
$$Do(P_i(\vec{t}_i, s''), s, s') \stackrel{def}{=} P_i(\vec{t}_i[s], s, s', s''), \quad (9)$$

where $\delta_i[s'']$ is the result of replacing every procedure call of the form $P_j(\vec{t}_j)$ in δ_i by the call $P_j(\vec{t}_j, s'')$.

Consider the following program in the blocks world from (Levesque et al. 1997) ($maketower(n)$ makes a tower of n blocks; $tower(x, m)$ is true if there is a tower of m blocks whose top one is x; $stack(x, n)$ places n blocks on the tower whose top one is x; $unstack(x, n)$ removes n blocks from the tower whose top is x):

proc $maketower(n)$
  $(\pi x, m)[tower(x, m)?;$
  if $m \leq n$ **then** $stack(x, n - m)$
    else $unstack(x, m - n)$
  endIf]
endProc;

proc $stack(x, n)$
  $n = 0? \mid (\pi y)[put(y, x); stack(y, n - 1)]$
endProc;

proc $unstack(x, n)$

$n = 0? \mid (\pi y)[on(x, y)?; movetotable(x);$
$unstack(y, n-1)]$

endProc;

$maketower(7); \neg(\exists x) on(x, A)?$

We have the following axioms for $maketower(n, s, s')$, $stack(x, n, s, s')$, and $unstack(x, n, s, s')$:

$maketower(n, s, s') \equiv \exists s_1.maketower(n, s, s', s_1),$
$stack(x, n, s, s') \equiv \exists s_1.stack(x, n, s, s', s_1),$
$unstack(x, n, s, s') \equiv \exists s_1.unstack(x, n, s, s', s_1),$
$\neg maketower(n, s, s', S_0),$
$maketower(n, s, s', do(a, s_1)) \equiv \exists x, m.[tower(x, m, s) \wedge$
$m \leq n \supset stack(x, n-m, s, s', s_1) \wedge$
$m > n \supset unstack(x, m-n, s, s', s_1)],$
$\neg stack(x, n, s, s', S_0),$
$stack(x, n, s, s', do(a, s_1)) \equiv (n = 0 \wedge s = s') \vee$
$\exists y, s''.s'' = do(put(y, x), s) \wedge stack(y, n-1, s'', s', s_1),$
$\neg unstack(x, n, s, s', S_0),$
$unstack(x, n, s, s', do(a, s_1)) \equiv (n = 0 \wedge s = s') \vee$
$[\exists y, s''.on(x, y, s) \wedge s'' = do(movetotable(x), s) \wedge$
$unstack(y, n-1, s'', s', s_1)].$

To see how our approach handles cycles, consider the following cyclic procedure:

proc P $\{P\}$ **endProc**
P

Clearly, the call P would never terminate. Our axioms for P are:

$P(s, s') \equiv \exists s_1.P(s, s', s_1),$
$\neg P(s, s', S_0),$
$P(s, s', do(a, s_1)) \equiv P(s, s', s_1)$

By induction using the foundational axioms, it is easy to see that these axioms entail $\forall s, s', s_1.\neg P(s, s', s_1)$, thus $\forall s, s'.\neg P(s, s')$.

Again the following result is not hard to see.

Theorem 2 *Given a Golog program below*

proc $P_1(\vec{v}_1)\delta_1$ **endProc** ; ...; **proc** $P_n(\vec{v}_n)\delta_n$ **endProc** ; δ_0

we have that

$$\Sigma \cup T_{\delta_0} \models \forall \vec{x}, s, s'.Do1(\delta_0, s, s') \equiv Do2(\delta_0, s, s'),$$

where $\vec{x}$ is the tuple of free variables in δ, $Do1(\delta_0, s, s')$ and $Do2(\delta_0, s, s')$ the macro operators defined in (Levesque et al. 1997) and this paper, respectively, and T_{δ_0} the set of axioms about the new predicates introduced in defining $Do2$, including those for P_δ for each δ such that δ^* occurs in some δ_i, $0 \leq i \leq n$, and (6) - (9) for each P_i.

Concurrent processes

ConGolog (De Giacomo, Lespérance, and Levesque 2000) extends Golog by adding some concurrency operators that allows two programs to be executed concurrently. Instead of the Do macro, the main construct is a transition predicate $Trans(\delta, s, \delta', s')$ which says one way to do δ in s is to perform a primitive or a testing action that will lead to situation s' with δ' as the remaining program to be executed. For example, $Trans(a; b, s, b, do(a, s))$ holds because performing the sequence $a; b$ in s will result in performing a, which leads to $do(a, s)$ with b as the remaining program to do.

In (De Giacomo, Lespérance, and Levesque 2000), the axiomatization of $Trans$ requires programs be reified and quantified over. For instance, the axiom for sequence is

$$Trans(\delta_1; \delta_2, s, \delta', s') \equiv$$
$$\exists \gamma.\delta' = (\gamma; \delta_2) \wedge Trans(\delta_1, s, \gamma, s') \vee$$
$$Final(\delta_1, s) \wedge Trans(\delta_2, s, \delta', s'),$$

where $Final(\delta, s)$ is a predicate which says that δ has finished in s.

Using $Trans$, and by quantifying over programs, Do is then defined as

$$Do(\delta, s, s') \stackrel{def}{=} \exists \delta'.Trans^*(\delta, s, \delta', s') \wedge Final(\delta', s'),$$

where $Trans^*$ is the transitive closure of $Trans$.

Here we show that even for concurrent processes, the Do macro can be defined using first-order axioms only, and without having to reify and quantify over programs.

Consider concurrent execution by interleaving $\delta_1 \parallel \delta_2$. Executing this program in s can result in s' if there is a sequence of primitive actions, some from δ_1 and the rest from δ_2, that can be executed in s to yield s'. For example, there are six possible ways to interleave $(P?; A)$ and $(B; A)$, resulting in the following four possible sequences of primitive and test actions:

$$(P?; A; B; A), (P?; B; A; A), (B; A; P?; A), (B; P?; A; A).$$

If we ignore the test actions, as they do not result in changes of situations, we then get two possible sequences of primitive actions: $[A, B, A]$ and $[B, A, A]$. Notice that these are *possible* interleavings. Not all of them may come to fruit. For example, $P?$ may not be true initially, thus cannot be performed as the first action.

If we are told that $[A, B, A]$ is the actual sequence of actions performed, then we know that the first action A must come from the process $(P?; A)$, and the last from $(B; A)$. However, if $[B, A, A]$ is the actual sequence, then for the two A's in the sequence, we do not know from which process each of them come.

To keep track of which action comes from which process in a sequence of primitive actions, we introduce two special primitive actions Y (for yes), and N (for no). Situations consisting of only these two actions are called "schedules':

$sch(S_0),$
$sch(do(a, s)) \equiv sch(s) \wedge (a = Y \vee a = N).$

For example, the schedule $do(Y, do(N, do(Y, do(Y, S_0))))$ $(do([Y, Y, N, Y], S_0))$ tells a process to first execute two actions, then wait for the other process to execute one action,

then execute one more action. Referring back to our earlier example, if we know that $[B, A, A]$ is the actual sequence, then for process $(P?; A)$, there can be only two possible schedules: $do([N, Y, N], S_0)$ or $do([N, N, Y], S_0)$. The first implies that the first A in the sequence $[B; A; A]$ comes from the process, and the second that the second A in the sequence comes from it. Notice that our "schedule situations" only tell when to execute primitive actions while ignore test actions. For process $(P?; A)$ to be successfully executed, we know that P must be tested and succeeded sometime before A is executed, but we do not care exactly when it was actually tested.

These special schedule situations will be used to model possible interleavings. Before we proceed, we first introduce some useful operators about situations:

- We say that two situations are of equal length, written $el(s, s')$, if the number of actions in s is the same as that in s':

$$el(S_0, S_0),$$
$$\neg el(do(a, s), S_0),$$
$$el(do(a, s), do(b, s')) \equiv el(s, s').$$

- We say one schedule s_1 is earlier than s_2, written $earlier(s_1, s_2)$, if at least one action is scheduled earlier in s_1:

$$earlier(s_1, s_2) \equiv \exists s_3.do(Y, s_3) \leq s_1 \wedge do(N, s_3) \leq s_2.$$

- A schedule can be split into two: $split(s, s_1, s_2)$ holds if for any action, it is scheduled to be performed in s iff it is scheduled to be performed either in s_1 or s_2, but not both:

$$split(S_0, S_0, S_0),$$
$$split(do(N, s), s_1, s_2) \equiv \exists s_3, s_4.split(s, s_3, s_4) \wedge$$
$$s_1 = do(N, s_3) \wedge s_2 = do(N, s_4),$$
$$split(do(Y, s), s_1, s_2) \equiv \exists s_3, s_4.split(s, s_3, s_4) \wedge$$
$$[(s_1 = do(Y, s_3) \wedge s_2 = do(N, s_4)) \vee$$
$$(s_1 = do(N, s_3) \wedge s_2 = do(Y, s_4))].$$

  It is easy to see that splitting preserves the length of the schedule:

$$sch(s) \wedge split(s, s_1, s_2) \supset el(s, s_1) \wedge el(s, s_2).$$

- We have two special schedules: $Full(s, s')$ if the schedule from s to s' is full (all "Y"), and $Empty(s, s')$ if the schedule from s to s' is empty (all "N"):

$$Full(s, s') \equiv \forall a, s_1(s < do(a, s_1) \leq s' \supset a = Y),$$
$$Empty(s, s') \equiv \forall a, s_1(s < do(a, s_1) \leq s' \supset a = N).$$

  Notice that $Full(s, s)$ and $Empty(s, s)$ are always true.

ConGolog has three main concurrency operators: $\delta_1 \parallel \delta_2$ (execute δ_1 and δ_2 concurrently by interleaving), $\delta_1 \gg \delta_2$ (interleaving δ_1 and δ_2 with priority to δ_1), and $\delta^{\parallel}$ (interleave δ any number of times).

The prioritized concurrency $\delta_1 \gg \delta_2$ needs special attention. Consider

$$((A; fail?) \mid B) \gg C.$$

Suppose that initially both A and C are executable, but B is not. Suppose further that if C is executed, then B becomes executable. According to the semantics in (De Giacomo, Lespérance, and Levesque 2000), the above program cannot be executed as $((A; fail?) \mid B)$ has priority, so its first action, which is A, is executed. But once A is executed, it is at a dead end. In this view, if a process has priority, then its first executable action is performed, regardless whether executing that action can lead to a successful execution of this process. However, if we insist on only executing actions that can lead to termination, then the process with lower priority, C, would be executed first as the process with higher priority cannot be executed. After C is performed, B from the other process can then be executed.

Which interpretation is more appropriate depends on the applications. We'll axiomatize both. In our framework, it is easier to axiomatize the latter, so we do it first.

Using schedule situations, we extend the Do macro to concurrency as follows: $Do(\delta, s, s')$ if with a full schedule, executing δ in s can lead to s':

$$Do(\delta, s, s') \stackrel{def}{=} \exists s_1.Full(S_0, s_1) \wedge el(s', s_1) \wedge Do(\delta, s, s', s_1), \tag{10}$$

where $Do(\delta, s, s', s_1)$ is a new macro meaning that executing the sequence of actions from s to s' according to schedule s_1 is a valid way to execute δ in s:

$$Do(A, s, s', s_1) \stackrel{def}{=} \exists s_2, s_3, s''.s < do(A, s'') \leq s' \wedge$$
$$s_2 < do(Y, s_3) \leq s_1 \wedge el(s, s_2) \wedge el(s'', s_3) \wedge$$
$$Empty(s_2, s_3) \wedge Empty(do(Y, s_3), s_1) \wedge Poss(A, s''),$$

$$Do(\varphi?, s, s', s_1) \stackrel{def}{=} \exists s_2, s''.s \leq s'' \leq s' \wedge$$
$$s_2 \leq s_1 \wedge el(s, s_2) \wedge Empty(s_2, s_1) \wedge \varphi[s''],$$

$$Do(\delta_1; \delta_2, s, s', s_1) \stackrel{def}{=} \exists s''.s_2.s_2 \leq s_1 \wedge el(s_2, s'') \wedge$$
$$Do(\delta_1, s, s'', s_2) \wedge Do(\delta_2, s'', s', s_1),$$

$$Do(\delta_1 \mid \delta_2, s, s', s_1) \stackrel{def}{=}$$
$$Do(\delta_1, s, s', s_1) \vee Do(\delta_2, s, s', s_1),$$

$$Do((\pi x)\delta(x), s, s', s_1) \stackrel{def}{=} \exists x.Do(\delta(x), s, s', s_1),$$

$$Do(\delta(\vec{x})^*, s, s', s_1) \stackrel{def}{=} P_\delta(\vec{x}, s, s', s_1),$$
$$P_\delta(\vec{x}, s, s', s_1) \equiv s = s' \vee$$
$$\exists s''.s_2.s < s'' \leq s' \wedge s_2 \leq s_1 \wedge el(s_2, s'') \wedge$$
$$Do(\delta, s, s'', s_2) \wedge P_\delta(\vec{x}, s'', s', s_1),$$

$$Do(\delta_1 \parallel \delta_2, s, s', s_1) \stackrel{def}{=} \exists s_2, s_3.split(s_1, s_2, s_3) \wedge$$
$$Do(\delta_1, s, s', s_2) \wedge Do(\delta_2, s, s', s_3),$$

$$Do(\delta_1 \gg \delta_2, s, s', s_1) \stackrel{def}{=} \exists s_2, s_3\{split(s_1, s_2, s_3) \wedge$$
$$Do(\delta_1, s, s', s_2) \wedge Do(\delta_2, s, s', s_3) \wedge$$
$$\neg\exists s_4, s_5, s_6, s''[split(s_1, s_4, s_5) \wedge earlier(s_4, s_2) \wedge$$
$$s_6 < s_4 \wedge s_6 < s_2 \wedge el(s_6, s) \wedge Do(\delta_1, s, s'', s_4)]\},$$

$$Do(\delta(\vec{x})^{\parallel}, s, s', s_1) \stackrel{def}{=} s = s' \vee P_\delta^{\parallel}(\vec{x}, s, s', s_1),$$
$$\exists s_2(s_2 \leq s_1 \wedge el(s, s_2) \wedge Empty(s_2, s_1)) \supset$$
$$P_\delta^{\parallel}(\vec{x}, s, s', s_1),$$

$$\neg\exists s_2(s_2 \leq s_1 \wedge el(s, s_2) \wedge Empty(s_2, s_1)) \supset$$
$$[P_\delta^{\|}(\vec{x}, s, s', s_1) \equiv$$
$$\exists s_2, s_3.split(s_1, s_2, s_3) \wedge s_1 \neq s_2 \wedge P_\delta^{\|}(\vec{x}, s, s', s_2) \wedge$$
$$Do(\delta(\vec{x}), s, s', s_3)].$$

Notice that for $Do(\delta, s, s', s_1)$ to make sense, the length of the terminating situation s' should be the same as the length of the schedule s_1, and only the segment of the schedule from s to s' is relevant. Thus, for example,

$$Do(\delta, do(a, S_0), do(b, do(a, S_0)), do(Y, do(N, S_0)))$$

holds iff

$$Do(\delta, do(a, S_0), do(b, do(a, S_0)), do(Y, do(Y, S_0)))$$

holds as the two schedules are the same as far as the last action is concerned. We explain the first two definitions in more details below:

- For a primitive action A, $Do(A, s, s', s_1)$ holds if the schedule s_1 has exactly one occurrence of "Y", and at that point, the corresponding action in s' is A. Reformatted in terms of sequences of actions, the definition says that $Do(A, s, s', s_1)$ holds if for some sequences of actions α, γ, β_i, and γ_i, $i = 1, 2$, we have:
$$s = do(\alpha, S_0),$$
$$s' = do(\beta_1, do(A, do(\beta_2, s))),$$
$$s_1 = do(\gamma_1, do(Y, do(\gamma_2, do(\gamma, S_0)))),$$
$$|\alpha| = |\gamma|,$$
$$|\beta_i| = |\gamma_i|, i = 1, 2,$$
$$a \in \gamma_i \supset a = N, i = 1, 2.$$

- For a test action, $Do(\varphi?, s, s', s_1)$ holds if the schedule s_1 is empty, and φ is true somewhere between s and s'. Again, it is easier to understand in terms of sequences of actions. The definition says that $Do(\varphi, s, s', s_1)$ holds if for some sequences of actions α, β, and γ, we have:
$$\exists s''.s \leq s'' \leq s' \wedge \varphi[s''],$$
$$s = do(\alpha, S_0),$$
$$s_1 = do(\gamma, do(\beta, S_0)),$$
$$|\alpha| = |\beta|,$$
$$a \in \gamma \supset a = N.$$

As an example, consider again the following prioritized concurrent program:

$$\delta = ((A; fail?) \mid B) \gg C,$$

where A, B, and C are primitive actions, and $fail$ is a fluent that is never true. Suppose that $Poss(A, S_0)$, $\neg Poss(B, S_0)$, $Poss(C, S_0)$, and $Poss(B, do(C, S_0))$. We show that

$$Do(\delta, S_0, do(B, do(C, S_0))).$$

We need to prove

$$Do(\delta, S_0, do(B, do(C, S_0)), do(Y, do(Y, S_0))).$$

Let $\delta_1 = (A; fail?) \mid B$. The above assertion follows from the following three assertions:

$$Do(\delta_1, S_0, do(B, do(C, S_0)), do(Y, do(N, S_0))), \quad (11)$$
$$Do(C, S_0, do(B, do(C, S_0)), do(N, do(Y, S_0))), \quad (12)$$
$$\neg\exists s_1, s_2, s.split(do(Y, do(Y, S_0)), s_1, s_2) \wedge$$
$$earlier(s_1, do(Y, do(N, S_0))) \wedge Do(\delta_1, S_0, s, s_1). \quad (13)$$

The first assertion (11) follows from

$$Do(B, S_0, do(B, do(C, S_0)), do(Y, do(N, S_0))),$$

which follows from the following facts:

$$el(S_0, S_0),$$
$$el(do(C, S_0), do(N, S_0)),$$
$$Empty(S_0, do(N, S_0)),$$
$$Empty(do(Y, do(N, S_0)), do(Y, do(N, S_0))),$$
$$Poss(B, do(C, S_0)).$$

The second assertion (12) follows easily from the following facts:

$$el(S_0, S_0),$$
$$Empty(S_0, S_0),$$
$$Empty(do(Y, S_0), do(N, do(Y, S_0))),$$
$$Poss(C, S_0).$$

For the last assertion (13), notice first that for

$$split(do(Y, do(Y, S_0)), s_1, s_2) \wedge earlier(s_1, do(Y, do(N, S_0)))$$

to be true, s_1 must be either $do(Y, do(Y, S_0))$ or $do(N, do(Y, S_0))$. But in either case, there is no s such that $Do(\delta_1, S_0, s, s_1)$ can be true.

Notice that under ConGolog's semantics (De Giacomo, Lespérance, and Levesque 2000), there is no s such that $Do(\delta, S_0, s)$ can be true. This shows that our treatment of the prioritized concurrency operator $\gg$ is different from the one in ConGolog (De Giacomo, Lespérance, and Levesque 2000). If we ignore operator $\gg$, then our semantics is equivalent to the one in (De Giacomo, Lespérance, and Levesque 2000). Before we prove this, we first prove some formal properties about our axiomatization.

We mentioned that in $Do(\delta, s, s', s_1)$, only the part of the schedule between s and s' matters. We now make this precise.

We say that two situations s_1 and s_2 are the same between s and s', written $eq(s_1, s_2, s, s')$, if there are sequences $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ of actions such that

- $s' = do(\alpha_1, s)$,
- $s_1 = do(\alpha_3, do(\alpha_2, s_3))$ and $s_2 = do(\alpha_4, do(\alpha_2, s_4))$,
- $|\alpha_1| = |\alpha_2|$,
- s_3, s_4, and s are of equal length.

In terms of situation calculus formulas, we have

$$eq(s_1, s_2, s, s') \stackrel{def}{=}$$
$$\exists s_3, s_4, s_5, s_6.s_3 \leq s_1 \wedge s_4 \leq s_2 \wedge el(s_3, s) \wedge el(s_4, s) \wedge$$
$$s_5 \leq s_1 \wedge s_6 \leq s_2 \wedge el(s_5, s') \wedge el(s_6, s') \wedge$$
$$\forall a, s_7, s_8.[s_3 \leq s_7 < s_5 \wedge s_4 \leq s_8 < s_6 \wedge el(s_7, s_8) \supset$$
$$do(a, s_7) \leq s_5 \equiv do(a, s_8) \leq s_6]$$

Lemma 1 *Under our axiomatization, we have that for any δ, s, s', s_1, s_2, if s', s_1, and s_2 are all of equal length, and the schedules s_1 and s_2 are the same between s and s', then $Do(\delta, s, s', s_1)$ iff $Do(\delta, s, s', s_2)$:*

$$s \leq s' \wedge el(s', s_1) \wedge el(s', s_2) \wedge eq(s_1, s_2, s, s') \supset Do(\delta, s, s', s_1) \equiv Do(\delta, s, s', s_2).$$

We mentioned that our formalization of prioritized concurrency $\gg$ is different from that in (De Giacomo, Lespérance, and Levesque 2000). We now show that when a program does not have this operator, and does not have the special nil action used in (De Giacomo, Lespérance, and Levesque 2000), then our axiomatization agrees with that in (De Giacomo, Lespérance, and Levesque 2000).

Theorem 3 *Let δ be a ConGolog complex action without the prioritized concurrency operator $\gg$, and without the special action nil. Let $Do1$ and $Do2$ be the macro operators defined in (De Giacomo, Lespérance, and Levesque 2000) and here in this paper, respectively. We have*

$$\Sigma \cup T_1 \cup T_2 \models \forall \vec{x}, s, s'.Do1(\delta, s, s') \equiv Do2(\delta, s, s'),$$

where $\vec{x}$ is the tuple of the free variables in δ, T_1 the set of axioms about $Trans$, $Trans^*$, and $Final$, and T_2 the set of axioms in this paper about sch, el, split, Full, Empty, P_{δ_1} for each δ_1 such that δ_1^* is in δ, and $P^{\|}_{\delta_1}$ for each δ_1 such that $\delta_1^{\|}$ is in δ.

This theorem is proved using the following lemma

Lemma 2 *Let δ be a ConGolog complex action without $\gg$ and nil. Let $s \leq s'$ be two situations, and $s_2 \leq s_1$ two schedules such that $el(s_2, s)$ and $el(s_1, s')$. Then $Do(\delta, s, s', s_1)$ iff for some sequence $s \leq S_1 \leq S_1' \leq S_2 \leq S_2' \leq \cdots \leq S_n \leq S_n' \leq s'$ of situation terms, and a sequence $\delta = \delta_1, ..., \delta_{n+1}$ of ConGolog complex actions, we have that*

- $Trans(\delta_i, S_i, \delta_{i+1}, S_i')$, $1 \leq i \leq n$;
- $Final(\delta_{n+1}, S_n')$;
- *the sequence $S_1 \leq S_1' \leq S_2 \leq S_2' \leq \cdots \leq S_n \leq S_n'$ conforms with the schedule s_1 starting from s_2: for any s_3, $s_2 < do(Y, s_3) \leq s_1$ iff there is an i such that $el(s_3, S_i)$ and $S_i < S_i'$.*

We have already seen an example program that mentions $\gg$ for which our semantics differs from the one in (De Giacomo, Lespérance, and Levesque 2000). We now proceed to show that our approach can also handle the $\gg$ operator as defined in (De Giacomo, Lespérance, and Levesque 2000). To do this, we introduce a new predicate $PDo(\delta, s, s', s_1)$ (partially do the program), meaning that δ has been carried out from s to s' according schedule s_1. The new definition of $Do(\delta_1 \gg \delta_2, s, s', s_1)$ is as below:

$$\begin{aligned}
&Do(\delta_1 \gg \delta_2, s, s', s_1) \stackrel{def}{=} \exists s_2, s_3\{split(s_1, s_2, s_3) \wedge \\
&\quad Do(\delta_1, s, s', s_2) \wedge Do(\delta_2, s, s', s_3) \wedge \\
&\quad \neg\exists s_4, s_5, s_6, s''[split(s_1, s_4, s_5) \wedge \\
&\quad earlier(s_4, s_2) \wedge do(Y, s_6) \leq s_4 \wedge do(N, s_6) \leq s_2 \wedge \\
&\quad s < s'' \leq s' \wedge el(do(Y, s_6), s'') \wedge \\
&\quad PDo(\delta_1, s, s'', do(Y, s_6))]\}.
\end{aligned}$$

The new predicate PDo is defined as follows. It is the same as Do on primitive actions, test actions, choices, and concurrencies, with Do replaced by PDo:

$$\begin{aligned}
&PDo(A, s, s', s_1) \equiv \exists s_2, s_3, s''.el(s, s_2) \wedge \\
&\quad \{Empty(s_2, s_1) \vee \\
&\quad [s < do(A, s'') \leq s' \wedge s_2 < do(Y, s_3) \leq s_1 \wedge \\
&\quad el(s'', s_3) \wedge Empty(s_2, s_3) \wedge Empty(do(Y, s_3), s_1) \wedge \\
&\quad Poss(A, s'')]\}, \\
&PDo(\varphi?, s, s', s_1) \equiv \exists s_2.s_2 \leq s_1 \wedge el(s, s_2) \wedge \\
&\quad Empty(s_2, s_1), \\
&PDo(\delta_1 \mid \delta_2, s, s', s_1) \equiv \\
&\quad PDo(\delta_1, s, s', s_1) \vee PDo(\delta_2, s, s', s_1), \\
&PDo((\pi x)\delta(x), s, s', s_1) \equiv \exists x.PDo(\delta(x), s, s', s_1), \\
&PDo(\delta_1 \parallel \delta_2, s, s', s_1) \equiv \exists s_2, s_3.split(s_1, s_2, s_3) \wedge \\
&\quad PDo(\delta_1, s, s', s_2) \wedge PDo(\delta_2, s, s', s_3), \\
&PDo(\delta_1 \gg \delta_2, s, s', s_1) \equiv \exists s_2, s_3\{split(s_1, s_2, s_3) \wedge \\
&\quad PDo(\delta_1, s, s', s_2) \wedge PDo(\delta_2, s, s', s_3) \wedge \\
&\quad \neg\exists s_4, s_5, s_6, s''[split(s_1, s_4, s_5) \wedge \\
&\quad earlier(s_4, s_2) \wedge do(Y, s_6) \leq s_4 \wedge do(N, s_6) \leq s_2 \wedge \\
&\quad s < s'' \leq s' \wedge el(do(Y, s_6), s'') \wedge \\
&\quad PDo(\delta_1, s, s'', do(Y, s_6))]\}.
\end{aligned}$$

Partially executing a sequence $\delta_1; \delta_2$ means either partially executing δ_1 or partially executing δ_2 after δ_1 is executed:

$$\begin{aligned}
&PDo(\delta_1; \delta_2, s, s', s_1) \equiv PDo(\delta_1, s, s', s_1) \vee \exists s'', s_2[\\
&\quad s_2 \leq s_1 \wedge Do(\delta_1, s, s'', s_2) \wedge PDo(\delta_2, s'', s', s_1)].
\end{aligned}$$

Similarly for iterations:

$$\begin{aligned}
&PDo(\delta^*, s, s', s_1) \equiv PDo(\delta, s, s', s_1) \vee \\
&\quad \exists s'', s_2.s_2 \leq s_1 \wedge Do(\delta, s, s'', s_2) \wedge PDo(\delta, s'', s', s_1), \\
&PDo(\delta^{\|}, s, s', s_1) \equiv PDo(\delta, s, s', s_1) \vee \\
&\quad \exists s_2, s_3.[split(s_1, s_2, s_3) \wedge s_3 \neq s_1 \wedge \\
&\quad PDo(\delta, s, s', s_2) \wedge PDo(\delta^{\|}, s, s', s_3)].
\end{aligned}$$

Notice that the last axiom will not lead to a cyclic definition as splitting s_1 into s_2 and s_3, with $s_3 \neq s_1$, will eventually lead to s_3 being an empty schedule.

Related work

If we consider generally the problem of formalizing semantics of programs, there is a huge literature about this in computer science. In particular, Golog and its original semantics was strongly influenced by dynamic logic (Harel 1979). The unique feature of Golog is that actions are specified axiomatically by successor state axioms, and the Do macro axiomatizes legal execution paths in the situation calculus.

Baier, Fritz and McIlraith (2007) proposed to translate a Golog program to a basic action theory. Later, Fritz, Baier and McIlraith (2008) extended it to ConGolog. Their motivation was to use Golog and ConGolog to encode domain specific control information in planning. Given a basic action theory D, and a goal G, instead of asking if there is an

executable sequence of actions that can achieve G: whether

$$D \models \exists s.Executable(s) \wedge G[s]$$

holds, they proposed to encode the domain specific control information about D as a ConGolog program δ, and ask if there is a sequence of actions that is a legal execution of δ and can achieve G: whether

$$D \models \exists s.Do(\delta, S_0, s) \wedge G[s].$$

To implement this, they translated D and δ to a new basic action theory D' in a new language, and showed that there is a mapping τ from sequences of actions (situations) in the language of D' to sequences of actions (situations) in the language of D such that $D' \models Executable(S) \wedge G[S]$ iff $D \models Do(\delta, S_0, \tau(S)) \wedge G[\tau(S)]$. Thus, the semantics of δ is captured by D'. While the axioms in D' are also first-order, they use natural numbers. In comparison, our axiomatization uses the same language of the original situation calculus. An interesting question is whether our first-order reformulation of Golog and ConGolog semantics could make the translation from D and δ to D' simpler.

While we consider Golog and ConGolog programs, Gabaldon (2003; 2004) considered "programs" written in temporal logic formulas, and their translation into first-order situation calculus.

Conclusions

We have shown that by using the foundational axioms in the situation calculus, it is possible to give a first-order semantics to Golog and ConGolog programs. In the case of Golog, this is achieved by introducing a new predicate when an iteration is encountered. In the case of ConGolog, this is done by using so-called scheduling situations. The overall message is that once we have second-order induction on the situations, other inductive structures and fixed-points can be defined in first-order terms. We hope the simpler first-order semantics will lead to more effective ways of doing program verification and synthesis in Golog and ConGolog.

Acknowledgments

My thanks to Hector Levesque for his comments on an earlier version of this paper, and for a pointer to a related work. I also thank the reviewers for their helpful comments. This work was supported in part by HK RGC under GRF 616013, and NSFC of China under grant 6137061

Appendix: Proofs

Proof of Theorem 1

Let δ be a complex action, M a model of $\Sigma \cup T_\delta$, and σ a variable assignment for $\vec{x}$, s, and s'. We need to show that

$$M, \sigma \models Do1(\delta, s, s') \equiv Do2(\delta, s, s').$$

This is done by induction on the structure of δ. Notice that $Do1$ and $Do2$ are identical except on iterations, so we need only to prove the inductive step for this construct. We know that

$$M, \sigma \models Do1(\delta^*, s, s')$$

iff there is a sequence of $S_0, .., S_n$, $0 \leq n$, of situation terms, such that under σ, $s = S_0$, $S_n = s'$, and for any $0 \leq i < n$,

$$M, \sigma \models Do1(\delta, S_i, S_{i+1}).$$

On the other hand,

$$M, \sigma \models Do2(\delta^*, s, s')$$

iff

$$M, \sigma \models P_\delta(\vec{x}, s, s').$$

Since M is a model of T_δ, by induction on s', it can be seen that the above holds iff there is a sequence of $S_0, .., S_n$, $0 \leq n$, of situation terms, such that $s = S_0$, $S_n = s'$, and for any $0 \leq i < n$,

$$M, \sigma \models Do2(\delta, S_i, S_{i+1}).$$

Thus the desired result holds by the inductive assumption that

$$M, \sigma \models Do2(\delta, S, S') \equiv Do1(\delta, S, S')$$

for any situation terms S and S'.

Proof of Theorem 2 (Sketch)

Inductively for any complex action δ extended with P_i, $1 \leq i \leq n$, define $Do1^k$ as follows:

$$P_i^0(\vec{v}_i, s, s') \equiv false,$$
$$P_i^k(\vec{v}_i, s, s') \equiv Do1^{k-1}(\delta_i, s, s'),$$
$$Do1^k(\delta, s, s') \stackrel{def}{=} Do1(\delta(\vec{P}/\vec{P^k}), s, s'),$$

where $\delta(\vec{P}/\vec{P^k})$ is the result of replacing each occurrence of $P_i(\vec{t})$ in δ by $P_i^k(\vec{t})$, for every $1 \leq i \leq n$. The second order definition of $Do1(\delta_0, s, s')$ in (Levesque et al. 1997) captures the least fixed points for P_i, meaning that $Do1(\delta_0, s, s')$ iff for some k, $Do1^k(\delta_0, s, s')$. From this, the result of the theorem follows.

Proof of Lemma 1

Suppose $el(s', s_1)$, $el(s', s_2)$, and $eq(s_1, s_2, s, s')$ hold. This means that s_1 and s_2 are the same after s, not just between s and s', as s' is really the end of s_1 and s_2. Suppose $Do(\delta, s, s', s_1)$. We prove by induction on δ that $Do(\delta, s, s', s_2)$ holds as well. The base cases for primitive actions and test actions are straightforward by noting that s_1 and s_2 are the same after s. Inductively, we have the following cases:

- $\delta = \delta_1; \delta_2$. From $Do(\delta, s, s', s_1)$, we have that for some s'' and s_3, $s_3 \leq s_1$, $el(s_2, s'')$, and

  $$Do(\delta_1, s, s'', s_3) \wedge Do(\delta_2, s'', s', s_1).$$

  hold. Now let $s_4 \leq s_2$ and $el(s_4, s_3)$. By inductive assumptions, we have

  $$Do(\delta_1, s, s'', s_4) \wedge Do(\delta_2, s'', s', s_2),$$

  which implies $Do(\delta, s, s', s_2)$.
- The cases for $\delta_1 \mid \delta_2$ and $(\pi x)\delta_1(x)$ are trivial.

- $\delta = \delta_1(\vec{x})^*$. From $Do(\delta, s, s', s_1)$, we have $P_{\delta_1}(\vec{x}, s, s', s_1)$. We show by induction on s that $P_{\delta_1}(\vec{x}, s, s', s_2)$ holds as well. The base case $s = s'$ is trivial. Inductively, assume $s < s'$ and the result holds for all s'' such that $s < s'' \leq s'$. From $P_{\delta_1}(\vec{x}, s, s', s_1)$, we have that for some s'' and s_3:

$$s < s'' \leq s' \wedge s_3 \leq s_1 \wedge el(s_3, s'') \wedge Do(\delta_1, s, s'', s_3) \wedge P_{\delta_1}(\vec{x}, s'', s', s_1).$$

By inductive assumption, we have

$$s < s'' \leq s' \wedge s_3 \leq s_1 \wedge el(s_3, s'') \wedge Do(\delta_1, s, s'', s_3) \wedge P_{\delta_1}(\vec{x}, s'', s', s_2).$$

Now let $s_4 \leq s_2$ be such that $el(s_4, s_3)$, by inductive assumption on δ_1, we have

$$s < s'' \leq s' \wedge s_4 \leq s_2 \wedge el(s_4, s'') \wedge Do(\delta_1, s, s'', s_4) \wedge P_{\delta_1}(\vec{x}, s'', s', s_2).$$

Thus $P_{\delta_1}(\vec{x}, s, s', s_2)$.

- $\delta = \delta_1 \parallel \delta_2$. From $Do(\delta, s, s', s_1)$, we have for some s_3, s_4 such that $split(s_1, s_3, s_4)$,

$$Do(\delta_1, s, s', s_3) \wedge Do(\delta_2, s, s', s_4).$$

Since s_1 and s_2 are the same after s, we can split s_2 the same way after s: $split(s_2, s_5, s_6)$ so that s_3 and s_5, and s_4 and s_6, respectively, are the same after s. By inductive assumption, we then have

$$Do(\delta_1, s, s', s_5) \wedge Do(\delta_2, s, s', s_6).$$

Thus $Do(\delta, s, s', s_2)$.

- The case for $\delta_1 \gg \delta_2$ is similar to the previous case.
- $\delta = \delta_1(\vec{x})^{\parallel}$. The case for $s = s'$ is trivial. Suppose $s < s'$. From $Do(\delta_1(\vec{x})^{\parallel}, s, s', s_1)$, we have $P^{\parallel}_{\delta_1}(\vec{x}, s, s', s_1)$. We show by induction on the number of "Y" actions in s_1 that $P^{\parallel}_{\delta_1}(\vec{x}, s, s', s_2)$ holds as well. If s_1 is all empty, then s_2 is all empty starting from s:

$$\exists s_3(s_3 \leq s_2 \wedge el(s, s_3) \wedge Empty(s_3, s_2)). \quad (14)$$

By our axiom about $P^{\parallel}$, $P^{\parallel}_{\delta_1}(\vec{x}, s, s', s_2)$ holds. Inductively, there are two cases. Case 1: s_1 is all empty starting from s. Then s_2 is all empty starting from s as well. Case 2: s_1 is not all empty starting from s. Then for some s_3 and s_4 such that $split(s_1, s_3, s_4) \wedge s_1 \neq s_3$, we have

$$P^{\parallel}_{\delta_1}(\vec{x}, s, s', s_3) \wedge Do(\delta_1(\vec{x}), s, s', s_4).$$

We can split s_2 in the same way: $split(s_2, s_5, s_6)$ so that $s_2 \neq s_5$, and s_5 and s_3, and s_6 and s_4, respectively, are the same after s. Now the number of "Y" actions in s_3 must be less than that in s_1. Thus by inductive assumptions,

$$P^{\parallel}_{\delta_1}(\vec{x}, s, s', s_5) \wedge Do(\delta_1(\vec{x}), s, s', s_6).$$

Thus $P^{\parallel}_{\delta_1}(\vec{x}, s, s', s_2)$.

This completes the inductive step, thus the proof of the lemma.

Proof of Lemma 2 (Sketch)

Let $s \leq s'$ be two situations, and $s_2 \leq s_1$ two schedules such that $el(s_2, s)$ and $el(s_1, s')$. We prove by induction on δ. There are two base cases, primitive actions and test actions. If $\delta = A$ is a primitive action, then $Do(\delta, s, s', s_1)$ iff for some s_3 and s'',

$$s < do(A, s'') \leq s' \wedge s_2 < do(Y, s_3) \leq s_1 \wedge el(s'', s_3) \wedge Empty(s_2, s_3) \wedge Empty(do(Y, s_3), s_1) \wedge Poss(A, s'')$$

iff for some s_3 and s'', the following are true

- $s \leq s'' < do(A, s'') \leq s'$,
- $Trans(A, s'', nil, do(A, s'')) \wedge Final(nil, do(A, s''))$,
- the sequence $s'' < do(A, s'')$ conforms with the schedule s_1 starting from s_2: it has exactly one "Y" which occurs at the position marked by s''.

Noting that $Trans(A, S, \delta', S')$ holds iff $\delta' = nil$, $S' = do(A, S)$, and $Poss(A, S)$, this proves the lemma for the case.

The case for $\delta = \varphi?$ is almost identical, noting that $Trans(\varphi?, S, \delta', S')$ iff $\varphi[S]$, $S' = S$, and $\delta' = nil$.

Inductively, there are several cases. Suppose $\delta = \delta^1; \delta^2$. Then $Do(\delta^1; \delta^2, s, s', s_1)$ iff for some s_3, s''

$$s_3 \leq s_1 \wedge el(s_3, s'') \wedge Do(\delta^1, s, s'', s_3) \wedge Do(\delta^2, s'', s', s_1).$$

The result then follows from the inductive assumptions on $Do(\delta^1, s, s'', s_3)$ and $Do(\delta^2, s'', s', s_1)$, and the definition of $Trans$ from (De Giacomo, Lespérance, and Levesque 2000). If $\delta = \delta^1 \parallel \delta^2$, then $Do(\delta^1 \parallel \delta^2, s, s', s_1)$ iff for some s_3 and s_4,

$$split(s_1, s_3, s_4) \wedge Do(\delta^1, s, s', s_3) \wedge Do(\delta^2, s, s', s_4).$$

Notice that $split(s_1, s_3, s_4)$ implies that $el(s_1, s_3)$ and $el(s_1, s_4)$. Thus the inductive assumption can apply to $Do(\delta^1, s, s', s_3)$ and $Do(\delta^2, s, s', s_4)$, from which the result follows. The other cases are similar.

Proof of Theorem 3

$Do2(\delta, s, s')$ iff $Do2(\delta, s, s', s_1)$, where s_1 is a full schedule such that $el(s', s_1)$, iff (by Lemma 2) for some sequence $s \leq S_1 \leq S'_1 \leq S_2 \leq S'_2 \leq \cdots \leq S_n \leq S'_n \leq s'$ of situation terms, and a sequence $\delta = \delta_1, ..., \delta_{n+1}$ of ConGolog complex actions, we have that

- $Trans(\delta_i, S_i, \delta_{i+1}, S'_i)$, $1 \leq i \leq n$;
- $Final(\delta_{n+1}, S'_n)$;
- the sequence $S_1 \leq S'_1 \leq S_2 \leq S'_2 \leq \cdots \leq S_n \leq S'_n$ conforms with the the full schedule s_1 starting from s_2, where $s_2 \leq s_1$ and $el(s, s_2)$: for any s_3 such that $s_2 \leq s_3 < s_1$, there is an i such that $el(s_3, S_i)$ and $S_i < S'_i$.

Given that s and s_2 have the same length, and s' and s_1 have the same length, the last condition implies that $S'_i = S_{i+1}$ for every $1 \leq i < n$, $s = S_1$, and $S'_n = s'$. Thus the above condition holds iff $Do1(\delta, s, s')$.

References

Baier, J. A.; Fritz, C.; and McIlraith, S. A. 2007. Exploiting procedural domain control knowledge in state-of-the-art planners. In *ICAPS-2007*, 26–33.

Burgard, W.; Cremers, A. B.; Fox, D.; Hähnel, D.; Lakemeyer, G.; Schulz, D.; Steiner, W.; and Thrun, S. 1999. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence* 114(1-2):3–55.

De Giacomo, G.; Lespérance, Y.; and Levesque, H. J. 2000. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence* 121 (1–2):109–169.

Fritz, C.; Baier, J. A.; and McIlraith, S. A. 2008. ConGolog, sin trans: Compiling congolog into basic action theories for planning and beyond. In *KR-2008*, 600–610.

Funge, J. 2000. Cognitive modeling for games and animation. *Communications of ACM* 43(7):40–48.

Gabaldon, A. 2003. Compiling control knowledge into preconditions for planning in the situation calculus. In *IJCAI-2003*, 1061–1066.

Gabaldon, A. 2004. Precondition control and the progression algorithm. In *KR-2004*, 634–643.

Harel, D. 1979. *First-Order Dynamic Logic*. New York: Springer-Verlag: Lecture Notes in Computer Science 68.

Lespérance, Y.; Levesque, H. J.; and Ruman, S. J. 1997. An experiment in using Golog to build a personal banking assistant. In *Intelligent Agent Systems: Theoretical and Practical Issues, volume 1209 of LNAI*, 27–43. Springer-Verlag.

Levesque, H.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. 1997. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming, Special issue on Reasoning about Action and Change* 31:59–84.

Lin, F., and Reiter, R. 1994. State constraints revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes* 4(5):655–678.

Lin, F. 2007. Situation calculus. In van Harmelen, F.; Lifschitz, V.; and Porter, B., eds., *Handbook of Knowledge Representation*. Elsevier.

McCarthy, J. 1968. Situations, actions and causal laws. In Minsky, M., ed., *Semantic Information Processing*. MIT Press, Cambridge, Mass. 410–417.

McIlraith, S. A., and Son, T. C. 2002. Adapting Golog for composition of semantic web services. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, 482–496.

Reiter, R. 1993. Proving properties of states in the situation calculus. *Artificial Intelligence* 64:337–351.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Model Checking Unbounded Artifact-Centric Systems

Alessio Lomuscio and Jakub Michaliszyn
Department of Computing, Imperial College London, UK

Abstract

Artifact-centric systems are a recent paradigm for representing and implementing business processes. We present further results on the verification problem of artifact-centric systems specified by means of FO-CTL specifications. While the general problem is known to be undecidable, results in the literature prove decidability for artifact systems with infinite domains under boundedness and conditions such as uniformity. We here follow a different approach and investigate the general case with infinite domains. We show decidability of the model checking problem for the class of artifact-centric systems whose database schemas consist of a single unary relation, and we show that that the problem is undecidable if artifact systems are defined by using one binary relation or two unary relations.

1 Introduction

Artifact-centric systems (ACS) have been put forward as a framework for reasoning about and implementing data-aware business processes (Alonso et al. 2004; Hull 2008; Hull et al. 2011). Artifacts are constructs consisting of *data* and *lifecycles*. The data component is given by means of a relational database, i.e., a set of finite relations with fixed *schema*. The lifecycles describe how the artifacts may interact and evolve over time.

ACS can be programmed via the Guard-Stage-Milestone (GSM) language (Hull et al. 2011). The iHub (Heath III et al. 2013) is a production and execution suite for ACS implemented in GSM. Both GSM and the iHub are designed to help stakeholders encode business interactions intuitively and efficiently.

A problem that naturally arises is whether ACS are correct against specifications. GSMC, a model checking tool for GSM systems, has recently been put forward (Gonzalez, Griesmayer, and Lomuscio 2012) to assess this problem.

While GSMC has shown considerable promise, verifying GSM programs via model checking remains highly problematic. Classical techniques based on model checking (Clarke, Grumberg, and Peled 1999) are typically insufficient as they normally tackle finite-state systems. Yet artifacts exhibit infinitely many different possible configurations. It is therefore important to establish at a fundamental level under what conditions an artifact-system can actually be verified.

The verification problem for ACS specified by quantified temporal specifications is known to be undecidable (Deutsch et al. 2009; Hariri et al. 2011; Belardinelli, Lomuscio, and Patrizi 2011) and considerable research has gone into the exploration of decidable fragments.

Contribution. In this paper we explore novel boundaries in the decidability conditions for the model checking problem of ACS against quantified, temporal specifications. While much of the recent work has focused on the identification of relational properties on the generated models for complete and decidable abstraction procedures, an alternative avenue of investigation consists in considering classes of specifications for the artifacts' transitions. To this end, we here define the logic CARL which we use to model the artifacts' lifecycles. By using CARL to model the artifacts' evolutions, we can express Boolean combinations of the basic relational properties: "**c**heck that the relation is empty, "**a**dd one arbitrary element to the relation", "**r**emove one arbitrary element", and "**l**eave the relation unchanged". We show that model checking ACS described via CARL against reachability specifications is undecidable thereby providing further insights on the difficulty of verifying data-aware systems.

We then focus on restricting the database schema appearing in the ACS. We show that the model checking problem remains undecidable if the schema contain either one relation of arity greater than one, or at least two arbitrary relations. Our undecidability proofs involve the encoding of two-counter machines (Minsky 1967).

Finally, we show that the model checking problem is decidable if we consider only artifacts whose schema consists of a single unary relation and sentence-atomic temporal specifications. We prove this result in three steps. Firstly, we study the expressive power of first-order logic over unary relations using Ehrenfeucht-Fraisse games. Then, we show how to transform an ACS and a specification into an infinite Kripke structure and a CTL specification. Finally, we prove that the resulting infinite Kripke structure can be encoded as a pushdown system. We then define a CTL* specification that the pushdown system satisfies if and only if the ACS satisfies the original FO-CTL specification. Since the former problem is decidable (Bozzelli 2007), we conclude that the latter is decidable as well.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Related Work. Previous research on verifying artifact-centric systems has focused on abstraction techniques where semantical conditions such as "uniformity" and "weak acyclicity" are shown to guarantee decidability under the assumption that systems are bounded, i.e., either the states or the runs do not contain more than a given number of predicates (Belardinelli, Lomuscio, and Patrizi 2012b; Hariri et al. 2013). Differently from these approaches, the investigation here presented makes no assumption on the boundedness of the artifact-systems to be studied.

Model checking artifact-centric systems against a quantified version of LTL was studied in (Deutsch et al. 2009). It was shown that the problem is generally undecidable, but becomes decidable if one considers *guarded artifact systems* and *guarded specifications*, a form of quantification where no variables appear free in a temporal context. A PSPACE model checking algorithm for fixed-arity schemas is also shown in (Deutsch et al. 2009). While the results presented are positive, compared to the classes of artifacts studied here, the expressivity of the fragment studied is limited.

More recently (Hariri et al. 2013) presented results on the model checking problem against specifications given in the first-order extension of the μ-calculus. The problem is undecidable in this setting; however, it is shown that by adding additional restrictions, including forbidding fresh, unbounded data along a run, the problem is decidable. Also in that line, and differently from this paper, systems are assumed to be bounded and sufficient conditions based on acyclicity for boundedness are established.

While all results above focus on complete procedures, in recent work (Belardinelli and Lomuscio 2013) partial model checking procedures against the universal fragment of FO-CTL have been explored. Differently from (Belardinelli, Lomuscio, and Patrizi 2012b) these are given for bounded but possibly non-uniform artifact-centric systems. Conversely, the results established in this paper concern unbounded, but uniform systems.

Scheme of the paper. In Section 2 we provide the background on ACS and relevant concepts, and we define the model checking problem. We observe that the problem is generally undecidable. In Section 3 we investigate the implications of restricting the database schemas and the expressive power of the language used to model the artifacts' lifecycles. We show that under various restrictions the problem remains undecidable. In Section 4 we characterise a scenario in which the model checking problem is decidable and explore its complexity. Taken together the results in the two sections show that the conditions we identify can be interpreted as being maximal for decidability. We conclude in Section 5.

2 Verifying Artifact Systems

Artifacts are operational models of *business process* (Cohn and Hull 2009). Each artifact is composed by two parts: a *data model*, which captures a fragment of the structure of the information relevant to the process, and a *lifecycle model*, which is a specification of their evolution. In this paper we follow the formalisation proposed in (Hariri et al. 2011) and define artifact systems by means of a (relational) database and a set of artifact transitions, which account for the artifact data models and the artifact lifecycles, respectively.

Definition 1 (Database schema). *A* database schema *is a set* $\mathcal{D} = \{P_1/a_1, \ldots, P_n/a_n\}$ *of* relation (or predicate) symbols P_i*, each associated with its arity* $a_i \in \mathbb{N}$.

Definition 2 (Database interpretation). *Given a database schema* $\mathcal{D}$*, a* $\mathcal{D}$-interpretation (or $\mathcal{D}$-instance) over an countable interpretation domain U *is a mapping* D *associating each relation symbol* P_i *with a* finite a_i*-ary relation over* U*, i.e.,* $D(P_i) \subseteq U^{a_i}$.

The set of all $\mathcal{D}$-interpretations over a given domain U is denoted by $\mathcal{I}_{\mathcal{D}}(U)$. The *active domain* of D, $adom(D)$, is the set of all U-elements occurring in some tuple of some predicate interpretation $D(P_i)$.

2.1 Artifact systems

To model transitions between states of the underlying databases, we use *unprimed* and *primed* relational symbols from $\mathcal{D}$, that refer to relations in the current and the successor state, respectively. Intuitively, given two states (i.e., two $\mathcal{D}$-interpretations) D and D', the operator $\oplus$ constructs a new "joint" interpretation $D \oplus D'$, interpreting unprimed relational symbols in D, and primed in D'.

As we study artifacts systems whose transitions are described by various logics, below we introduce the notion of an *artifact system over a logic* $\mathcal{L}_{\mathcal{D}}$.

Definition 3 ($\mathcal{L}_{\mathcal{D}}$-artifact system). *An* $\mathcal{L}_{\mathcal{D}}$-artifact system *is a tuple* $\mathcal{S} = \langle \mathcal{D}, U, D_0, \mathbb{T} \rangle$*, where:*

- $\mathcal{D} = \{P_1/a_1, \ldots, P_n/a_n\}$ *is a database schema;*
- U *is a countable interpretation domain;*
- D_0 *is an* initial database instance*;*
- $\mathbb{T}$ *is a finite set of sentences of the logic* $\mathcal{L}_{\mathcal{D}}$*, called* artifact transitions*.*

In the definition above we assume that the sentences of the logic $\mathcal{L}_{\mathcal{D}}$ state properties of $D \oplus D'$, i.e., the notion "$D \oplus D' \models_{\mathcal{L}_{\mathcal{D}}} \Phi$" is defined for every sentence Φ of $\mathcal{L}_{\mathcal{D}}$.

The semantics of an artifact system is given in terms of its possible executions, captured by a Kripke structure, whose states are instances of the database schema and whose transitions correspond to executions of artifact transitions.

Definition 4 (Kripke structures). *A Kripke structure is a tuple* $\mathcal{K} = \langle \Sigma, D_0, \tau, \pi \rangle$*, where* Σ *is the* set of states*;* $D_0 \in \Sigma$ *is the* initial state*;* $\tau \subseteq \Sigma \times \Sigma$ *is the* transition relation*;* π *is the labelling function.*

For convenience, we assume a single initial state, but all the results presented below also hold if we allow any finite number of initial states. When the labelling function is irrelevant, we will drop it and represent a Kripke structure as a triple $\mathcal{K} = \langle \Sigma, D_0, \tau \rangle$.

Note that in the definition above we include states not reachable from D_0 by τ. This is not the case below.

Definition 5 (Model of an artifact system). *Given an artifact system* $\mathcal{S} = \langle \mathcal{D}, U, D_0, \mathbb{T} \rangle$ *over* $\mathcal{L}_{\mathcal{D}}$*, the* model of $\mathcal{S}$ *is the Kripke structure* $\mathcal{K}_{\mathcal{S}} = \langle \Sigma, D_0, \tau \rangle$*, where:*

- $\Sigma \subseteq \mathcal{I}_{\mathcal{D}}(U)$ *is the* set of states *reachable from* D_0 *using* τ*;*
- $D_0 \in \Sigma$ *is the* initial state*;*
- τ *is the* transition relation *such that* $\tau(D, D')$ *for some artifact transition* $\Phi \in \mathbb{T}$*,* $D \oplus D' \models_{\mathcal{L}_{\mathcal{D}}} \Phi$*.*

We denote the unique model of an artifact system $\mathcal{S}$ by $\mathcal{K}_{\mathcal{S}}$.

Following (Belardinelli, Lomuscio, and Patrizi 2011), we define *runs* on a Kripke structure.

Definition 6 ($\mathcal{K}$-runs). *Given an artifact system* S *and its model* $\mathcal{K} = \langle \Sigma, D_0, \tau \rangle$*, a* $\mathcal{K}$-run r *from a* $\mathcal{K}$*-state* $D \in \Sigma$ *is an infinite sequence of* $\mathcal{K}$*-states* $r = D^0 \rightarrow D^1 \rightarrow \cdots$ *such that* $D^0 = D$ *and* $\tau(D^i, D^{i+1})$*, for* $i \geq 0$*. For every run* r *and* $i \geq 0$*, we define* $r(i)$ *as* D^i*.*

Note that, unlike much of the literature on the subject, we do not require the runs to be bounded.

Also note that we consider only infinite, countable interpretation domains. To make the notation lighter, we assume without loss of generality that the interpretation domain is always the set of naturals $\mathbb{N}$.

2.2 Model checking artifact systems

In the following we use the logic $FO^{V}_{\mathcal{D},C}$ to define the transitions of artifact-centric systems. The logic $FO^{V}_{\mathcal{D},C}$ is the fragment of first-order logic in which formulae are built by using predicates from $\mathcal{D} \oplus \mathcal{D}'$, constants from C, and variables from V.

Definition 7 (Logic $FO^{V}_{\mathcal{D},C}$). *Given a database schema* $\mathcal{D} = \{P_1/a_1, \ldots, P_n/a_n\}$*, the language of the logic* $FO^{V}_{\mathcal{D},C}$ *is defined by the following BNF:*

$$\Phi ::= t = t' \mid P_i(\vec{t}) \mid P'_i(\vec{t}) \mid \neg\Phi \mid \Phi \vee \Phi \mid \forall x \Phi,$$

where $t, \bar{t} \in C \cup V$ *and* $\vec{t}$ *is an* a_i*-tuple of elements from* $C \cup V$*.*

We use parentheses as standard when required.

Let $* = \{x, y, z, x_1, \ldots\}$ be a countable set of variables and $\mathbf{2} = \{x, y\}$. We consider the following logics: $FO^{*}_{\mathcal{D},\mathbb{N}}$, $FO^{*}_{\mathcal{D},\emptyset}$, and $FO^{\mathbf{2}}_{\mathcal{D},\emptyset}$. We denote formulae in these logics with Greek upper case letters Φ, Ψ, while specifications against which systems are checked will be denoted by Greek lower case letters φ, ψ.

We use the standard abbreviations $\exists$, $\top$, $\bot$, $\wedge$, $\Rightarrow$, and $\neq$. Free and bound variables are defined as standard. A *sentence* is a formula with no free variables.

A U*-assignment* is a total function $\sigma : V \mapsto U$. For technical convenience, we assume that every U-assignment σ is extended to C and is the identity on it, i.e., $\forall c \in C, \sigma(c) = c$. Given an assignment σ, we denote by σ^x_u the U-assignment s.t. $\sigma^x_u(x) = u$ and $\sigma^x_u(\overline{x}) = \sigma(\overline{x})$, for every $\overline{x} \in V$ s.t. $\overline{x} \neq x$.

We adjust the standard active-domain semantics used in the artifact systems literature (Belardinelli, Lomuscio, and Patrizi 2011; 2012a) to the setting here defined.

Definition 8 (Active-Domain Semantics for FO-formulae). *Given a database schema* $\mathcal{D}$*, two* $\mathcal{D}$*-interpretations* D, D' *over* $\mathbb{N}$*, an* $\mathbb{N}$*-assignment* σ*, and a FO-formula* $\Phi \in FO^{*}_{\mathcal{D},C}$ *over* $\mathcal{D}$*, we inductively define whether* $D \oplus D'$ *satisfies* Φ *under* σ*, written* $(D \oplus D', \sigma) \models \Phi$*, as follows:*

$(D \oplus D', \sigma) \models t = \bar{t}$ *iff* $\sigma(t) = \sigma(\bar{t})$

$(D \oplus D', \sigma) \models P_i(\vec{t})$ *iff* $\langle \sigma(t_1), \ldots, \sigma(t_\ell) \rangle \in D(P_i)$

$(D \oplus D', \sigma) \models P'_i(\vec{t})$ *iff* $\langle \sigma(t_1), \ldots, \sigma(t_\ell) \rangle \in D'(P_i)$

$(D \oplus D', \sigma) \models \neg\Phi$ *iff* $(D \oplus D', \sigma) \not\models \Phi$

$(D \oplus D', \sigma) \models \Phi \vee \Psi$ *iff* $(D \oplus D', \sigma) \models \Phi$ *or* $(D \oplus D', \sigma) \models \Psi$

$(D \oplus D', \sigma) \models \forall x \Phi$ *iff for every* $u \in adom(D)$ $(D \oplus D', \sigma^x_u) \models \Phi$

A formula Φ *is* true *in* $D \oplus D'$*, written* $D \oplus D' \models \Phi$*, iff* $(D \oplus D', \sigma) \models \Phi$ *for all* $\mathbb{N}$*-assignments* σ*.*

Notice that two constants map the same element iff they are the same constant.

We focus on the problem of verifying an artifact system against a temporal specification of interest. We adjust the notion of sentence-atomic FO-CTL (Belardinelli, Lomuscio, and Patrizi 2011) to the case of logic $FO^{*}_{\mathcal{D},\mathbb{N}}$.

Definition 9 (FO ^{sa} CTL formulae). *Given an artifact system* $\mathcal{S} = \langle \mathcal{D}, \mathbb{N}, D_0, \mathbb{T} \rangle$*, the language of* sentence-atomic *FO-CTL (denoted by FO* ^{sa} *CTL) formulae over* $\mathcal{S}$ *is inductively defined as follows:*

$$\varphi ::= \Phi_P \mid \neg\varphi \mid \varphi \vee \varphi \mid AX\varphi \mid A\varphi\mathcal{U}\varphi \mid E\varphi\mathcal{U}\varphi$$

where Φ_P *is an* $FO^{*}_{\mathcal{D},\mathbb{N}}$ *sentence where no primed terms nor primed predicates appear.*

Note from above that while we use the $FO^{*}_{\mathcal{D},\mathbb{N}}$ logic as the building block for artifacts' descriptions we do not use primed variables in the specifications.

The notions of free and bound variables can be extended in the obvious way to FO ^{sa} CTL. We use the standard abbreviations $EX\varphi \equiv \neg AX\neg\varphi$, $AF\varphi \equiv A\top\mathcal{U}\varphi$, $AG\varphi \equiv \neg E\top\mathcal{U}\neg\varphi$, $EF\varphi \equiv E\top\mathcal{U}\varphi$, and $EG\varphi \equiv \neg A\top\mathcal{U}\neg\varphi$.

For a system S, the semantics of FO ^{sa} CTL formulae is provided in terms of its model $\mathcal{K}_{\mathcal{S}}$.

Definition 10 (Satisfaction for FO ^{sa} CTL). *Consider a system* $\mathcal{S}$ *and its model* $\mathcal{K}_{\mathcal{S}}$*, a formula* φ *of FO* ^{sa} *CTL and a* $\mathcal{K}_{\mathcal{S}}$*-state* $D \in \Sigma$*, the satisfaction relation* $\models$ *is inductively defined as follows:*

$(\mathcal{K}_{\mathcal{S}}, D) \models \Phi_P$ *iff* $D \models \Phi_P$

$(\mathcal{K}_{\mathcal{S}}, D) \models \neg\varphi$ *iff* $(\mathcal{K}_{\mathcal{S}}, D) \not\models \varphi$

$(\mathcal{K}_{\mathcal{S}}, D) \models \varphi \vee \psi$ *iff* $(\mathcal{K}_{\mathcal{S}}, D) \models \varphi$ *or* $(\mathcal{K}_{\mathcal{S}}, D) \models \psi$

$(\mathcal{K}_{\mathcal{S}}, D) \models AX\varphi$ *iff for all* $\mathcal{K}_{\mathcal{S}}$*-runs* r *s.t.* $r(0) = D$*,* $(\mathcal{K}_{\mathcal{S}}, r(1)) \models \varphi$

$(\mathcal{K}_{\mathcal{S}}, D) \models A\varphi\mathcal{U}\psi$ *iff for all* $\mathcal{K}_{\mathcal{S}}$*-runs* r *s.t.* $r(0) = D$*,* $\exists k \geq 0$ *s.t.* $(\mathcal{K}_{\mathcal{S}}, r(k)) \models \psi$ *and* $\forall j$ *s.t.* $0 \leq j < k$*,* $(\mathcal{K}_{\mathcal{S}}, r(j)) \models \varphi$

$(\mathcal{K}_{\mathcal{S}}, D) \models E\varphi\mathcal{U}\psi$ *iff for some* $\mathcal{K}_{\mathcal{S}}$*-run* r*,* $r(0) = D$*,* $\exists k \geq 0$ *s.t.* $(\mathcal{K}_{\mathcal{S}}, r(k)) \models \psi$*, and* $\forall j$ *s.t.* $0 \leq j < k$*,* $(\mathcal{K}_{\mathcal{S}}, r(j)) \models \varphi$

A formula φ is true *in $\mathcal{K}_\mathcal{S}$, written $\mathcal{K}_\mathcal{S} \models \varphi$, iff $(\mathcal{K}_\mathcal{S}, D_0) \models \varphi$. We say that $\mathcal{S}$ satisfies φ, written $\mathcal{S} \models \varphi$, iff $\mathcal{K}_\mathcal{S} \models \varphi$.*

Observe that satisfaction of $FO^*_{\mathcal{D},\mathbb{N}}$ is defined on joins $D \oplus D'$ as it deals with primed and unprimed predicates. Here we write $D \models \Phi_P$ because we only need to interpret only the unprimed predicates.

In the rest of this paper we focus on the following problem.

Definition 11 (Model checking problem). *Given a logic $\mathcal{L}_\mathcal{D}$, an $\mathcal{L}_\mathcal{D}$-artifact system $\mathcal{S}$ and a FO $\underline{^{sa}}$ CTL specification φ, the* model checking problem *involves establishing whether $\mathcal{K}_\mathcal{S} \models \varphi$.*

Observe that the input can be given in finite terms, as the description of an $\mathcal{L}_\mathcal{D}$-artifact system is finite; so the problem is well-defined. This problem can be shown to be undecidable in the general case.

Theorem 12. *Model checking an arbitrary $FO^*_{\mathcal{D},\mathbb{N}}$-artifact system against a FO $\underline{^{sa}}$ CTL specification is undecidable.*

Proof idea. The proof in (Belardinelli, Lomuscio, and Patrizi 2011) can be adapted to this case. □

In the following we will explore the decidability for restrictions of the problem above.

3 Model Checking $CARL_{\mathcal{D},\mathbb{N}}$–Artifact Systems

One possibility to obtain decidability is to consider a weak form of quantification when writing artifacts' descriptions. For example we could consider the two variable fragment of first-order logic (Grädel, Kolaitis, and Vardi 1997) or the guarded fragment (Grädel 1999).

In this paper we follow a different approach and address the question of whether one can choose a reasonable logic $\mathcal{L}_\mathcal{D}$ in the artifacts' descriptions so that model checking $\mathcal{L}_\mathcal{D}$-artifact systems is decidable.

First, we identify some essential properties of the artifact systems. Then, we define the logic capable of expressing only Boolean combinations of such properties. Finally, we show that in most cases model checking artifact systems defined using this logic is undecidable.

In the proofs that follow, we often use the FO $\underline{^{sa}}$ CTL specification $EG\,\top$. This is intended to show that the negative results we present cannot be overcome by adopting a weaker specification language.

3.1 The logic $CARL_{\mathcal{D},\mathbb{N}}$

The logic $CARL_{\mathcal{D},\mathbb{N}}$ describes properties of unary relations only. $CARL_{\mathcal{D},\mathbb{N}}$ expresses Boolean combinations of the following properties of an unary relation P:

- P contains a constant c;
- P is empty;
- One element is added to P, i.e., $P' = P \cup \{x\}$ for some $x \notin P$;
- One element is removed from P, i.e., $P = P' \cup \{x\}$ for some $x \notin P'$;
- P does not change, i.e., $P = P'$.

These properties are very basic and natural, but are sufficient to express interesting properties for artifact systems. However, as we see below, even the properties above already induce undecidability of the model checking problem.

Definition 13 (Logic $CARL_{\mathcal{D},C}$). *Given a database schema $\mathcal{D} = \{P_1/1, \ldots, P_n/1\}$, the language of the logic $CARL_{\mathcal{D},C}$ is inductively defined by the following BNF:*

$$\begin{aligned}\Phi ::= \quad & \neg\Phi \mid \Phi \vee \Phi \mid P_i(c) \mid P'_i(c) \mid \textsc{Empty}(P_i) \mid \\ & \textsc{Equal}(P_i) \mid \textsc{Insert}(P_i) \mid \textsc{Delete}(P_i)\end{aligned}$$

where c is an element of the set of constants C.

Recall that $P_i(c)$ refers to the relation P_i *before* the transition, and $P'_i(c)$ refers to the relation *after* the transition. For convenience the formal semantics of $CARL_{\mathcal{D},C}$ is given by translating it into $FO^2_{\mathcal{D},\mathbb{N}}$.

Definition 14 (Semantics of $CARL_{\mathcal{D},\mathbb{N}}$). *We define a translation of $\mathfrak{T}$ to $FO^2_{\mathcal{D},\mathbb{N}}$ as follows.*

$$\begin{aligned}
\mathfrak{T}(\neg\Phi) &= \neg\mathfrak{T}(\Phi) \\
\mathfrak{T}(\Phi_1 \vee \Phi_2) &= \mathfrak{T}(\Phi_1) \vee \mathfrak{T}(\Phi_2) \\
\mathfrak{T}(P_i(c)) &= P_i(c) \\
\mathfrak{T}(P'_i(c)) &= P'_i(c) \\
\mathfrak{T}(\textsc{Empty}(P_i)) &= \forall x.\neg P_i(x) \\
\mathfrak{T}(\textsc{Equal}(P_i)) &= \forall x.P'_i(x) \Leftrightarrow P_i(x) \\
\mathfrak{T}(\textsc{Insert}(P_i)) &= \exists x.\neg P_i(x) \wedge P'_i(x) \\
&\quad \wedge \forall y.y \neq x \Rightarrow (P_i(y) \Leftrightarrow P'_i(y)) \\
\mathfrak{T}(\textsc{Delete}(P_i)) &= \exists x.P_i(x) \wedge \neg P'_i(x) \\
&\quad \wedge \forall y.y \neq x \Rightarrow (P_i(y) \Leftrightarrow P'_i(y))
\end{aligned}$$

Then, the semantics of $CARL_{\mathcal{D},\mathbb{N}}$ is defined using the semantics of $FO^*_{\mathcal{D},\mathbb{N}}$ by considering $(D \oplus D', \sigma) \models \Phi$ iff $(D \oplus D', \sigma) \models \mathfrak{T}(\Phi)$.

3.2 Undecidability of model checking $CARL_{\mathcal{D},\mathbb{N}}$-artifacts systems

A key result of this section is the following theorem.

Theorem 15. *For any database schema $\mathcal{D}$ that contains two unary relation symbols, model checking $CARL_{\mathcal{D},\mathbb{N}}$-artifact systems against FO $\underline{^{sa}}$ CTL is undecidable.*

The proof uses two-counter finite automata (Minsky machines) defined as follows.

Definition 16. *A* two-counter automaton *is a tuple $\mathcal{A} = \langle Q, q_0, q_f, R\rangle$, where $Q = \{q_0, q_1, \ldots, q_f\}$, q_0 is an initial state, q_f is a final state, and $R : Q \times \{\top, \bot\} \times \{\top, \bot\} \times Q \times \{-1, 0, 1\} \times \{-1, 0, 1\}$ is the transition relation.*

A *configuration* for a two-counter automata is a triple (q_i, n, m) where $q_i \in Q$ and $n, m \in \mathbb{N}$. Let $Empty(x)$ be equal $\top$ if $x = 0$ and $\bot$ otherwise. We define a relation $\rightarrow^R$ such that for two configuration $\beta_1 = (q_i, n, m)$, $\beta_2 = (q_j, n', m')$ we have $\beta_1 \rightarrow^R \beta_2$ iff we have $(q_i, Empty(n), Empty(m), q_j, n' - n, m' - m) \in R$. A *run* is a possibly infinite sequence of configurations such that for all consecutive configurations β_1, β_2 we have $\beta_1 \rightarrow^R \beta_2$.

A run is *accepting* if it contains a configuration containing q_f.

The following lemma follows from (Minsky 1967).

Lemma 17. *The problem whether there exists an infinite run of a given two-counter automaton is undecidable.*

We can now we prove Theorem 15.

Proof. For a given two-counter automaton $\mathcal{A} = \langle Q, q_0, q_f, R\rangle$, we show how to define a $CARL_{\mathcal{D},\mathbb{N}}$-artifact systems $S^{\mathcal{A}}$ and an FO^{sa}CTL formula φ such that $S^{\mathcal{A}} \models \varphi$ iff $\mathcal{A}$ has an infinite run. Firstly, we construct a system with tree relations, and then we show how one of the relations can be removed.

Three relations. Let $S^{\mathcal{A}} = \langle \mathcal{D}^{\mathcal{A}}, \mathbb{N}, D_0^{\mathcal{A}}, \mathbb{T}^{\mathcal{A}}\rangle$, where $\mathcal{D}^{\mathcal{A}} = \{P_1/1, P_2/1, P_3/1\}$, $D_0^{\mathcal{A}}$ be such that P_1 and P_2 are empty and P_3 is a singleton relation containing 0. Our aim is to define the set $\mathbb{T}^{\mathcal{A}}$ so that each database interpretation $D = \{P_1^D, P_2^D, P_3^D\}$ in the model $\mathcal{K}_{S^{\mathcal{A}}}$ of $S^{\mathcal{A}}$ satisfies $|P_3^D \cap \{0, \dots, f\}| = 1$ and represents a configuration $(q_i, |P_1^D|, |P_2^D|)$ where $q_i \in P_3^D$. In other words, the size of P_1^D/P_2^D represents the value of the first/second counter and P_3^D contains the state.

Let $empty_{\top}(R) = \textsc{Empty}(R)$ and $empty_{\bot}(R) = \neg\textsc{Empty}(R)$. Let $modify_{-1}(R) = \textsc{Delete}(R)$, $modify_0(R) = \textsc{Equal}(R)$, and $modify_{+1}(R) = \textsc{Insert}(R)$.

For each tuple $t = (q_i, \phi_1, \phi_2, q_j, \Delta_1, \Delta_2) \in R$, we define an artifact transition α_t as:

$$P_3(i) \wedge empty_{\phi_1}(P_1) \wedge empty_{\phi_2}(P_2) \wedge P_3'(j) \wedge \bigwedge_{k\neq j, k\leq f} \neg P_3(k) \wedge modify_{\Delta_1}(P_1) \wedge modify_{\Delta_2}(P_2)$$

Finally, let $\mathbb{T}^{\mathcal{A}}$ be the set of all α_t. It can be observed that paths in $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ represent the computations of $\mathcal{A}$. Therefore, there is an infinite path in $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ if and only if there is an infinite run in of $\mathcal{A}$. The existence of an infinite path corresponds to the truth of the formula $EG\top$.

Two relations. We show that we do not need a relation to store the states. The *configuration description* of a configuration (q, n_1, n_2) of $\mathcal{A}$ is a triple (q, v_1, v_2) for $v_i \in \{\top, \bot, ?\}$ s.t. if $v_i = \top$ then $n_i = 0$, and if $v_i = \bot$ then $n_i > 0$. The *transition description* is a triple (q, Δ_1, Δ_2) for $\Delta_i \in \{-1, 0, 1\}$ that is intended to represent the artifact transition "change the value of the i-th counter by Δ_i and change the state to q".

Let $\mathcal{Z}_c = \{0, 1, \dots, 9|Q| - 1\}$ and $\mathcal{Z}_a = \{9|Q|, \dots, 18|Q| - 1\}$. We choose an arbitrary bijection function $encode_c : Q \times \{\bot, \top, ?\} \times \{\bot, \top, ?\} \to \mathcal{Z}_c$ to encode the configuration descriptions as numbers and $encode_a : Q \times \{-1, 0, +1\} \times \{-1, 0, +1\} \to \mathcal{Z}_a$ to encode transition descriptions. Below we identify descriptions with their encodings.

To simplify the proof, we describe the set $\mathbb{T}$ informally.

For $X \subseteq \mathcal{Z}_a \cup \mathcal{Z}_c$, let $exactly(X, R)$ be a formula stating that $R \cap (\mathcal{Z}_a \cup \mathcal{Z}_c) = X$. For $R \in \{P_1, P_1', P_2, P_2'\}$, such a formula can be easily expressed in $CARL_{\mathcal{D},\mathbb{N}}$.

A configuration $\beta = (q_i, n, m)$ will be encoded as a database interpretation $D^{\beta} = (P_1^{\beta}, P_2^{\beta})$ such that $exactly(\{(q, ?, ?)\}, P_1)$, $exactly(\emptyset, P_2)$, $|P_1^{\beta}| = n + 1$ and $|P_2^{\beta}| = m$. So, the initial interpretation of $\mathcal{S}^{\mathcal{A}}$ will be s.t. $P_1 = \{(q_0, ?, ?)\}$ and $P_2 = \emptyset$.

The update of a configuration is performed in several steps by using the rules described below. We will denote transitions informally; e.g., "add n to P_1" stands for $\neg P_1(n) \wedge P_1'(n) \wedge \textsc{Insert}(P_1)$, etc. The rules below should be defined for all appropriate $q, q' \in Q$, $e_1, e_2 \in \{\bot, \top\}$ and $\Delta_1, \Delta_2 \in \{-1, 0, 1\}$.

We use the following example to illustrate the encoding. Let $\nu = (q_1, 0, 1)$, $\nu' = (q_2, 1, 1)$, be such $\nu \to^R \nu'$ and assume that $|Q| = 5$, i.e. $\mathcal{Z}$ contains only numbers less than 90. Assume that ν is encoded as $P_1 = \{(q_1, ?, ?)\}, P_2 = \{95\}$.

1. If $P_1((q, ?, ?))$, then if $\textsc{Empty}(P_2)$, add $(q, ?, \top)$ to P_2, otherwise add $(q, ?, \bot)$ to P_2. In both cases, remove $(q, ?, ?)$ from P_1.
   In the example, P_1 and P_2 change as follows: $P_1 = \{\}, P_2 = \{95, (q_1, ?, \top)\}$.
2. If $\textsc{Empty}(P_1)$ and $P_2((q, ?, e_2))$, add $(q, \top, e_2)$ to P_1, otherwise add $(q, \bot, e_2)$ to P_1. In both cases, remove $(q, ?, \bot)$ from P_2.
   In the example, P_1 and P_2 change as follows: $P_1 = \{(q, \top, \bot)\}, P_2 = \{95\}$.
3. If $P_1((q, e_1, e_2))$ and $(q, e_1, e_1, q', \Delta_1, \Delta_2) \in R$, then add (q', Δ_1, Δ_2) to P_1.
   In the example, P_1 and P_2 change as follows: $P_1 = \{(q, \top, \bot), (q', +1, 0)\}, P_2 = \{95\}$.
4. If $P_1((q', \Delta_1, \Delta_2))$ and $P_1((q, e_1, e_2))$, remove (q, e_1, e_2) from P_1 and modify P_2 according to Δ_2, i.e., if $\Delta_2 = -1$ then remove one element and so forth.
   In the example, P_1 and P_2 change as follows: $P_1 = \{(q', +1, 0)\}, P_2 = \{95\}$.
5. If $P_1((q', \Delta_1, \Delta_2))$ and P_2 does not contain (q, e_1', e_2') for any e_1', e_2', and $\neg P_2((q', \Delta_1, \Delta_2))$, add (q', Δ_1, Δ_2) to P_2.
   In the example, P_1 and P_2 change as follows: $P_1 = \{(q', +1, 0)\}, P_2 = \{95, (q', +1, 0)\}$.
6. If P_1 and P_2 contain (q', Δ_1, Δ_2), remove (q', Δ_1, Δ_2) from P_1 and add $(q', ?, ?)$ to P_2. *In the example, P_1 and P_2 change as follows:* $P_1 = \{\}, P_2 = \{95, (q', +1, 0), (q', ?, ?)\}$.
7. If $P_2((q', \Delta_1, \Delta_2))$ and $\neg P_1((q', \Delta_1, \Delta_2))$, then modify P_1 according to Δ_1 and remove (q', Δ_1, Δ_2) from P_2.
   In the example, P_1 and P_2 change as follows: $P_1 = \{1543\}, P_2 = \{95, (q', ?, ?)\}$.
8. If $P_2((q', ?, ?))$, remove $(q', ?, ?)$ from P_2 and add it to P_1.
   In the example, P_1 and P_2 change as follows: $P_1 = \{1543, (q', ?, ?)\}, P_2 = \{95\}$.

Clearly, $P_1 = \{1543, (q', ?, ?)\}, P_2 = \{95\}$ describes the configuration $(q', 1, 1)$.

Note that the rules above are formalised by means of conjunctions. For example, and instance of the rule 1 is formalised as $P_1((q,?,?)) \wedge \textsc{Empty}(P_2) \wedge \neg P_2((q,?,\top)) \wedge P_2'((q,?,\top)) \wedge \neg P_1'((q,?,?)) \wedge \textsc{Insert}(P_2) \wedge \textsc{Delete}(P_1)$. In this case, the conjunct $\textsc{Delete}(P_1)$ assures that $(q,?,?)$ is the only element deleted from P_1, while $\textsc{Insert}(P_2)$ and $\neg P_2((q,?,\top))$ guarantee that $(q,?,\top)$ is the only element added to P_2.

Paths in $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ are such that each 8-th state represents the configuration of $\mathcal{A}$, and the subpath containing every 8-th state of paths of $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ represents computations. So there exists an infinite path in $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ if and only if there is an infinite run of $\mathcal{A}$. □

3.3 Undecidability of model checking $CARL_{\mathcal{D},\emptyset}$-artifacts systems

In the proof of Theorem 15 we use arbitrary many constants. However, this is not the source of undecidability.

Theorem 18. *Model checking $CARL_{\mathcal{D},\emptyset}$-artifact systems against FO $\underline{^{sa}}$ CTL is undecidable.*

Proof. For a given two-counter automaton $\mathcal{A}$, we show how to define $CARL_{\mathcal{D},\emptyset}$-artifact systems $S_{\mathcal{A}}$ and an FO $\underline{^{sa}}$ CTL formula φ such that $S \models \varphi$ iff $\mathcal{A}$ has an infinite run. The proof is similar to the proof of Theorem 15 in case of three relations, except that now we use additional relations to represent the state of $\mathcal{A}$.

More precisely, we define $S^{\mathcal{A}}$ as $\langle \mathcal{D}^{\mathcal{A}}, \mathbb{N}, D_0^{\mathcal{A}}, \mathbb{T}^{\mathcal{A}} \rangle$, where $\mathcal{D} = \{P_1/1, \ldots, P_{|Q|+2}/1\}$, D_0 is such that all P_i are empty except for P_3 which is a singleton relation containing 0.

For each tuple $t = (q_i, \phi_1, \phi_2, q_j, \Delta_1, \Delta_2) \in R$, we define an artifact transition α_t as:
$\neg\textsc{Empty}(P_{i+2}) \wedge \mathit{empty}_{\phi_1}(P_1) \wedge \mathit{empty}_{\phi_2}(P_2) \wedge \neg\textsc{Empty}(P'_{j+2}) \wedge \bigwedge_{k \neq j} \textsc{Empty}(P'_{k+2}) \wedge \mathit{modify}_{\Delta_1}(P_1) \wedge \mathit{modify}_{\Delta_2}(P_2)$

It can be shown that $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ represent the computations of $\mathcal{A}$. □

3.4 Further results

Since there is a translation from $CARL_{\mathcal{D},\mathbb{N}}$ to $FO^{\mathbf{2}}_{\mathcal{D},\mathbb{N}}$, we obtain the following.

Corollary 19. *Model checking $FO^{\mathbf{2}}_{\mathcal{D},\mathbb{N}}$-artifact systems against FO $\underline{^{sa}}$ CTL whose database schema contains two unary symbols is undecidable.*

Corollary 20. *Model checking $FO^{\mathbf{2}}_{\mathcal{D},\emptyset}$-artifact systems against FO $\underline{^{sa}}$ CTL is undecidable.*

It can also be shown that including a single binary relation leads to undecidability. This result can be extended to any relations of arity greater that two.

Theorem 21. *Model checking $FO^{\mathbf{2}}_{\mathcal{D},\mathbb{N}}$-artifact systems whose database contains a single binary relation against FO $\underline{^{sa}}$ CTL is undecidable.*

Proof. We use the constants $1, 2, 3$ to simulate three relations.

As before we define $S^{\mathcal{A}}$ as $\langle \mathcal{D}^{\mathcal{A}}, \mathbb{N}, D_0^{\mathcal{A}}, \mathbb{T}^{\mathcal{A}} \rangle$. This time, we take $\mathcal{D} = \{P_1/2\}$ and D_0 is such that $P_1 = \{(3,0)\}$. We modify the set of artifact transitions from the proof in case of three unary relations in the following manner:

- Replace $P_i(t)$ by $P_1(i,t)$ and $P_i'(t)$ by $P_1'(i,t)$.
- Replace $\textsc{Empty}(P_i)$ by $\forall x \neg P_1(i,x)$.
- Replace $\textsc{Equal}(P_i)$ by $\forall x. P_1'(i,x) \Leftrightarrow P_1(i,x)$.
- Replace $\textsc{Insert}(P_i)$ by $\exists x. \neg P_1(i,x) \wedge P_1'(i,x) \wedge \forall y. y \neq x \Rightarrow (P_1(i,x) \Leftrightarrow P_1'(i,x))$.
- Replace $\textsc{Delete}(P_i)$ by $\exists x. \neg P_1'(i,x) \wedge P_1(i,x) \wedge \forall y. y \neq x \Rightarrow (P_1(i,x) \Leftrightarrow P_1'(i,x))$.

It can be checked that paths in $\mathcal{K}_{\mathcal{S}^{\mathcal{A}}}$ represent the computations of $\mathcal{A}$. So an infinite path exists iff there exists an infinite computation of $\mathcal{A}$. □

4 Model Checking $FO^{*}_{\mathcal{D},\mathbb{N}}$–Artifact Systems with a Single Unary Relation

While the results of the previous section are negative, in the following we identify a fragment for which the model checking problem is decidable.

Theorem 22. *Model checking $FO^{*}_{\mathcal{D},\mathbb{N}}$-artifact systems whose database consists of a single unary relation symbol against FO $\underline{^{sa}}$ CTL is decidable.*

The above identifies a restriction on the database schema, but allows the most general logic here considered to define the transitions of the systems. It is instructive to see that restricting our attention to unary relations still enables us to define interesting classes of systems.

Example 23. *An infinite state counter machine can help organising queues. The task of the machine is to print numbered tickets, display the ticket number on a screen, and remove the number from memory. The machine has three states: ON where it prints tickets and displays numbers, $CLOSING$ where it displays numbers but will not print tickets, and OFF where it remains idle.*

The machine can be described as an artifact system with a database consisting of a single unary relation. We can use the constants 0, 1 and 2 to represent the states ON, $CLOSING$ and OFF, and all other natural numbers to represent tickets. An example of a property that may be expressed in FO $\underline{^{sa}}$ CTL is whether it is always possible to remove all numbers from the machine's memory. It follows from Theorem 22 that checking this specification is decidable.

The rest of this section is organised as follows. Firstly, we investigate the expressive power of the logic $FO^{*}_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$. In the following subsection we define the logic $\mathfrak{Card}$ and we show that any set of artifact transitions defined in $FO^{*}_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$ can be replaced by an equivalent set of formulae from the logic $\mathfrak{Card}$. Then, in Subsection 4.2 we show how to transform an artifact system $\mathcal{S}$ into an infinite Kripke structure $\mathcal{K}'$ and an FO $\underline{^{sa}}$ CTL property φ into a CTL property φ' such that $\mathcal{K}_{\mathcal{S}} \models \varphi$ iff $\mathcal{K}' \models \varphi'$. In Subsection 4.3 we define a pushdown system $\mathcal{P}$ and a CTL* formula φ'' such that $\mathcal{K}' \models \varphi'$ iff $\mathcal{P} \models \varphi''$. We conclude in Subsection 4.4 by giving the algorithm solving the decidability of Theorem 22.

In the following we assume $\mathcal{D} = \{P/1\}$.

4.1 The logic Card

A *grounded literal* over $\mathcal{D}$ is one of the following formulae: $P(c)$, $\neg P(c)$, $P'(c)$, $\neg P'(c)$, where c stands for any constant.

Definition 24. *A formula is called a* Card-transition *if it is of the form $\Upsilon \wedge \bigwedge_{i\in I} \exists_{\gamma_i b_i} x\Phi_i$ where Υ is a conjunction of grounded literals, $\gamma_i \in \{\geq, \leq\}$, b_i are natural numbers (including 0), and Φ_i are a boolean combinations of $P(x)$ and $P'(x)$.*

The formulae of the logic Card *are disjunctions of* Card-transitions.

Intuitively, $\exists_{\leq b} x\Phi(x)$ represents the fact that there are at most b distinct elements satisfying Φ; conversely $\exists_{\geq b} x\Phi(x)$ signifies that there are at least b distinct elements satisfying Φ. The formal semantics of the logic Card is defined by the translation into $FO^*_{\mathcal{D},\mathbb{N}}$ that simply replaces each $\exists_{\geq b}\Phi(x)$ with $\exists x_1, \ldots, x_b. \bigwedge_{i\neq j} x_i \neq x_j \wedge \bigwedge_i \Phi(x_i)$ and $\exists_{\leq b}\Phi(x)$ with $\forall x_1, \ldots, x_{b+1}. \bigwedge_{i\neq j} x_i \neq x_j \Rightarrow \bigvee_i \neg\Phi(x_i)$.

As we prove below, the logics Card and $FO^*_{\mathcal{D},\mathbb{N}}$ have the same expressive power.

Example 25. *Consider the formula $\forall y(y \neq c_1 \Rightarrow (P(y) \Leftrightarrow P'(y)))$. This is equivalent to the disjunction of the following formulae:*

- $P(c_1) \wedge P'(c_1) \wedge \exists_{\leq 0} x(P(x) \Leftrightarrow \neg P'(x))$
- $\neg P(c_1) \wedge \neg P'(c_1) \wedge \exists_{\leq 0} x(P(x) \Leftrightarrow \neg P'(x))$
- $P(c_1) \wedge \neg P'(c_1) \wedge \exists_{=1} x(P(x) \wedge \neg P'(x))$ $\wedge \exists_{\leq 0} x(\neg P(x) \wedge P'(x))$
- $\neg P(c_1) \wedge P'(c_1) \wedge \exists_{\leq 0} x(P(x) \wedge \neg P'(x))$ $\wedge \exists_{=1} x(\neg P(x) \wedge P'(x))$

where $\exists_{=1}\Phi$ stands for $\exists_{\leq 1}\Phi \wedge \exists_{\geq 1}\Phi$.

Lemma 26. *Let Ψ be a $FO^*_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$ sentence. Then, there is a Card formula Ψ' equivalent to Ψ such that the size of Ψ' is exponential in the size of Ψ.*

Proof. The logic $FO^*_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$ is a fragment of first-order logic, so we can use standard tools. The result follows by applying Ehrenfeucht-Fraisse games to the case of unary relations. Indeed, for any property Ψ of $FO^*_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$ there is a number $n \leq |\Psi|$ such that if duplicator wins an n rounds game on the two structures, then both structures satisfy Ψ or none of them does.

Let C be the set of constants from Ψ. Given two structures $\mathcal{A}, \mathcal{B}$ we write $\mathcal{A} \sim_C \mathcal{B}$ if for all constants $c \in C$ we have $P^{\mathcal{A}}(c)$ iff $P^{\mathcal{B}}(c)$ and $P'^{\mathcal{A}}(c)$ iff $P'^{\mathcal{B}}(c)$.

Let $cnt(\Phi(x))$ be the number of elements not from C satisfying $\Phi(x)$ and $cnt_n(\Phi(X))$ be equal to $min(n + 1, cnt(\Phi(x)))$. For each model $\mathcal{A}$, we define $cnt_n(\mathcal{A})$ as a tuple (r, s, t, u) where

$$\begin{aligned} r &= cnt_n(P_{\mathcal{A}}(x) \wedge P'_{\mathcal{A}}(x)) \\ s &= cnt_n(P_{\mathcal{A}}(x) \wedge \neg P'_{\mathcal{A}}(x)) \\ t &= cnt_n(\neg P_{\mathcal{A}}(x) \wedge P'_{\mathcal{A}}(x)) \\ u &= cnt_n(\neg P_{\mathcal{A}}(x) \wedge \neg P'_{\mathcal{A}}(x)) \end{aligned}$$

That is, if $cnt_n(\mathcal{A}) = (1, n + 1, 3, 4)$, then $\mathcal{A}$ contains 1 world not from C satisfying $P_{\mathcal{A}}(x) \wedge P'_{\mathcal{A}}(x)$, more than n worlds not from C satisfying $P_{\mathcal{A}}(x) \wedge \neg P'_{\mathcal{A}}(x)$ and so on.

Let $\mathcal{A}$ and $\mathcal{B}$ be two structures such that $\mathcal{A} \sim_C \mathcal{B}$ and $cnt_n(\mathcal{A}) = cnt_n(\mathcal{B})$. On these structures, duplicator has a winning strategy and therefore Ψ cannot distinguish them.

We construct a Card-transition $\Psi_{A,A',r,s,t,u}$ such that $\mathcal{A}$ satisfies $\Psi_{A,A',r,s,t,u}$ iff $cnt_n(\mathcal{A}) = (r, s, t, u)$ and $\mathcal{A} \sim_C \mathcal{B}$. Let $r' = r + |A \cap (C \setminus A')|$, $s' = s + |(C \setminus A) \cap A'|$, $t' = t + |A \cap A'|$ and $u' = u + |(C \setminus A) \cap (C \setminus A')|$. Let $\Psi_{A,A',r,s,t,u}$ be the conjunction of the following formulae:

1. $\bigwedge_{c\in A} P(c) \wedge \bigwedge_{c\in C\setminus A} \neg P(c)$
2. $\bigwedge_{c\in A'} P'(c) \wedge \bigwedge_{c\in C\setminus A'} \neg P'(c)$
3. $\exists_{\geq r'} x(P(x) \wedge P'(x))$
4. only if $r < n + 1$: $\exists_{\leq r'} x(P(x) \wedge P'(x))$
5. $\exists_{\geq s'} x(P(x) \wedge \neg P'(x))$
6. only if $s < n + 1$: $\exists_{\leq s'} x(P(x) \wedge \neg P'(x))$
7. $\exists_{\geq t'} x(\neg P(x) \wedge P'(x))$
8. only if $t < n + 1$: $\exists_{\leq t'} x(\neg P(x) \wedge P'(x))$
9. $\exists_{\geq u'} x(\neg P(x) \wedge \neg P'(x))$
10. only if $u < n + 1$: $\exists_{\leq u'} x(\neg P(x) \wedge \neg P'(x))$

The formula $\Psi_{A,A',r,s,t,u}$ is indeed a Card-transition.

Let $\mathcal{X}_\Psi$ be the set of all the formulae of the form $\Psi_{A,A',r,s,t,u}$ such that there is a structure that satisfies $\Psi \wedge \Psi_{A,A',r,s,t,u}$, and let Ψ' be the disjunction of all the formulae in $\mathcal{X}_\Psi$. We show that any structure $\mathcal{A}$ satisfies Ψ if and only if it satisfies Ψ'.

Assume that $\mathcal{A}$ satisfies Ψ. Let $A = \{c \in C \mid P^{\mathcal{A}}(c)\}$, $A' = \{c \in C \mid P'^{\mathcal{A}}(c)\}$ and $(r, s, t, u) = cnt_n(\mathcal{A})$. Obviously, $\mathcal{A}$ satisfies $\Psi_{A,A',r,s,t,u} \wedge \Psi$, therefore $\Psi_{A,A',r,s,t,u}$ is in $\mathcal{X}_\Psi$ and $\mathcal{A}$ satisfies Ψ'.

Assume that $\mathcal{A}$ satisfies Ψ'. It follows that $\mathcal{A}$ satisfies $\Psi_{A,A',r,s,t,u}$ for some A, A', r, s, t, u such that there is a structure $\mathcal{B}$ satisfying $\Psi_{A,A',r,s,t,u} \wedge \Psi$. So $\mathcal{A} \sim_C \mathcal{B}$ and $cnt_n(\mathcal{A}) = cnt_n(\mathcal{B})$, and therefore, by the Ehrenfeucht-Fraisse argument presented above, $\mathcal{A}$ satisfies Ψ.

Therefore we have constructed a Card formula Ψ' that is equivalent to Ψ. A quick check shows that the number of elements in $\mathcal{X}_\Psi$ is bounded by $2^{2|C|} \cdot n^4$, which is exponential in size of Ψ. Note that if we consider formulae with a fixed number of constants, then the resulting formula is polynomial in size of Φ. □

The above proof is constructive and can be mechanised. As remarked, the construction gives a Card formula which is exponential in the size of the original $FO^*_{\mathcal{D}\oplus\mathcal{D}',\mathbb{N}}$ formula.

Note that for any set of artifact transitions $\mathbb{T}$ and formulae Ψ, Ψ', the sets $\mathbb{T} \cup \{\Psi, \Psi'\}$ and $\mathbb{T} \cup \{\Psi \vee \Psi'\}$ are equivalent; i.e., they prescribe the same transitions. Therefore, in the rest of this section we assume without loss of generality that all the artifact transitions are Card-transitions.

4.2 Infinite Kripke structures

Let $\mathcal{S} = \langle \mathcal{D}, \mathbb{N}, D_0, \mathbb{T}\rangle$ be a Card-artifact system and let $\mathcal{K}_\mathcal{S} = \langle \Sigma, D_0, \tau\rangle$ be the model of $\mathcal{S}$. Let φ be an FO $\underline{sa}$ CTL formula and $\mathcal{C}_a$ be the set of all constants appearing in $\mathbb{T}$ and φ. We define a Kripke structure $\mathcal{K}'$ and a CTL specification φ' such that $\mathcal{K}_\mathcal{S} \models \varphi$ iff $\mathcal{K}' \models \varphi'$.

Recall that the states of $\mathcal{K}_\mathcal{S}$, the model of $\mathcal{S}$, are in fact instances of P. The aim of this section is to define a Kripke structure $\mathcal{K}$ whose states represent the instances of P in the following way: an instance $D = \{P\}$ is represented by a pair $abs(D) = (P \cap \mathcal{C}_a, |P|)$, where $P \cap \mathcal{C}_a$ is the set of *relevant* constants that are in P and $|P|$ is the number of elements in P.

Let us consider $\mathcal{K}' = \langle \Sigma', s'_0, \tau', \pi' \rangle$ where:

- $\Sigma' = \{abs(s) \mid s \in \Sigma\}$ is the set of abstractions of all states in Σ,
- $s'_0 = abs(D_0)$,
- $\pi' : \Sigma' \to \mathcal{C}_a \cup \{e_0, e_1, \dots\}$ is a labelling such that $\pi((A, n)) = A \cup \{e_0, e_1, \dots, e_n\}$,
- $\tau' = \{(abs(D), abs(D')) \mid \tau(D, D')\}$.

We define a translation $\mathfrak{T}_{CTL}$ from FO^{sa} CTL formulae into CTL formulae. Let Ψ be a $FO^*_{\mathcal{D},\mathbb{N}}$ sentence and Ψ' be an equivalent sentence that is a disjunction of $\mathfrak{C}$ard-transitions. We translate Ψ' into a CTL formula Ψ'' by replacing every $P(c_i)$ with c_i, $\exists_{\geq k} x P(x)$ with e_k, $\exists_{\leq k} x P(x)$ with $\neg e_{k+1}$, $\exists_{\geq k} x \neg P(x)$ with $\top$ (recall that P is always finite), and $\exists_{\leq k} x \neg P(x)$ with $\bot$. Finally, we replace Ψ with Ψ''.

We define $\mathfrak{T}_{CTL}(\psi)$ as the result of applying the above translation to all the $FO^*_{\mathcal{D},\mathbb{N}}$ sentences in ψ, and we define $\varphi' = \mathfrak{T}_{CTL}(\varphi)$.

The following lemma follows from the fact that the logic $\mathfrak{C}$ard cannot distinguish between elements other than constants. So when checking for satisfaction there is no need to keep the precise information about the content of P.

Lemma 27. *For an artifact system S, its model $\mathcal{K}_\mathcal{S}$, an FO^{sa} CTL specification φ, let K' and φ' be the infinite Kripke structure and the CTL specification defined as above. Then, $\mathcal{K}_\mathcal{S} \models \varphi$ iff $\mathcal{K}' \models \varphi'$.*

Proof (sketch). We show by induction that for every subformula ψ of φ and every $D \in \Sigma$, we have $\mathcal{K}_\mathcal{S}, D \models \psi$ iff $\mathcal{K}', abs(D) \models \mathfrak{T}_{CTL}(\psi)$.

Consider the case when $\psi = AX\psi_1$ and let D be any state of Σ. Firstly, assume that $\mathcal{K}', abs(D) \models \mathfrak{T}_{CTL}(\psi)$. Let $D_1, \dots, D_n$ be all successors of D (i.e. $\tau(D, D_i)$ for all i). By the definition of τ', for each i we have $\tau'(abs(D), abs(D_i))$. Since $abs(D)$ satisfies $\mathfrak{T}_{CTL}(AX\psi_1)$, $abs(D_i)$ satisfies ψ_1 and, by inductive assumption, D_i satisfies ψ_1. Therefore, D satisfies ψ.

Assume that $\mathcal{K}_\mathcal{S}, D \models \psi$. Let (A', n') be a successor of $abs(D) = (A, n)$. This means that there are two states E, E' such that $abs(E) = (A, n)$, $abs(E') = (A', n')$, and $\tau(E, E')$. Let $n_c = |P^E \cap P^{E'} \setminus C_a|$ be the number of common elements in the relation P of E and E' except for constants from C_a. Let D' be such that $P^{D'}$ is a sum of A', any n_c elements of D and $n' - n_c - |A'|$ elements that are not in P^D or C_a. Then, for any $\mathfrak{C}$ard-transition Φ defined by using the constants from C_a we have $E, E' \models \Phi$ iff $D, D' \models \Phi$.

Therefore, we have $\tau(D, D')$; so D' satisfies ψ_1 and, by the induction hypothesis, $abs(D') = (A', n')$ satisfies $\mathfrak{T}_{CTL}(\psi_1)$. Since (A', n') was an arbitrary successor of $abs(D)$, we conclude that $\mathcal{K}', abs(D) \models \mathfrak{T}_{CTL}(\psi)$.

The remaining cases are similar and omitted. □

Finite representation. For convenience above we assumed that the labelling π' has an infinite codomain. However, φ' uses only finitely many propositional variables. In the rest of this section we assume that the codomain of π' is $\mathcal{C}_a \cup \{e_0, \dots, e_l\}$, where l is the greatest index such that e_l appears in φ'.

While the relation τ' is infinite, we can give a finite description for it. To do this, let b be the smallest natural number greater than all the numbers that appear in $\mathfrak{C}$ard-transitions of $\mathcal{S}$. Let $bound_b(n) = max(-b, min(n, b))$ be a function that returns $-b$ if $n < -b$, b is $n > b$ and n otherwise.

Let T be a set of tuples such that $(A, n_b, A', n'_b, \Delta) \in T$ if and only if for some n, n' such that $n_b = bound_b(n)$, $n'_b = bound_b(n')$, $\Delta = bound_b(n' - n)$ and $\tau'((A, n,), (A', n'))$.

This representation is sufficient since τ' is defined by $\mathfrak{C}$ard-transitions that do not distinguish sets larger than b. In other words, given a set T, we can construct τ' as follows: $\tau'((A, n,), (A', n'))$ if and only if $(A, bound_b(n_b), A', bound_b(n'_b), bound_b(n' - n)) \in T$.

Clearly, for any $(A, n_b, A', n'_b, \Delta) \in T$ we have $A, A' \subseteq C_a$, $n, n' \in \{0, \dots, b\}$ and $\Delta \in \{-b, \dots, b\}$, so T is finite.

4.3 Pushdown systems

Definition 28. *A* pushdown system *is a tuple $\mathcal{P} = (\Sigma^\mathcal{P}, q_0^\mathcal{P}, \Gamma^\mathcal{P}, \bot^\mathcal{P}, \tau^\mathcal{P}, \pi^\mathcal{P})$, where $\Sigma^\mathcal{P}$ is a set of states containing $q_0^\mathcal{P}$, $\Gamma^\mathcal{P}$ is a set of stack symbols containing the stack bottom symbol $\bot^\mathcal{P}$, $\Delta^\mathcal{P} \subseteq (\Sigma^\mathcal{P} \times \Gamma^\mathcal{P}) \times (\Sigma^\mathcal{P} \times (\Gamma^\mathcal{P} \setminus \{\bot^\mathcal{P}\})^*)$ is a finite set of transition rules, and $\pi^\mathcal{P} : \Sigma^\mathcal{P} \times \Gamma^\mathcal{P} \to Var$ is a labelling function for some fixed set of propositional variables Var.*

A configuration *of a pushdown system is a pair (s, α), where $s \in \Sigma^\mathcal{P}$ and $\alpha \in \Gamma^{\mathcal{P}*}$ starts from $\bot^\mathcal{P}$ and does not contain $\bot^\mathcal{P}$ at any other position. We define the transition relation $\to_\mathcal{P}$ as follows: $(s, \alpha\gamma) \to_\mathcal{P} (s', \alpha\alpha')$, where $\gamma \in \Gamma^\mathcal{P}$ and $\alpha, \alpha' \in \Gamma^{\mathcal{P}*}$, iff $(s, \gamma, s', \alpha') \in \tau^\mathcal{P}$.*

Runs are defined as usual (Hopcroft and Ullman 1979). It is known that model checking pushdown systems against CTL* is decidable (Bozzelli 2007).

In what follows, given a Kripke model we define a pushdown system whose configurations represent states (A, n) of the Kripke structure by using the system states to encode A and the stack to store n. In doing so we will ensure that the content of the stack is always a prefix of the word $012 \dots (b-1)b^\omega$. This will allow us to check easily the exact value of n if $n < b$. Since we can remove only one element from the stack at a time, we will use a number of auxiliary configurations to simulate a change of state of Kripke structure.

Formally, let $B = \{-b, \dots, b\}$, $B_+ = \{0, \dots, b\}$ and $P_C = \{A \mid A \subseteq C_a\}$. Below we define a pushdown system $\mathcal{P} = (P_C \cup P_C \times B \times B, \emptyset, B_+, 0, \tau'', \pi'')$ and a CTL* formula φ'' s.t. $\mathcal{P} \models \varphi''$ iff $\mathcal{K}' \models \varphi'$.

We represent a configuration of $\mathcal{P}$ of the form (q, α) as $(q, |\alpha| - 1)$. Since $(q, 0)$ is represented by $(q, 0)$, no ambiguity arises.

A configuration of the form (A, n) represents the state (A, n) of $\mathcal{K}'$. A configurations of the form $((A, n, k), m)$ for

$|n| < b$ is an auxiliary configuration that represents an intermediate state such that $min(b, n+m) = k$ and that the state precedes the state $(A, n+m)$ that will appear in $n+1$ steps. Similarly, a configuration of the form $((A, b, k), m)$ represents the situation where the value stored in the stack can be increased by any number greater than or equal to b. Finally, a configuration of the form $((A, -b, k), m)$ encodes the situation where the value stored in the stack can be decreased by any number greater than or equal to b.

Let $add\_one(m) = m.(min(m+1, b))$ where "." stands for concatenation. Formally, τ'' contains the following rules.

- $((A, b, k), m, (A, b, k), add\_one(m))$ for all possible A, k, m.
- $((A, n, k), m, (A, n-1, k), add\_one(m))$ for all $n > 0$ and all possible A, k, m.
- $((A, n, k), m, (A, n+1, k), \emptyset)$ for all $n < 0$ and all possible A, k, m.
- $((A, -b, k), m, (A, -b, k), \emptyset)$ for all possible A, k, m.
- $(A, n_b, (A', n, n'_b), n_b)$ for all $(A, n_b, A', n'_b, n) \in T$.
- $((A, 0, n'_b), n'_b, A, n'_b)$ and all possible A, n'_b.

Example 29. *Assume that* $\tau'((\emptyset, 4), (\{7\}, 3))$ *and* $b = 2$. *The corresponding computation of* $\mathcal{P}$ *is* $(\emptyset, 01222) \rightarrow_{\mathcal{P}} ((\{7\}, -1, 2), 01222) \rightarrow_{\mathcal{P}} ((\{7\}, 0, 2), 0122) \rightarrow_{\mathcal{P}} (\{7\}, 0122)$.

For $\tau'((\{7\}, 3), (\emptyset, 6))$, *the corresponding computation is:* $(\{7\}, 0122) \rightarrow_{\mathcal{P}} ((\emptyset, 2, 2), 0122) \rightarrow_{\mathcal{P}} ((\emptyset, 2, 2), 01222) \rightarrow_{\mathcal{P}} ((\emptyset, 1, 2), 012222) \rightarrow_{\mathcal{P}} ((\emptyset, 0, 2), 0122222) \rightarrow_{\mathcal{P}} (\emptyset, 0122222)$.

Finally, we define π'' as $\pi''((A, n, k), m) = \emptyset$ and $\pi''(A, m) = A \cup \{e_0, \ldots, e_m\}$.

Note that not all runs of $\mathcal{P}$ correspond to runs of $\mathcal{K}'$. For example, for $b = 2$ the run $(\emptyset, 0) \rightarrow_{\mathcal{P}} ((\emptyset, 2, 2), 0) \rightarrow_{\mathcal{P}} ((\emptyset, 2, 2), 01) \rightarrow_{\mathcal{P}} ((\emptyset, 2, 2), 012) \rightarrow_{\mathcal{P}} \ldots$ continuously increases the stack and will never reach a state of the form (A, m). To avoid such runs, we add a fairness constraint by considering only the paths satisfying e_0 infinitely often.

To transform φ' to a CTL* formula φ'', we define a function $\mathfrak{T}^*$ by induction as follows.

$$
\begin{array}{lcl}
\mathfrak{T}^*(p) & = & p, \text{ where } p \text{ is a propositional variable} \\
\mathfrak{T}^*(\neg\psi) & = & \neg\mathfrak{T}^*(\psi) \\
\mathfrak{T}^*(\psi_1 \vee \psi_2) & = & \mathfrak{T}^*(\psi_1) \vee \mathfrak{T}^*(\psi_2) \\
\mathfrak{T}^*(AX\psi) & = & A(GFe_0 \Rightarrow \neg e_0 U(e_0 \wedge \mathfrak{T}^*(\psi))) \\
\mathfrak{T}^*(A\psi_1 \mathcal{U} \psi_2) & = & A(GFe_0 \Rightarrow (e_0 \Rightarrow \mathfrak{T}^*(\psi_1))U \\
 & & (e_0 \wedge \mathfrak{T}^*(\psi_2))) \\
\mathfrak{T}^*(E\psi_1 \mathcal{U} \psi_2) & = & E(GFe_0 \wedge (e_0 \Rightarrow \mathfrak{T}^*(\psi_1))U \\
 & & (e_0 \wedge \mathfrak{T}^*(\psi_2)))
\end{array}
$$

Consider $\varphi'' = \mathfrak{T}^*(\varphi')$. We have the following.

Lemma 30. *For an infinite Kripke structure* K' *and a CTL specification* φ', *let* P *and* φ'' *be the pushdown system and the CTL** *specification defined as above. Then,* $\mathcal{K}' \models \varphi'$ *iff* $\mathcal{P} \models \varphi''$.

The proof of the above lemma follows from the construction of $\mathcal{P}$ and φ''.

4.4 Model checking

In view of the translation defined above we can now give the algorithm that establishes Theorem 22.

Algorithm `Input`: $FO^*_{\mathcal{D},\mathbb{N}}$-artifact system $\mathcal{S}$ built on a unary relation, FO $\underline{sa}$ CTL formula φ.
`Output`: TRUE or FALSE.

1. Transform the set of artifact transitions to the set of €ard-transitions as described in Subsection 4.1.
2. Generate a finite representation of the Kripke structure $\mathcal{K}'$ and a CTL formula φ' as described in Subsection 4.2.
3. Generate a pushdown system $\mathcal{P}$ and a CTL* formula φ'' as defined in Subsection 4.3.
4. Return the result of model checking $\mathcal{P}$ against φ''.

The Algorithm above gives the proof of Theorem 22. Notice that in the last step we employ the algorithm for model checking the whole of CTL*; so it can also be applied against sentence atomic FO-CTL*.

An upper bound for the model checking problem of Theorem 22 can be established by observing that model checking pushdown systems against CTL* is 2EXPTIME-complete (Bozzelli 2007) and it is applied to a pushdown system exponential in the size of the input.

Corollary 31. *The problem of model checking artifact systems whose database consists of a single unary relation against sentence-atomic FO-CTL** *is in* 3EXPTIME.

5 Conclusions

Artifact-centric systems have been put been forward as a new paradigm in data-aware services and they are increasingly being used as a user-oriented architecture upon which services can be deployed. The issue of verification of artifact-centric systems remains of paramount importance both at theoretical and at practical level.

A number of classes of artifact-centric systems have been proposed for which the verification problem is decidable. As discussed in the related work part of the introduction, a key requirement made of all these system is boundedness of the systems' executions. While it has been shown this can be imposed on the system at program level (Belardinelli, Lomuscio, and Patrizi 2011), there remain classes of scenarios whereby no bound can in principle be set either on the states or the runs of the system.

To explore the implications of this we have here addressed the verification of unbounded artifact-centric systems against sentence-atomic FO-CTL specifications. Our results show that not only the problem is undecidable in general, but that it remains so even when admitting only the simple artifact-centric descriptions built on the logics $CARL_{\mathcal{D},\mathbb{N}}$ and $CARL_{\mathcal{D},\emptyset}$.

While this points to a bleak picture for the verification of unbounded systems in general terms, we were able to identify a class of unbounded systems for which verification against sentence-atomic FO-CTL is decidable. This class is in a way "maximal" with respect to the relations in the database schema.

References

Alonso, G.; Casati, F.; Kuno, H. A.; and Machiraju, V. 2004. *Web Services - Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer.

Belardinelli, F., and Lomuscio, A. 2013. Decidability of model checking non-uniform artifact-centric quantified interpreted systems. In *Proceedings of the 23rd International Joint Conference on Artificail Intelligence (IJCAI'13)*, 725–731. AAAI Press.

Belardinelli, F.; Lomuscio, A.; and Patrizi, F. 2011. Verification of Deployed Artifact Systems via Data Abstraction. In *Proceedings of the 9th International Conference on Service-Oriented Computing (ICSOC'11)*, volume 7200 of *Lecture Notes in Computer Science*, 142–156. Springer.

Belardinelli, F.; Lomuscio, A.; and Patrizi, F. 2012a. An abstraction technique for the verification of artifact-centric systems. In *Proceedings of the 13th International Conference Principles of Knowledge Representation and Reasoning (KR12)*, 319–328. AAAI Press.

Belardinelli, F.; Lomuscio, A.; and Patrizi, F. 2012b. Verification of gsm-based artifact-centric systems through finite abstraction. In *Proceedings of the 10th International Conference on Service Oriented Computing (ISCOC'12)*, volume 7636 of *Lecture Notes in Computer Science*, 17–31. Springer.

Bozzelli, L. 2007. Complexity results on branching-time pushdown model checking. *Theoretical computer science* 379(1):286–297.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 1999. *Model Checking*. Cambridge, Massachusetts: The MIT Press.

Cohn, D., and Hull, R. 2009. Business Artifacts: A Data-Centric Approach to Modeling Business Operations and Processes. *IEEE Data Engineering Bulletin* 32(3):3–9.

Deutsch, A.; Hull, R.; Patrizi, F.; and Vianu, V. 2009. Automatic verification of data-centric business processes. In *Proceedings of the 12th International Conference on Database Theory (ICDT'09)*, volume 361 of *ACM International Conference Proceeding Series*, 252–267. ACM.

Gonzalez, P.; Griesmayer, A.; and Lomuscio, A. 2012. Verifying GSM-based business artifacts. In *Proceedings of the 19th International Conference on Web Services (ICWS'12)*, 25–32. IEEE Press.

Grädel, E.; Kolaitis, P. G.; and Vardi, M. Y. 1997. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic* 3(1):53–69.

Grädel, E. 1999. On the restraining power of guards. *Journal of Symbolic Logic* 64(4):1719–1742.

Hariri, B. B.; Calvanese, D.; Giacomo, G. D.; Masellis, R. D.; and Felli, P. 2011. Foundations of relational artifacts verification. In *Proceedings of the 9th Int. Conference on Business Process Management (BPM'11)*, volume 6896 of *Lecture Notes in Computer Science*, 379–395. Springer.

Hariri, B. B.; Calvanese, D.; Montali, M.; Giacomo, G. D.; and Deutsch, A. 2013. Verification of relational data-centric dynamic systems with external services. In *Proceedings of the 32nd Symposium on Principles of Database Systems (PODS'13)*, 163–174. ACM.

Heath III, F. T.; Boaz, D.; Gupta, M.; Vaculín, R.; Sun, Y.; Hull, R.; and Limonad, L. 2013. Barcelona: A design and runtime environment for declarative artifact-centric BPM. In *Proceedings of the 11th International Conference on Service-Oriented Computing (ICSOC13)*, volume 8274 of *Lecture Notes in Computer Science*, 705–709. Springer.

Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Adison-Wesley Publishing Company.

Hull, R.; Damaggio, E.; De Masellis, R.; Fournier, F.; Gupta, M.; Heath, III, F. T.; Hobson, S.; Linehan, M.; Maradugu, S.; Nigam, A.; Sukaviriya, P. N.; and Vaculin, R. 2011. Business Artifacts with Guard-Stage-Milestone Lifecycles: Managing Artifact Interactions with Conditions and Events. In *Proceedings of the 5th ACM International Conference on Distributed Event-Based Systems (DEBS'11)*, 51–62. ACM.

Hull, R. 2008. Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In *Proceedings of the 2008 Confederated International Conferences CoopIS, DOA, GADA, IS, and ODBASE*, volume 5332 of *Lecture Notes in Computer Science*, 1152–1163. Springer.

Minsky, M. 1967. *Computation: finite and infinite machines*. Prentice-Hall.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Forgetting in Action

David Rajaratnam[1], Hector J. Levesque[2], Maurice Pagnucco[1], Michael Thielscher[1]

[1]University of New South Wales
Sydney, Australia
{*daver,morri,mit*}*@cse.unsw.edu.au*

[2]University of Toronto
Toronto, Canada
hector@cs.toronto.edu

Abstract

In this paper we develop a general framework that allows for both knowledge acquisition and forgetting in the Situation Calculus. Based on the Scherl and Levesque (Scherl and Levesque 1993) possible worlds approach to knowledge in the Situation Calculus, we allow for both sensing as well as explicit forgetting actions. This model of forgetting is then compared to existing frameworks. In particular we show that forgetting is well-behaved with respect to the contraction operator of the well-known AGM theory of belief revision (Alchourrón, Gärdenfors, and Makinson 1985) but that knowledge forgetting is distinct from the more commonly known notion of logical forgetting (Lin and Reiter 1994).

Introduction

Typical Situation Calculus models do not consider the need for agents to forget knowledge. This reflects the types of applications to which formal models of agent action and behaviour have typically been applied. However, it is possible to identify classes of applications for which the need to model forgetting in an agent becomes crucial. Consequently, in this paper we develop a general model for forgetting acquired knowledge in the Situation Calculus. We then highlight useful sub-classes of this model.

The focus of this paper is to develop the theoretical machinery of agent forgetting. However, in order to motivate the importance of forgetting, we highlight two broad classes of potential applications: the modelling of bounded agents and the formal analysis of security protocols.

The most immediate application of forgetting is to model agents with limited resources (e.g., robots), or agents that need to deal with vast knowledge bases (e.g., cloud computing), or more ambitiously, dealing with the problem of lifelong learning. In all such cases it is no longer reasonable to assume that all knowledge acquired over the operation of an agent can be retained indefinitely. Furthermore, recent research on bounded Situation Calculus theories (Giacomo, Lespérance, and Patrizi 2012; 2013) shows that ensuring bounds on such theories can guarantee decidability of progression. The theory of forgetting we introduce here can be used to guarantee bounded Situation Calculus theories.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

A further motivation for the concept of forgetting is evident in the formal analysis of security and cryptographic protocols. Cryptographic protocols have previously been analysed using the Situation Calculus equipped with a notion of the knowledge of agents (Delgrande, Hunter, and Grote 2010). However, such an encoding cannot always represent the class of *nonmonotonic cryptographic protocols* (Rubin and Honeyman 1994), where a notion of forgetting can be important. For example, in the analysis of credit card protocols it is a critical requirement to model vendors that forget (i.e., not retain) customer credit card details.

In order to illustrate our approach to forgetting we consider the following running example. *A robot needs to enter a room that is protected by a closed door with a keypad lock. When the robot senses that the door is closed, it needs to download the key combination from an external data source (e.g., the cloud or a database). It can then use this key combination to open the door and enter the room. Furthermore, since the door will now be open, the robot no longer needs the key combination, so is free to forget this information.*

The rest of the paper proceeds as follows. First we introduce the Situation Calculus (McCarthy 1963; Reiter 2001) and its epistemic extension (Lesperance et al. 1995) that allows for knowledge acquisition without forgetting. We then present our approach that handles both knowledge acquisition and forgetting and perform an extensive analysis of its properties and the conditions under which both knowledge acquisition and forgetting can occur. Having established our approach to forgetting, we then place it within the broader context of two of the main models that have been developed within the literature: *AGM belief revision* (Alchourrón, Gärdenfors, and Makinson 1985) and *logical forgetting* (Lin and Reiter 1994). In particular, we show that our approach is well-behaved with respect to the AGM belief contraction postulates, but is distinct from that of logical forgetting since knowledge forgetting can provide for more fine-grained control over what is forgotten. Finally, we provide some concluding remarks and discuss directions for future research.

Background

Situation Calculus

The Situation Calculus provides a formal language based on classical first-order logic to describe dynamic domains (Mc-

Carthy 1963; Reiter 2001). It distinguishes three types of terms: *situations* representing histories as the world evolves; *fluents* denoting domain properties that may change as a result of actions; and, *actions* that can be performed by the reasoner.

The function $do(a, s)$ represents the situation that results from performing action a at situation s, while S_0 denotes the initial situation where no actions have taken place. For each action, a *precondition axiom* $Poss(a, s)$ specifies the conditions under which action a is possible in situation s and *successor state axioms* specify how the value of fluents change as the result of actions. Assuming perfect knowledge and no sensing actions, the precondition and successor state axioms for our example scenario can be encoded as follows:

$$\begin{array}{lcl} Poss(enter, s) & \equiv & Open(s) \wedge \neg InRoom(s) \\ Poss(usekey, s) & \equiv & \exists c.Key(c, s) \wedge \neg Open(s) \\ InRoom(do(a, s)) & \equiv & InRoom(s) \vee a = enter \\ Key(c, do(a, s)) & \equiv & Key(c, s) \\ Open(do(a, s)) & \equiv & Open(s) \vee a = usekey \end{array}$$

The first states that it is possible to enter the room if the robot is not already there and the door is open. The second states that it is possible to use the key code to open the door if there is a key code and the door is not already open. Finally, three successor state axioms are defined to indicate that: the robot will be in the room if it just entered it or was already there; key codes for the door do not change; and, the door will be open if it was already opened or the key code was entered.

With this formalisation it is possible to follow how the world evolves as actions are performed and, in particular, that the robot will be in the room if it inputs the key code, thus opening the door, and then enters the room.

It is useful to be able to refer to the relationship between situations resulting from a sequence of actions. In order to do this a partial order relation $<$ is defined over situations in order to indicate that one situation is the result of performing a sequence of actions in another situation:

$$\neg s < S_0, \quad s < do(a, s') \equiv s \leq s'$$

where $s \leq s'$ is shorthand for $s < s' \vee s = s'$. For a more comprehensive formulation of what is required of a Situation Calculus basic action theory, we refer to (Reiter 2001).

Situation Calculus with Knowledge

In its basic form the Situation Calculus can be used to show how the world changes in the face of actions. However, it lacks the ability to model how an agent with imperfect knowledge can both gain knowledge and act on that knowledge. To deal with this broader problem the Situation Calculus has been extended using a possible worlds framework to capture a notion of knowledge with sensing actions (Scherl and Levesque 1993). As we shall be drawing repeated comparisons between this extension and our own approach we shall refer to this extension as the *SL framework*.

Within the SL framework, an agent can be said to *know* that some fact ϕ is true if and only if ϕ is true in all possible states of the world that the agent can be in. This can be formalised using a special epistemic predicate $K(s', s)$ to represent the fact that in situation s the agent considers the world could equally be in situation s'. Situation s' is said to be *accessible* from s. Notions of knowledge are defined in terms of this epistemic relation:[1]

$$\begin{array}{l} \mathbf{Know}(\phi, s) \stackrel{\text{def}}{=} \forall s'.K(s', s) \supset \phi[s'] \\ \mathbf{KnowIf}(\phi, s) \stackrel{\text{def}}{=} \mathbf{Know}(\phi, s) \vee \mathbf{Know}(\neg\phi, s) \end{array}$$

The first definition states that an agent in situation s *knows* that fluent ϕ holds when ϕ holds in all *K* accessible situations. The second states that to *know if* something holds means to either know that it holds or to know that it doesn't hold. The predicate *K* is required to be reflexive, transitive and Euclidean, ensuring that the agent has introspection about its knowledge; it knows whether it knows something.

To allow for introspective statements, that is, to represent that an agent knows that it knows something, **Know** can be treated as a term when used in conjunction with the pseudo-variable *now* (Lesperance et al. 1995). This usage is made clear from the following expansion:

$$\mathbf{Know}(\mathbf{Know}(\phi, now), s) \equiv \forall s'.K(s', s) \supset \mathbf{Know}(\phi, s')$$

Now, reasoning about knowledge is not very interesting unless that knowledge can itself change over time and sensing actions provide a mechanism to achieve this. As with non-sensing actions, sensing actions require precondition axioms to determine when they are executable. For our example scenario we introduce two sensing actions *isopen* and *getkey* that are always executable. The first senses if the door is open while the second models the process of querying a data source for the correct key combination:

$$\begin{array}{lcl} Poss(getkey, s) & \equiv & True \\ Poss(isopen, s) & \equiv & True \end{array}$$

Next, the process of knowledge acquisition is encoded through the successor state axiom of the *K* relation. An example of such a successor state axiom, containing sensing actions for our robot scenario is as follows:

$$\begin{array}{l} K(s^*, do(a, s)) \equiv \exists s'.s^* = do(a, s') \wedge Poss(a, s') \wedge K(s', s) \\ \qquad \wedge a = isopen \supset [Open(s') \equiv Open(s)] \\ \qquad \wedge a = getkey \supset [\forall k.Key(k, s') \equiv Key(k, s)] \end{array}$$

For non-sensing actions, the accessible situations are simply the successors of all the situations that were previously accessible. However, for sensing actions only those situations that agree with the actual situation on the sensed fluent remain accessible. So, in the example scenario, the *isopen* sensing action ensures that only situations where the state of the door is the same are accessible, while the *getkey* action ensures that only the situations where all the keys are the same are accessible. Effectively, after performing the *isopen* action, the agent will know if the door is open and after the *getkey* action the agent will know about any keys.

The final component of the representation is to specify the truth of fluents in the initial situation and their possible

[1] $\phi[s]$ is the commonly used notation to represent that formula ϕ holds in the situation s; we use the name **KnowIf** rather than **KnowWhether** introduced in (Scherl and Levesque 1993).

alternatives. For example, that the door is closed but the robot doesn't know it and the door's key combination is C_1 but the robot only knows that it must be one of C_1 or C_2. Figure 1(a) shows a visualisation of this scenario and the accessibility of the alternative possible situations as the robot executes the sequence of actions resulting in it entering the room. Note, for simplicity and presentation we have only specified two possible keys but, of course, this could be extended to many key combinations through the specification of more alternative initial situations.

There is an important observation that can be made from the inclusion of the *K* relation and sensing actions. Namely, that it is possible to construct cases of hybrid sensing actions that both sense and change the environment by allowing sensing actions to also be contained in the non-*K* (i.e., fluent) successor state axioms. In such cases knowledge acquisition cannot be guaranteed because the sensing action could itself change the value of the sensed fluent. To avoid such pathological cases it is typical to consider only *pure* sensing actions where the sensing action only occurs in the successor state axiom for *K*.

A General Account of Knowledge and Forgetting

In this section we present our extension to the SL framework by incorporating the ability to forget the results of past sensing actions. The possible worlds approach is maintained, however we provide a simpler construction in terms of a possible situations relation, $W(s, s')$, which is then used to derive more complex epistemic properties.

Axiom 1 *Let* W *be a relation that is reflexive, transitive and Euclidean. The* possible worlds successor state axiom *for* W *is:*

$$\mathrm{W}(s^*, \mathrm{do}(a, s)) \equiv \exists s'.(s^* = \mathrm{do}(a, s') \wedge \mathrm{W}(s', s) \wedge \mathit{Poss}(a, s'))$$

This successor state axiom works in a similar manner to the original axiom for *K* in the sense that it tracks the evolution of possible situations as a result of actions. However, unlike the original *K* relation, *W* makes no reference to how the agent's knowledge changes as a result of these actions.

To account for sensing actions we first introduce the notion of a *sensing equivalence* relation *SEQ* as a means of separating the generic aspects of the formalism from those that are specific to a particular scenario:[2]

Definition 1 *Consider a Situation Calculus basic action theory* Σ*, a set of sensing actions* $a_1, \ldots, a_n$ *and corresponding sensed fluents* $\phi_1, \ldots, \phi_n$*. Then* $\mathrm{SEQ}(a, s', s)$ *is a* sensing equivalence *relation iff there exists an axiom in* Σ *of the form:*

$$\mathrm{SEQ}(a, s', s) \equiv \bigwedge_{i=1}^{n} a = a_i \supset [\nabla_i.(\phi_i[s'] \equiv \phi_i[s])],$$

where each ∇_i *is a composition of first-order quantifiers over the non-situation free variables in* ϕ_i*.*

[2]Later versions of the SL framework use similar approaches, introducing a sensing result function (Scherl and Levesque 2003) or sensing fluent (Levesque 1996).

In the definition of sensing equivalence each of the sensing actions a_i senses whether a formula ϕ_i is true or whether it is false in a given situation. For brevity we simply say that a_i *senses* ϕ_i. The equivalence specified by an *SEQ* relation is problem specific and encodes the knowledge that is acquired by the sensing actions. For the example scenario we can provide the following sensing equivalence relation:

$$\begin{aligned}\mathit{SEQ}(a, s', s) \equiv\ & (a = \mathit{isopen} \supset [\mathit{Open}(s') \equiv \mathit{Open}(s)]) \wedge \\ & (a = \mathit{getkey} \supset [\forall k.\mathit{Key}(k, s') \equiv \mathit{Key}(k, s)])\end{aligned}$$

The *W* relation and successor state axiom in combination with the sensing equivalence relation can now be used to both capture the original Scherl and Levesque notion of knowledge as well as define a more general notion of knowledge that allows for forgetting actions.

Definition 2 *Consider a Situation Calculus basic action theory, extended with the successor state axiom for* W *and a sensing equivalence relation* SEQ*. Then* $\mathrm{K_M}$ *is a* knowledge acquisition relation *defined as:*

$$\begin{aligned}\mathrm{K_M}(s', s) \stackrel{\mathrm{def}}{=}\ & \mathrm{W}(s', s) \wedge \\ & [\neg\exists \hat{s}', \hat{s}, a.\mathrm{W}(\hat{s}', \hat{s}) \wedge \neg\mathrm{SEQ}(a, \hat{s}', \hat{s}) \wedge \\ & \quad \mathrm{do}(a, \hat{s}') \leq s' \wedge \mathrm{do}(a, \hat{s}) \leq s]\end{aligned}$$

Essentially, this definition states that every *W* relation is also a K_M relation unless it has been *blocked* by a sensing action for which the two situations disagree with respect to the sensed fluents. The definition of **Know** would then be defined in terms of K_M in the standard manner outlined earlier. For space reasons we do not prove this here but it is straightforward to observe that the original notion of agent knowledge is exactly captured by this new formulation.

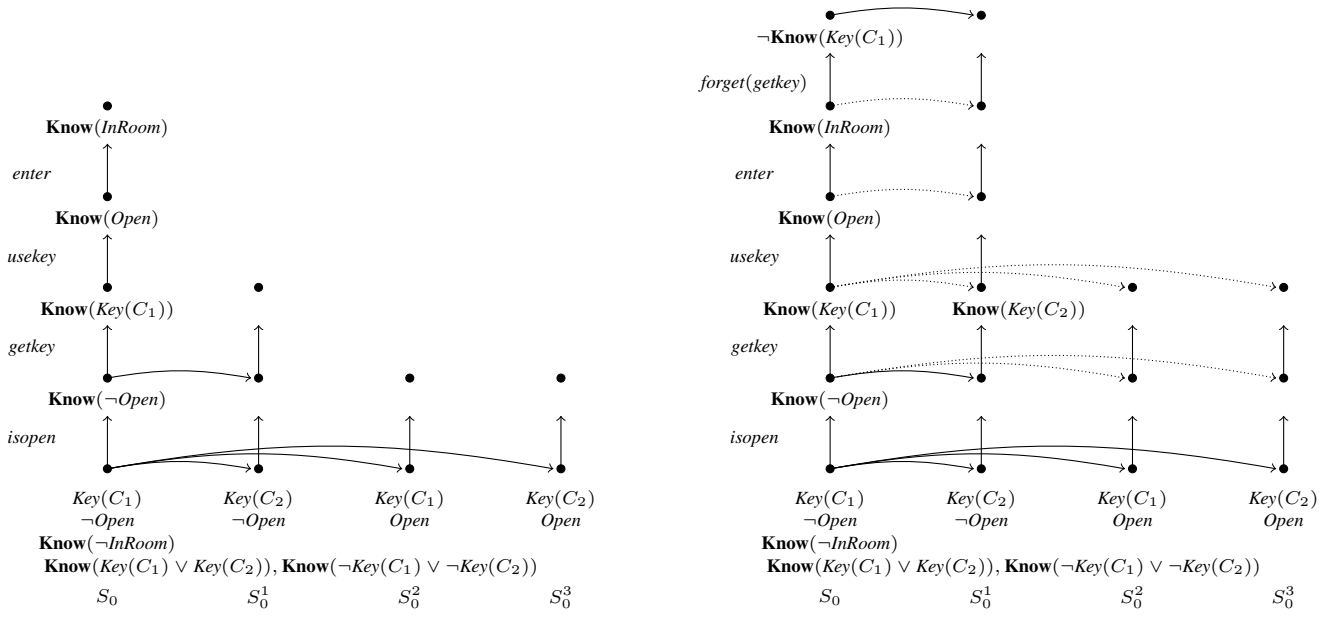
However, we are interested in defining a more general notion of knowledge. This is possible with the help of an explicit forgetting action *forget* that enables an agent to forget previously acquired knowledge.

Definition 3 *Consider a Situation Calculus basic action theory extended with the successor state axiom for* W *and a sensing equivalence relation* SEQ*. Then* K *is a* general knowledge relation *defined as:*

$$\begin{aligned}\mathrm{K}(s', s) \stackrel{\mathrm{def}}{=}\ & \mathrm{W}(s', s) \wedge \\ & [\neg\exists \hat{s}', \hat{s}, a.\mathrm{W}(\hat{s}', \hat{s}) \wedge \neg\mathrm{SEQ}(a, \hat{s}', \hat{s}) \wedge \\ & \quad \mathrm{do}(a, \hat{s}') \leq s' \wedge do(a, \hat{s}) \leq s \wedge \\ & \quad \{\neg\exists s^{*\prime}, s^*, b.b = \mathrm{forget}(a) \wedge \\ & \qquad \mathrm{do}(a, \hat{s}') < \mathrm{do}(b, s^{*\prime}) \leq s' \wedge \\ & \qquad \mathrm{do}(a, \hat{s}) < \mathrm{do}(b, s^*) \leq s\}]\end{aligned}$$

Again, here **Know** is defined in the usual manner. *K* works in a similar manner to K_M in that it allows sensing actions to act as a block on the accessible situations. However, it further allows this block to itself be blocked by a forgetting action, thus re-instating a *K* accessible situation that might otherwise be inaccessible. In effect the agent will have forgotten what it had previously learned by sensing.

It is important to note that the single parameter of the forgetting action is itself an action, with the only useful case being that of a sensing action. Consequently, this construction formalises the notion of an agent that forgets the epistemic commitments of a previous sensing action rather than directly forgetting the value of a specified fluent.



(a) Knowledge in the SL Framework

(b) Knowledge with Forgetting

Figure 1: The progression of actions leading to the robot being in the room, and in (b) forgetting the key combination. Vertical lines indicate actions that are performed from the accessible situations. Solid curved horizontal lines represent *K* accessibility from the actual situation S_0 to the alternative possible situations, while in (b) dotted lines represent *W* only accessibility.

Furthermore, the agent retains the knowledge that the sensing action itself did in fact take place by virtue of the action history encoded in the situation. However, it simply cannot remember the result of that action. We can understand this behaviour as the distinction between *semantic* and *episodic* memory (see, for example, (Schacter, Gilbert, and Wegner 2011)). Specifically, we are concerned with forgetting semantic memory. Forgetting of episodic memory corresponds to the robot forgetting some of its past actions which can be achieved in a Situation Calculus context through the use of *progression* (Reiter 2001).

In order to simplify the encoding of knowledge acquisition and forgetting as part of a Situation Calculus basic action theory we introduce some restrictions as follows:

Definition 4 A Situation Calculus Epistemic Basic Action Theory (EBAT) *is a Situation Calculus Basic Action Theory* Σ *extended with the successor state axiom for* W *such that:*

1. *A precondition axiom for forgetting actions:* $Poss(\text{forget}(a), s) \equiv True$*;*
2. *The successor state axioms of* Σ *do not contain a reference to the function symbol* forget*;*
3. *The sensing equivalence relation (*SEQ*) does not contain a reference to the function symbol* forget*;*
4. *All sensing actions are* pure*; and,*
5. *Along with the other standard basic action theory axioms for the Situation Calculus extended with knowledge (see (Scherl and Levesque 2003)).*

These restrictions result in three distinct and non-overlapping sets of actions: sensing actions, forgetting actions and actions that change the world. As they are concerned with the agent's knowledge, where convenient we refer to the sensing and forgetting actions as *epistemic actions*.

The requirement for forgetting actions to always be executable and not mentioned in the successor state axioms is not strictly necessary but simplifies the formalisation. In particular, not allowing forgetting actions to be contained in either successor state axioms or the sensing equivalence relation ensures that forgetting actions serve only one purpose, namely to forget, and cannot be misappropriated for multiple purposes. Of course, one can always simulate such multi-purpose actions if necessary by introducing individual actions that must be executed in a block. Clearly these restrictions are not onerous.

Framework Analysis and Properties

We now turn to analysing the properties of our framework. In particular, we establish intuitive knowledge theorems for the framework. This does however require certain restrictions on the formalisation of a given scenario, the reasons for which require some explanation.

Limits of the Possible Worlds Approach

Contingent facts Firstly, it is important to observe that our framework inherits the limitations of the SL framework in that the set of initial situations establishes the boundaries over what knowledge about the world is subject to change and what must remain either known or unknown throughout. We refer to such changeable knowledge as *contingent* and

define this notion in relation to individual situations:

$$\textbf{Contingent}(\phi, s) \stackrel{\text{def}}{=} (\exists s'.W(s', s) \wedge \phi[s']) \wedge (\exists s'.W(s', s) \wedge \neg\phi[s'])$$

This establishes that a proposition is contingent in a situation if there is at least one related possible situation in which the proposition is true and another in which it is false. For completeness we say that a proposition is *necessarily* the case in a situation if it is not contingent in that situation.

The contingent facts are precisely those facts that are potentially discoverable about the world and therefore the target for sensing actions. However, it is important to observe that the set of contingent facts can vary as situations evolve. For example, if a door being open is contingent in some initial situation but an agent subsequently senses that the door is open and walks through it, then in the resulting situation the state of the door will no longer be contingent because there cannot be a possible world in which the agent walked through the closed door. Less obviously, a fact that was previously necessarily the case can become contingent after performing some action. For example, the action of replicating the truth value of a contingent fluent.

Knowledge acquisition without sensing While sensing is the primary means by which an agent can acquire knowledge, the theoretical construction of the framework masks a second method by which knowledge can be acquired without explicit sensing actions. This is a property that is implicit in the original SL framework as well as its various extensions, such as that of Shapiro *et al.* (Shapiro et al. 2011).

In essence the problem stems from how the precondition axioms are encoded and whether the representation of these axioms is intended to model the actions that an agent knows that it can execute, or the executability of actions irrespective of the agent's knowledge of whether those actions are executable. To clarify this problem, consider the scenario where the robot does not initially know if the door is open. Now, if the door happens to be open in the actual situation, then a precondition axiom defined without reference to the robot's knowledge allows it to walk through the door without knowing that it is open. However, in the alternative possible situation where the door was closed, this action was not possible, hence there will be no successor situation corresponding to the situation where the door is closed. Consequently, by definition the robot will "know" that the door is closed, despite never having sensed this about its environment.

At first blush this may not appear to be problematic. One might, for example, argue that if a person has their eyes closed and successfully walks through a door then they will know that the door must have been open in order for the action to have succeeded. However, such reasoning ignores the many sensing actions that would be taking place in such a scenario. Namely, that the person is able to know that the action was successful because they didn't feel the pain of hitting the door and finding themselves on the floor.

This problem can in practice be avoided with some added theoretical machinery. In particular, in order to formalise reasoning about agent *abilities*, the SL framework has been extended with the notion of an *action selection function* (Lesperance et al. 1995). This is a second-order construct that allows conditions to be defined such that an agent *can know whether or not it can get* to a situation where some property holds. Provided that one only considers sequences of actions where the agent knows that it can get to some state, then the problem of knowledge acquisition without sensing can be avoided.

Nevertheless the issue remains a troubling property of the framework. It is particularly problematic for our extension as it can affect the contingency of facts and consequently the conditions under which forgetting can take place.

Fortunately, the solution to this problem is relatively simple. Namely, to ensure that all action preconditions are specified so as to model the actions that the agent knows that it can execute. We shall refer to these as actions with *known preconditions*. With this in mind, the precondition axioms for the sample scenario can now be re-stated as follows:

$$\mathit{Poss}(\mathit{enter}, s) \equiv \textbf{Know}(\mathit{Open}, s) \wedge \textbf{Know}(\neg \mathit{InRoom}, s)$$
$$\mathit{Poss}(\mathit{usekey}, s) \equiv \exists k.\textbf{Know}(\mathit{Key}(k), s) \wedge \textbf{Know}(\neg \mathit{Open}, s)$$

Now it is possible for the robot to execute the *enter* action only when it knows that it is not currently in the room and knows that the door is open. Similarly, in order for the robot to use the key combination to open the door, it must know the combination and know that the door is closed. Note, actions that are always executable, such as the example sensing actions, implicitly have known preconditions (i.e., a tautology is true in every possible world and hence is known).

Epistemic Properties

We can now turn to establishing that the knowledge with forgetting framework genuinely satisfies the intuitive properties of an epistemic action theory. Namely, we establish that it satisfies the basic introspective properties of knowledge, that knowledge can indeed be acquired through sensing actions and, finally, that knowledge can also be forgotten through forgetting actions.

Introspection As is the case of the SL framework, our extension can be shown to also satisfy a strong notion of knowledge introspection. As a first step, it is useful to establish some key properties of the defined *K* relation.

Lemma 1 *Consider a Situation Calculus EBAT. The defined relation* K *is* reflexive*,* transitive *and* Euclidean*.*

Proof: Reflexivity is immediate since for every $W(s, s)$ relation $K(s, s)$ exists by definition. We now consider only the transitive case as the Euclidean case follows an identical argument. Consider if $K(s, s')$ and $K(s', s'')$ but $\neg K(s, s'')$. W is transitive so as $W(s, s')$ and $W(s', s'')$ therefore $W(s, s'')$. Hence $K(s, s'')$ is being blocked from holding by some sensing action a such that the fluent it senses, ϕ, is different between s and s''. But then the value of ϕ in s' must be different to one of either s or s''. Hence either $K(s, s'')$ or $K(s', s'')$ must be false and a contradiction follows. □

These properties of the *K* relation can now be used to establish that knowledge is indeed introspective.

Theorem 1 (Introspection) *Let Σ be a Situation Calculus EBAT, then for any ϕ and situation s:*

$$\Sigma \models \textbf{Know}(\phi, s) \textit{ iff } \Sigma \models \textbf{Know}(\textbf{Know}(\phi, now), s)$$

Proof: Assume that LHS holds but the RHS doesn't hold. Hence $\forall s'.K(s', s) \supset \phi[s']$ holds but $\forall s'.K(s', s) \supset (\forall s''.K(s'', s') \supset \phi[s''])$ does not. However, the only way for the second formula not to hold is if there is some s' and s'' such that $K(s', s)$ and $K(s'', s')$ and $\neg\phi[s'']$. But, from Lemma 1, K is transitive so $K(s'', s)$ must hold and contradicts with the LHS.

Now assume that the RHS holds but the LHS doesn't. So there exists some s^* such that $K(s^*, s)$ and $\neg\phi[s^*]$. Substituting s^* for an instance of s' in the RHS we have that $\forall s''.K(s^*, s) \supset K(s'', s^*) \supset \phi[s'']$ and since $K(s^*, s)$ holds so $\forall s''.K(s'', s^*) \supset \phi[s'']$. Now K is reflexive so $K(s^*, s^*)$ holds and therefore $\phi[s^*]$ must hold, which contradicts the assumption that the LHS doesn't hold. □

Knowledge Acquisition Naturally, it is important that an agent can actually gain knowledge through sensing actions, namely after sensing a contingent fluent the agent will know the truth value of that fluent.

Theorem 2 *Let Σ be a Situation Calculus EBAT, a_ϕ be a pure sensing action that senses fluent ϕ and is executable in situation s, then:*

1. $\Sigma \models \textbf{Know}(\phi, \text{do}(a_\phi, s))$*, where ϕ is true in s, or*
2. $\Sigma \models \textbf{Know}(\neg\phi, \text{do}(a_\phi, s))$*, otherwise.*

Proof: Case 1. Consider any s' such that $W(s', s)$ and $\neg\phi[s']$. If $Poss(a, s')$, then there will be a $W(do(a_\phi, s'), do(a_\phi, s))$ but a_ϕ detects the value of ϕ therefore $\neg SEQ(a_\phi, s', s)$ holds. Alternatively, $\neg Poss(a, s')$ so there will be no $W(do(a_\phi, s'), do(a_\phi, s))$. In either case $K(do(a_\phi, s'), do(a_\phi, s))$ will not hold. Hence the only situations in which $K(do(a_\phi, s'), do(a_\phi, s))$ holds will be those where the fluent ϕ will also hold, so $\textbf{Know}(\phi, do(a_\phi, s))$. Case 2. follows in an identical manner. □

Together Theorems 1 and 2 establish that the new framework does indeed capture the properties of the SL framework.

Forgetting In a similar manner to the establishment that not all facts are *contingent*, forgetting allows only some facts to be forgotten. We now establish precise conditions under which this can take place.

The simplest case of forgetting is where a forgetting action is performed immediately after its corresponding sensing action. Interestingly, even in this case there are some conditions that are required. The critical property is to preserve the contingency of a fluent after the sensing action. Unfortunately, even for a pure sensing action, contingency is not necessarily preserved, and it is possible to provide a simple example to highlight this fact.

Consider a sensing action a_ϕ that senses ϕ and has the precondition that it can sense ϕ only when ϕ is itself known (i.e., $Poss(a_\phi, s) \equiv \textbf{Know}(\phi, s)$). A set of possible situations can then be defined such that ϕ is both known and contingent. In such a case, after performing the sensing action, ϕ will still be known but will no longer be contingent.

The contrived nature of this example provides some hint that such sensing actions are not the typical types of sensing actions that we would want to consider and therefore we restrict our actions to remove such pathological cases.

Definition 5 *An action a is* contingency preserving *w.r.t. ϕ in s if ϕ is contingent in s iff ϕ is contingent in* $\text{do}(a_\phi, s)$.

In general it is reasonable to expect that an action that senses a fluent should be contingency preserving for that fluent, although not necessarily other fluents. Furthermore, it is straightforward to observe that, because forgetting actions are always possible, therefore they are contingency preserving for all fluents in all situations. We now establish the simplest condition under which forgetting can take place; cases where a fluent is forgotten immediately after sensing.

Lemma 2 *Let Σ be a Situation Calculus EBAT such that fluent ϕ is contingent in s and a_ϕ is a pure sensing action that preserves the contingency of ϕ in s, then:*

$$\Sigma \models \neg\textbf{KnowIf}(\phi, \text{do}([a_\phi, \text{forget}(a_\phi)], s))$$

Proof: Both, $\Sigma \models \neg\textbf{Know}(\phi, do([a_\phi, forget(a_\phi)], s))$ and $\Sigma \models \neg\textbf{Know}(\neg\phi, do([a_\phi, forget(a_\phi)], s))$ must be shown, but the proofs are identical so consider only the first. Similarly, whether ϕ is *true* or *false* in s are mirrored so only consider the first case. ϕ is contingent in s and a_ϕ preserves contingency, therefore after executing a_ϕ, ϕ will be contingent in $do(a_\phi, s)$ although ϕ will also be known (Theorem 2). Hence there is some $\neg SEQ(a, s', do(a_\phi, s))$ that is blocking $K(s', do(a_\phi, s))$ from holding. Forgetting actions preserve contingency so, after performing the forgetting action $forget(a_\phi)$, then any such block will itself be blocked by the forgetting action and therefore there will exist a world s'' such that $K(s'', do([a_\phi, forget(a_\phi)], s))$ and $\neg\phi[s'']$ hence ϕ will not be known. □

Of course, sensing followed immediately by forgetting is not a particularly useful scenario. Therefore we now consider the case where some arbitrary number of actions are executed in between performing the sensing and forgetting actions. However, in order for this to be possible, it is necessary to establish that the actions that occur between a sensing action and its corresponding forgetting action are in some sense independent of the fact being forgotten.

The key point in establishing a notion of the independence of fluents and actions is to observe that the preconditions and successor state axioms of a basic action theory establish dependencies between actions in terms of how they relate to fluents. If the value of some fluent is related to an action, then it may not be possible to forget the value of that fluent without first forgetting the action itself. However, as already outlined, our framework only provides a mechanism to forget the value of fluents and does not provide a mechanism to forget the history of past actions.

To properly capture these notions requires the definition of a number of properties. Firstly, it is important to capture how non-epistemic actions interact with particular fluents.

Definition 6 (Consequential Independence) *For a Situation Calculus EBAT Σ, an action a is said to be* consequentially independent *from ϕ iff:*

$$\Sigma \models \forall s.\phi[s] \equiv \phi[\text{do}(a, s)]$$

This definition captures whether or not an action can change the value of a fluent under some condition. Importantly, it makes no reference to the executability of an action in a situation. Instead, it only examines the effects of actions irrespective of whether some precondition has been satisfied. Hence it is a strong requirement that genuinely establishes that an action can have no direct effect on some property of the environment. For example, that picking up a cup has no direct effect on the state of a light.

As well as consequential independence, it is also necessary to consider the *epistemic independence* of fluents from an agent's actions.

Definition 7 (Epistemic Independence (non-forgetting)) *For a Situation Calculus EBAT Σ, a non-forgetting action a is said to be* epistemically independent *of ϕ in s iff:*

$$(\mathbf{Know}(\phi, s) \equiv \mathbf{Know}(\phi, \mathrm{do}(a, s))) \wedge$$
$$(\mathbf{Contingent}(\phi, s) \equiv \mathbf{Contingent}(\phi, \mathrm{do}(a, s)))$$

This establishes two specific criteria for an action to be considered epistemically independent of a fluent. Most obviously it cannot change the agent's knowledge of that fluent. This applies to directly sensing the fluent but also any indirect epistemic ramifications of the action. For example, sensing that a light is on means knowing that the switch is also turned on. However, equally important is that the action must preserve the contingency of a fluent.

Definition 7 only captures the notion of epistemic independence for non-forgetting actions. However, forgetting actions also affect what an agent knows and it is therefore necessary to extend this notion to forgetting actions.

Definition 8 (Epistemic Independence (forgetting)) *For a Situation Calculus EBAT Σ, a forgetting action* $\mathrm{forget}(a_\phi)$ *is said to be* epistemically independent *from ϕ in s iff a_ϕ is epistemically independent of ϕ in s.*

Simply, a forgetting action is epistemically independent of a fluent in exactly the cases where its corresponding sensing action is epistemically independent of that fluent.

We can now define a general notion of the independence of actions from fluents by grouping together the various specific types of independence.

Definition 9 (Independence) *For a Situation Calculus EBAT Σ, an action a is independent from fluent ϕ in situation s iff a is consequentially independent of ϕ in s and a is epistemically independent (forgetting and non-forgetting) of ϕ in s.*

Now that we are armed with a formal notion of the independence of actions from fluents, we can establish the conditions under which a fluent will be forgettable in a situation.

Definition 10 (Forgettable) *We say that a fluent ϕ is* forgettable *in situation s iff there is a situation s' such that $s = \mathrm{do}([a_\phi, a_1, \ldots, a_n], s')$, where ϕ is contingent in s', a_ϕ is a pure sensing action that senses ϕ and preserves the contingency of ϕ in s', and:*

1. *action a_1 is independent of ϕ in $\mathrm{do}(a_\phi, s')$; and,*
2. *for each $1 \leq i < n$, action a_{i+1} is independent of ϕ in $\mathrm{do}([a_\phi, a_1, \ldots, a_i]), s')$.*

It can now be established that our defined notion of forgettability does indeed capture the right conditions to allow the results of a sensing action to be known in one situation but forgotten in a some later situation.

Theorem 3 *Let Σ be a Situation Calculus EBAT such that fluent ϕ is forgettable in situation s and action a_ϕ senses ϕ, then:*

$$\Sigma \models \neg\mathbf{KnowIf}(\phi, \mathrm{do}(\mathrm{forget}(a_\phi), s))$$

Proof: Expanding, both $\Sigma \models \neg\mathbf{Know}(\phi, do(forget(a_\phi), s))$ and $\Sigma \models \neg\mathbf{Know}(\neg\phi, do(forget(a_\phi), s))$ must be shown, but the proofs are identical so consider only the first case. Similarly, whether ϕ is *true* or *false* in s are mirrored so only consider the first case. Applying the definition of ϕ being forgettable in s we have situation $s = do([a_\phi, a_1, \ldots, a_n], s')$, where ϕ is contingent in s'. Now provable by induction on the size of $a_1, \ldots, a_n$. Base case consider $n = 1$: Similar to Lemma 2. ϕ is known but contingent after action a_ϕ. Now, a_1 is independent of ϕ in $do(a_\phi, s')$ so ϕ is known and contingent in $do([a_\phi, a_1], s')$. Because ϕ is contingent in $do([a_\phi, a_1], s')$ so there is some $W(s'', do([a_\phi, a_1], s'))$ where $\neg\phi[s'']$ holds but is being blocked by some $\neg SEQ(a_\phi, s', s^*)$. Hence after executing $forget(a_\phi)$ this block will itself be blocked so there will be some $K(s''', do([a_\phi, a_1, forget(a_\phi)], s')$ where $\neg\phi[s''']$ so will not be known in $do([a_\phi, a_1, forget(a_\phi)], s')$. Now assume that this holds for $n = i$ and prove for $n = i + 1$. The proof for this is similar to the base case and stems from ϕ being contingent and known in $do([a_\phi, \ldots, a_i], s')$. Since ϕ is independent of a_{i+1} in $do([a_\phi, \ldots, a_i], s')$ so ϕ will also be known and contingent in $do([a_\phi, \ldots, a_{i+1}], s')$. ϕ being unknown in $do([a_\phi, \ldots, a_{i+1}, forget(a_\phi)], s')$ then follows in a straightforward manner. □

Theorem 3 establishes that knowledge can indeed be forgotten; after having been previously acquired through a sensing action.

It is worth observing that a sensing action immediately followed by its corresponding forgetting action results in an agent forgetting both the sensed value and any derived consequences. However, the presence of intermediate actions in between sensing and forgetting can prevent these consequences being lost. For example, if our example scenario (Figure 1(b)) had also included an initially accessible situation where there was no actual key, then after performing all the actions and forgetting the value of the key, the robot would nevertheless have retained the acquired knowledge that a key did in fact exist.

Introspection about Action Consequences A final property that is worth noting concerns the introspective nature of knowledge and what the agent currently knows about what it will know in the future. It is possible to show that in the current situation the agent knows that after sensing a fluent it will know if that fluent is true or false, however it also knows that it will no longer know this fact after subsequently forgetting the sensing action. Due to space restrictions we only state the theorem without proofs.

Theorem 4 *Let Σ be a Situation Calculus EBAT such that ϕ is forgettable in situation s and action a_ϕ senses ϕ, then:*

$$\Sigma \models \begin{array}{l} \mathbf{Know}(\mathbf{KnowIf}(\phi, \mathrm{do}(a_\phi, now)), s) \ \wedge \\ \mathbf{Know}(\neg\mathbf{KnowIf}(\phi, \mathrm{do}([a_\phi, \mathrm{forget}(a_\phi)], now)), s) \end{array}$$

Relationship to Existing Frameworks

In this section we explore the relationship of our framework to well-known formalisms in the literature that provide for similar behaviours.

AGM Belief Contraction

While we are concerned with notions of knowledge, there is nevertheless a close link to the research on belief change. Belief change is concerned with establishing how (rational) agents should change their beliefs in light of new information. The most well-known formal account of belief change is that of AGM belief revision (Alchourrón, Gärdenfors, and Makinson 1985) which defines rationality postulates that should be preserved by belief change operators.

Of particular interest to our account of forgetting is that the effects of a forgetting action are comparable to the AGM belief contraction operator. For reference, the traditional AGM belief contraction postulates are as follows.

(K–1) For any sentence ϕ and any belief set K, $K - \phi$ is a belief set. (**closure**)
(K–2) If $\phi \notin Cn(\emptyset)$, then $\phi \notin K - \phi$. (**success**)
(K–3) $K - \phi \subseteq K$. (**inclusion**)
(K–4) If $\phi \notin K$, then $K - \phi = K$. (**vacuity**)
(K–5) If $\phi \in K$, then $K \subseteq Cn((K - \phi) \cup \{\phi\})$. (**recovery**)
(K–6) if $\phi \leftrightarrow \psi \in Cn(\emptyset)$ then $K - \phi = K - \psi$. (**extensionality**)

In showing the link to belief contraction, it is first necessary to provide a bridge from the AGM framework to ours. Fortunately, we can adopt the approach already outlined in Shapiro *et al.* (Shapiro et al. 2011). In particular Shapiro *et al.* observe that, in the Situation Calculus framework, an agent is said to believe (in our case, know) a formula if the theory entails that the formula is believed (known). However, unless the theory is complete with respect to the beliefs (knowledge) of the agent, then we may not be able to determine whether some formulas are actually believed (known) or not. In contrast, within belief change frameworks, the beliefs of an agent are explicitly associated with a belief set, with the implicit closed-world assumption that those formulas not in the belief set are not believed. To align the two frameworks Shapiro *et al.* therefore assume a model M of the theory Σ which is used to fix the belief state of the agent. We adopt this approach here and assume a model of the theory in order to fix on the *knowledge state* of the agent.

Definition 11 *We denote the* knowledge state *of a situation s (in M) by $\mathbf{K}(s)$ as:*

$$\mathbf{K}(s) = \{\phi : M \models \mathbf{Know}(\phi, s)\}$$

We can now provide a definition of what it means to expand a knowledge state by a formula.

Definition 12 *We denote the expansion of a knowledge state at situation s by ϕ (in M) by:*

$$s + \phi \stackrel{\text{def}}{=} \{\psi : M \models \mathbf{Know}(\phi \supset \psi, s)\}$$

It is worth observing that this notion of knowledge state expansion should not be confused with sensing. Sensing actions sense particular fluents and have precondition axioms that determine when that action is possible. On the other hand, knowledge state expansion simply defines the resulting state when given an initial state and an arbitrary formula that is taken to be known.

The AGM allows for the contraction of any (non-tautological) formula. However, our framework is more selective about what can be forgotten and therefore requires some qualifications.

Definition 13 *We denote the* forgettable *formulas at a situation s by $\mathbf{F}(s)$ as:*

$$\mathbf{F}(s) = \{\phi, \neg\phi : \phi \textit{ is forgettable in } s\}$$

An AGM belief contraction-like operator can now be defined within our extended Situation Calculus framework that specifies the conditions under which contraction is possible.

Definition 14 ($s - \phi$) *We denote the contraction of s by ϕ as $s - \phi$ and define it as follows:*

$$s - \phi \stackrel{\text{def}}{=} \begin{cases} \mathrm{do}(\mathrm{forget}(a_\phi), s), & \textit{if } a_\phi \textit{ senses } \phi \textit{ and} \\ & \phi \textit{ is forgettable in } s; \\ s, & \textit{otherwise.} \end{cases}$$

As a notational convenience we can indicate when two formulas are equivalent with respect to a Situation Calculus basic action theory Σ:

$$\phi \equiv_\Sigma \psi \stackrel{\text{def}}{=} \Sigma \models \forall s.\phi[s] \equiv \psi[s]$$

Finally, we can provide and prove rationality postulates for our contraction operator that mirror the original AGM belief contraction postulates.

Theorem 5 (Contraction) *For any situation s and fluents ϕ and ψ that are forgettable in s:*

(**K**–1) $\mathbf{K}(s - \phi)$ *is deductively closed.* (**closure**)
(**K**–2) $\phi \notin \mathbf{K}(s - \phi)$. (**success**)
(**K**–3) $\mathbf{K}(s - \phi) \subseteq \mathbf{K}(s)$. (**inclusion**)
(**K**–4) *If* $\phi \notin \mathbf{K}(s)$ *and* $\neg\phi \notin \mathbf{K}(s)$ *then* $\mathbf{K}(s - \phi) = s$. (**vacuity**)
(**K**–5) *If* $\phi \in \mathbf{K}(s)$, *then* $\mathbf{K}(s) \subseteq \mathbf{K}(s - \phi) + \phi$ *or* $\mathbf{K}(s) \subseteq \mathbf{K}(s - \phi) + \neg\phi$ (**recovery**)
(**K**–6) *if* $\phi, \psi \in \mathbf{F}(s)$ *and* $\phi \equiv_\Sigma \psi$ *then* $\mathbf{K}(s - \phi) = \mathbf{K}(s - \psi)$. (**extensionality**)

Proof: For (**K**–1) it is straightforward to see that *K* as defined is always deductively closed. (**K**–2) is a direct consequence of Theorem 3. (**K**–3) can be shown by examining the definition of *K* and confirming that what is known after a forgetting action will be a subset of what is known before the forgetting action. (**K**–4) is a direct result of the definition of contraction that only forgettable fluents (or their negations)

can be contracted and that only fluents in the knowledge state can be forgotten. (**K**–5) is a consequence of deductive closure, expansion operator $s + [\neg]\phi$, and the defined contraction operator. (**K**–6) is a consequence of the deductive closure of $\mathbf{K}(s)$, $\mathbf{K}(s - \phi)$ and that the resulting knowledge states $\mathbf{K}(s - \phi)$ and $\mathbf{K}(s - \psi)$, ϕ is true in every situation in which ψ in true. □

Theorem 5 shows that forgetting in the Situation Calculus is well-behaved with respect to the AGM contraction postulates. It is worth clarifying the definition of the recovery postulate (**K**–5). While contraction is defined here with respect to a forgetting action, expansion is not defined in terms of sensing actions but rather in terms of a knowledge set. One reason for this is that an expansion operator defined in terms of sensing actions could easily be shown to fail recovery in cases where a sensing action's precondition is not satisfied after the forgetting action. For example, consider a robot that senses whether a light is on in the bedroom, then moves to the kitchen. If it then forgets that the light was on in the bedroom it cannot simply re-sense the fluent as it would first need to move back to the bedroom. However, an important aspect of the AGM recovery postulate is that it provides a notion of minimality; namely, that contraction does not throw away more than it needs to. Consequently, the chosen definition of expansion fits closely the intention of the AGM in defining a notion of recovery.

Literal Forgetting

In this paper we are interested in the notion of *knowledge forgetting*. However, the more commonly studied form of forgetting is that of *logical forgetting* (Lin and Reiter 1994). In a formal sense, logical forgetting involves the replacement of a formula with one that is logically weaker and comes in two forms: forgetting a grounded literal or fact (also known as *literal forgetting*) and forgetting a relation.

While logical and knowledge forgetting are superficially unrelated, nevertheless recent work has established a connection between literal forgetting and belief erasure (Nayak, Chen, and Lin 2007). Therefore, it is worth exploring whether there is any connection between logical and knowledge forgetting. However, we shall show that in general there is no direct equivalence between the two and that logical forgetting is more drastic in what is forgotten than is strictly necessary for knowledge forgetting.

In the following discussion we only provide a brief outline of the formalism behind literal forgetting, however the interested reader is referred to (Lin and Reiter 1994). Literal forgetting is defined in terms of models of theories where conditions are provided such that a theory T' is the result of *forgetting about* ground atom p in theory T (written *forget*$(T; p)$). A key result is that, for finite theories, literal forgetting can be achieved through a number of purely syntactic manipulations:

1. For formula φ and ground atom $P(\vec{t})$, let $\varphi[P(\vec{t})]$ denote the replacement of occurrences of $P(\vec{t'})$ in ϕ by the disjunctive tautology: $[\vec{t} = \vec{t'} \wedge P(\vec{t})] \vee [\vec{t} \neq \vec{t'} \wedge P(\vec{t'})]$.
2. Denote $\varphi^+_{P(\vec{t})}$ as the result of replacing $P(\vec{t})$ by $true$ in $\varphi[P(\vec{t})]$, and $\varphi^-_{P(\vec{t})}$ as the result of replacing $P(\vec{t})$ by $false$ in $\varphi[P(\vec{t})]$.

Hence for $T = \{\varphi\}$ and ground atom p:

$$\mathit{forget}(T, p) = \{\varphi^+_p \vee \varphi^-_p\}$$

Example 1 (from (Lin and Reiter 1994)) *Let $T = \{\varphi\}$, where $\varphi = \forall x.student(x)$ and we want to forget that $student(John)$. Now, $\varphi[student(John)]$ is:*

$$\forall x.[x = John \wedge student(John)] \vee [x \neq John \wedge student(x)].$$

Hence $\varphi^+_{student(John)}$ is equivalent to:

$$\forall x.x \neq John \supset student(x)$$

and $\varphi^-_{student(John)}$ is equivalent to:

$$\forall x.x \neq John \wedge student(x)$$

So forget$(T; student(John))$ *is equivalent to:*

$$\{\forall x.x \neq John \wedge student(x)\}$$

As is clear from Example 1, literal forgetting provides for a strong form of forgetting that removes the given literal entirely from the theory. Unfortunately, for our purposes, this is too strict a requirement for the form of knowledge forgetting that we require. This can be seen with reference to the robot scenario.

Firstly, in order to align the two frameworks, we can reuse the notion of a knowledge state defined in terms of a model (Definition 11). Now, consider the knowledge state of the robot in the example scenario from the point after it has entered the room but before it performs the forgetting action (Figure 1(b)). At this point the robot will certainly know that C_1 is the key combination. However it will also have the base knowledge that exactly one of C_1 and C_2 is the key combination. Hence the robot's knowledge state will contain the following key related clauses: $\mathit{Key}(C_1), \mathit{Key}(C_1) \vee \mathit{Key}(C_2), \neg\mathit{Key}(C_1) \vee \neg\mathit{Key}(C_2)$. Then, after forgetting, the robot will no longer know that C_1 is the key but critically it will still know that the key is exactly one of C_1 or C_2. Hence it will retain the key related facts: $\mathit{Key}(C_1) \vee \mathit{Key}(C_2), \neg\mathit{Key}(C_1) \vee \neg\mathit{Key}(C_2)$. The robot has forgotten the result of the sensing action and no more.

On the other hand, if we were to perform logical forgetting of $\mathit{Key}(C_1)$ on the robot's state, then applying the syntactic method outlined earlier, all references to the literal $\mathit{Key}(C_1)$ will be removed from the knowledge state and each of the three key related clauses will be replaced by $True$. Hence the robot will not only have forgotten that C_1 is the key but also that the key is one of C_1 or C_2.

This example highlights that knowledge forgetting is indeed distinct from logical forgetting. In particular, the knowledge forgetting approach can provide for a more fine-grained form of forgetting where appropriate.

Conclusion and Future Work

In this paper we have motivated the need for agents to be able to forget knowledge. Based on an existing approach to

modelling agent knowledge acquisition in the Situation Calculus (Scherl and Levesque 1993), we developed a formal framework that allows for an agent to both acquire and forget knowledge. An extensive analysis of the framework was provided, highlighting important cases under which knowledge acquisition and forgetting can take place. We further evaluated our new framework against existing models that provide for similar behaviour. In particular, we showed that our model is well-behaved with respect to the well-known AGM contraction postulates (Alchourrón, Gärdenfors, and Makinson 1985), but is distinct from the notion of logical forgetting (Lin and Reiter 1994). In this latter case it was shown that knowledge forgetting can provide for more fine-grained control over what information can be forgotten.

This work presents a number of interesting avenues for future research. Importantly, the relationship to notions of forgetting in epistemic logics (Zhang and Zhou 2009; van Ditmarsch et al. 2009) should be explored. While not concerned with environmental change as a result of actions, nevertheless there are commonalities with Situation Calculus formalisms that deserve further exploration. For example, the formalisation of knowledge forgetting from Zhang and Zhou (Zhang and Zhou 2009) defines four postulates for a forgetting operator, which it should be possible to show are satisfied by our notion of forgetting.

It would also be useful to see if the techniques used to achieve forgetting in epistemic logics, for example the *introspective forgetting* approach (van Ditmarsch et al. 2009), can be applied to our framework. Depending on the requirements of the problem domain, this could potentially provide for alternative forgetting actions, perhaps with different properties and fewer restrictions over what knowledge can and cannot be forgotten.

Finally, it would be interesting to explore the applications that were motivated in the introduction. In particular, the application to the analysis of nonmonotonic cryptographic protocols would most probably necessitate an extension of this model to the multi-agent context. Using previous research that extends the original Scherl and Levesque knowledge framework to the multi-agent setting in the context of game playing (Schiffel and Thielscher 2011), it would then be necessary to see if similar techniques could be applied to our extended framework.

Acknowledgements

This research was supported under Australian Research Council's (ARC) *Discovery Projects* funding scheme (project number DP 120102144). The fourth author is the recipient of an ARC Future Fellowship (project number FT 0991348) and is also affiliated with the University of Western Sydney.

References

Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic* 50(2):510–530.

Delgrande, J.; Hunter, A.; and Grote, T. 2010. On the representation and verification of cryptographic protocols in a theory of action. In *Proceeding of the 8th International Conference on Privacy, Security and Trust (PST)*, 39–45. IEEE.

Giacomo, G. D.; Lespérance, Y.; and Patrizi, F. 2012. Bounded situation calculus action theories and decidable verification. In *Principles of Knowledge Representation and Reasoning: Proceedings of the 13th International Conference (KR'12)*. AAAI Press.

Giacomo, G. D.; Lespérance, Y.; and Patrizi, F. 2013. Bounded epistemic situation calculus theories. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*. IJCAI/AAAI.

Lesperance, Y.; Levesque, H. J.; Lin, F.; and Scherl, R. B. 1995. Ability and knowing how in the situation calculus. *Studia Logica* 66:2000.

Levesque, H. J. 1996. What is planning in the presence of sensing? In *Proceedings of the 13th National Conference on Artificial Intelligence*, AAAI'96, 1139–1146. AAAI Press.

Lin, F., and Reiter, R. 1994. Forget it! In *In Proceedings of the AAAI Fall Symposium on Relevance*, 154–159.

McCarthy, J. 1963. Situations, actions and causal laws. Stanford University Artificial Intelligence Project Memo 2.

Nayak, A. C.; Chen, Y.; and Lin, F. 2007. Forgetting and update – an exploration. In *Formal Models of Belief Change in Rational Agents*, Dagstuhl Seminar Proceedings.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press.

Rubin, A., and Honeyman, P. 1994. Nonmonotonic cryptographic protocols. In *Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings*, 100–116.

Schacter, D. L.; Gilbert, D. T.; and Wegner, D. M. 2011. *Psychology*. Worth Publishers. chapter 6, 240–241.

Scherl, R., and Levesque, H. 1993. The frame problem and knowledge-producing actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 689–695.

Scherl, R. B., and Levesque, H. J. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence* 144(12):1 – 39.

Schiffel, S., and Thielscher, M. 2011. Reasoning about general games described in GDL-II. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 846–851.

Shapiro, S.; Pagnucco, M.; Lespérance, Y.; and Levesque, H. 2011. Iterated belief change in the situation calculus. *Artificial Intelligence* 175(1):165–192.

van Ditmarsch, H. P.; Herzig, A.; Lang, J.; and Marquis, P. 2009. Introspective forgetting. *Synthese* 169(2):405–423.

Zhang, Y., and Zhou, Y. 2009. Knowledge forgetting: Properties and applications. *Artificial Intelligence* 173(16-17):1525–1537.

Computing Narratives of Cognitive User Experience for Building Design Analysis KR for Industry Scale Computer-Aided Architecture Design

Mehul Bhatt, Carl Schultz, Madhura Thosar
Cognitive Systems, and
Spatial Cognition Research Center (SFB/TR 8)
University of Bremen, Germany

Abstract

We present a cognitive design assistance system equipped with analytical capabilities aimed at anticipating architectural building design performance with respect to people-centred functional design goals. The paper focuses on the system capability to generate *narratives of visuo-locomotive user experience* from digital computer-aided architecture design (CAAD) models. The system is based on an underlying declarative narrative representation and computation framework pertaining to conceptual, geometric, and qualitative spatial knowledge. The semantics of the declarative narrative model, i.e., the overall representation and computation model, is founded on: (**a**) conceptual knowledge formalised in an OWL ontology; (**b**) a general spatial representation and reasoning engine implemented in constraint logic programming; and (**c**) a declaratively encoded (narrative) construction process (based on search over graph structures) implemented in answer-set programming.

We emphasise and demonstrate: complete system implementation, scalability, and robust performance & integration with industry-scale architecture industry tools (e.g., Revit, ArchiCAD) & standards (BIM, IFC).

Motivation

Professional planners, designers, engineers, and architects are required to imagine and anticipate user experiences and requirements given a *design*;[1] designers embed these aspects into their designs typically via an iterative refinement process that is primarily driven by creativity, heuristics, and interdisciplinary domain expertise. Whilst achieving the correspondence between physical (design) *structure and function*, architects go through a process of creative visuo-spatial abstraction, design conceptualisation, and the translation of an abstract specification into a concrete product that can be built in the physical world. In doing so, the architect must adopt and anticipate the perspective of a range of possible stakeholders, people groups, and situations, e.g., regular users, users with special needs (blindness, wheel-chair access, elderly), children, fire-fighters, and emergency situations. Put simply, the architect is confronted with the task to produce an elaborate human-centred description of the *perceptual* and *locomotive experience* of the users of the building being designed.

Contemporary CAAD Technology A Computer-Aided Architecture Design (CAAD) model consists of a detailed 3D description of a physical environment. Built on basic geometric primitives such as *point, line-segments, and polygons*, professionally designed CAAD models produced using contemporary design tools such as ArchiCAD and Autodesk consist of an elaborate specification sufficiently suited for civil engineering and product deployment tasks, e.g., involving ensuring structural consistency, advanced building simulation of aspects such as energy, air-flow, and cost-estimation.

Within architectural computing, recent years have witnessed the development of novel forms of representational and computational paradigms, also inherently geometrically-driven, such as parametric and generative design. However, what CAAD models and advanced structural engineering and simulation methods lack is a human-centred, semantic perspective of the design. In essence, within state of the art CAAD technology, the design conception, semantic modelling, and design communication (e.g., by 3D visualization) modalities have continued to retain their essential engineering-centred "*geometric*" character over the years.

Next-Generation CAAD Next-generation people-centred design *systems, frameworks, assistive tools, educational aids, and design policies* necessitate foundational abstraction and computational building blocks where the modalities of human perception, action, environmental experience, and design conception and semantics are central. Our research in this context addresses the following questions:

Q1. Contemporary computer-aided architecture design (CAAD) tools provide robust geometric modeling and structural engineering methods; how can the future evolution of (architectural) design computing bring notions of design semantics, structure, function, and people-centred design to the fore at an ontological, representational and computational level?

Q2. What is the role of specialized forms of visuo-spatial

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] By design, we refer to spatial design in general, and building architecture design in particular. By conventional design systems, we mean computer-aided architectural design (CAAD) tools.

perception, abstraction, and commonsense spatial reasoning, within the broader realm of design computing, spatial design assistance, and tools for design learning and education?

Q3. What is the nature and form of the analytical feedback that designers and planners expect during the early design conception and iterative refinement phase? What are the implications of this from the viewpoint of the usability, interface, and human-computer interaction design aspects of architectural design (assistance) systems?

From the viewpoint of (Q1–Q3), the particular emphasis of our research has been on investigating the in-roads from the field of KR as foundational technologies within next-generation CAAD systems. Our perspective on artificial intelligence (AI) for (architecture) design is founded on the articulation of the *Science of Design* by Herbert Simon, and with Simon's interpretation of design as a "*decision-making process under constraints of physics, logic, and cognition*" (Baldwin, 2007). This view of the scientific design process underlies much of what artificial intelligence has to offer by way of its formal representational and computational apparatus to the domain of design computing.[2]

Cognitive CAAD Our basic proposition is that the foundational informatics of (architecture) design systems, tools, and assistive analytical aids concerned with creative spatial design and engineering tasks should also be based on modalities of visual and spatial cognition at the scale of everyday human perception and thinking. Toward this end, we propose that *design semantics, commonsense spatial reasoning and cognition*, and *visuo-spatial abstraction and computing* should be the driving forces underlying the foundations of next-generation design computing systems and paradigms. Next-generation CAAD should approach architecture computing from the perspective of *spatial informatics*, and appeal to *space* at the scale of everyday human perception and thinking in the context of *spatial cognition* (Bhatt, Schultz, and Freksa, 2013).

Our research addresses the representation of *space* from a formal modelling and computational viewpoint, i.e., *space*, as it is interpreted within the computer science disciplines concerned with the investigation of artificial intelligence and knowledge representation in general, and logic-based geometric and spatial representation and reasoning in particular (Bhatt et al., 2011). This field report focusses on the KR-specific aspects in the backdrop of system components concerned with design knowledge abstraction, representation, and reasoning. This report elaborates on (F1–F5):

I. Scope — Architecture Design

F1. *professional architectural design* as a function-driven iterative refinement process involving the structuring of *empty space*

F2. the *structuring of empty space* by an architect as being driven by the anticipation of visuo-spatial and locomotive user experience of people in built-up space (and in different situations)

II. Methods — AI for Design Computing

F3. the need for next-generation design tools and frameworks to regard human-centred modalities of *visuo-spatial perception and cognition* as being central;

F4. with respect to the "design as problem solving" paradigm of Simon (Simon, 1969, 1996), the development of artificial intelligence and spatial cognition & computation driven people-centred *analytical design computing* foundations for next-generation design systems

III. Demonstration — Declarative Narrativisation

F5. the computational *narrativisation of user experience* by way of declaratively modelled analytical descriptions of user experience externalised using natural language. This serves as: (1) a concrete example of our concept for KR for next-generation design computing foundations; and (2) a narrative-based *cognitivity benchmark* for qualitatively evaluating the functionality of our people-centred analytical design computing framework.

The paper presents an overview of the overall cognitive design assistance system, whilst emphasising:

- the underlying (general) declarative techniques for commonsense conceptual and qualitative spatial representation and reasoning rooted in methods such as formal ontology and description logics, constraint logic programming, and answer-set programming (Gelfond and Lifschitz, 1988; Baral, 2003); and
- the system capability to generate *narratives of cognitive user experience* based on the underlying KR-driven model and implementation as a model for next-generation people-centred analytical design computing foundations.

The paper illustrates case-studies and results obtained from a fully implemented system. The case-study focusses on *narratives of user experience* in the Museum Calouste Gulbenkian in Lisbon, Portugal. However, we emphasise that the narrative generation capability is functional for any digital model of a building that conforms to architecture industry standards, namely, the Building Information Model (BIM) (Eastman et al., 2008) and Industry Foundation Classes (IFC) (Froese et al., 1999). In this context, we also provide empirical results about the performance of the computational framework, focussing on the core design analysis algorithms and graph search processes implemented in answer set programming.

[2] In recent years, several interdisciplinary initiatives comprising of computer scientists, engineers, psychologists, and designers have addressed the application of artificial intelligence techniques for solving problems that accrue at several stages of the design process: design creativity and conceptualization, functionality specification, geometric modelling, structural consistency & code-checking, optimization, collaborative (design) workflow management, and a plethora of other issues. The journal "Artificial Intelligence for Engineering Design, Analysis and Manufacturing" completed two decades of publishing in 2007 and its anniversary publication is a good overview of the area (Brown, 2007; Gero, 2007). A sketch of '*40 years of design research*' is available in (Bayazit, 2004).

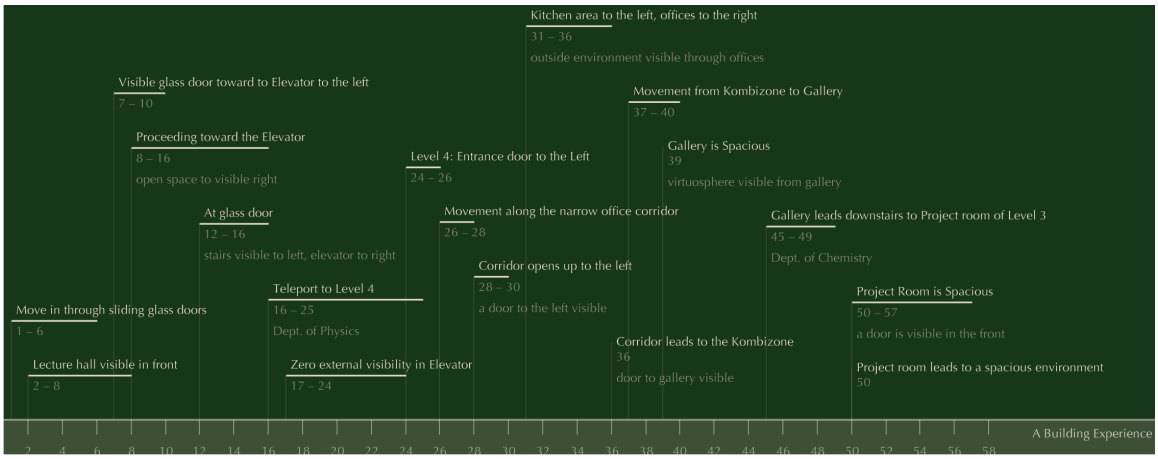


Figure 1: A Visuo-Locomotive Building Experience

Narrative and Design Cognition

A *narrative* in its most general (dictionary definition) form corresponds to "*a spoken or written account of connected events; a story*". Narratives serve a crucial role in everyday human perception and cognition; narrativisation of everyday perceptions by humans, and the significance of narratives in communication, interaction, and belief formation has been investigated under several frameworks, e.g., discourse analysis and narratology (Herman, Jahn, and Ryan, 2005), the narrative paradigm (Fisher, 1987), and through several other interdisciplinary initiatives involving the arts, humanities, and natural sciences. Most recently, the trinity of logic, language, and computer science has begun nurturing the field of "*computational models of narratives*" (Mani, 2012; Finlayson et al., 2013). To understand the nature of narratives from the viewpoint of the research presented in this paper, consider the following task:

> **Moving around in a building**. You enter a building (e.g., a museum, or air-port), possibly for the first time; as you walk around, guided by its internal structure, you (are required to) form and record your experience based on visuo-spatial, locomotive, and environmental affordance-based perceptions in the building.

Given the objective to externalise the observed perceptions in the building as required above, a human subject would be able to achieve the task using a range of modalities grounded in language, diagrams, schematisations etc. The experience may be described using a range of descriptive modalities such as written or spoken natural language (e.g., involving expressive motion, path, and qualitative spatio-linguistic predicates), diagrammatic representations (e.g., sequence graphs, bubble diagrams, schematisations of the environment), wayfinding experience (rotation or turn actions performed, getting lost) etc. For instance, a natural language description of the experience could be as follows (Fig. 1):

> **A Narrative of user experience**. As you move in through the passage of the sliding doors, you see a circular lecture hall directly in front through the glass panel, the elevator on the left...Exiting the elevator on Level four, there is a door to the left, leading up to a long, narrow corridor with an sequence of offices on the right...

Basically, human cognitive processes concerned with perceptual information processing would be able to conceptualise and externalise *a story* —linguistic or otherwise– that reports the building experience with relative ease; a large-scale experiment —typical in the field of environmental psychology (Bechtel and Churchman, 2002)— with many subjects would serve as a good reflection of the collective narrative of user experience in the environment under consideration; Fig. 1 presents one example of such an aggregate qualitative description of the experience of users in a given building. Architects concerned with designing a building are confronted with *imagining and anticipating* the perceptual experience of building users during the initial (design) conception phase, at a time when all that exists is *empty space* (Fig. 3(a), and 3(g)). In general, architects must envision the cognitive experiences of a range of people / user groups in different situations (in addition to externalising their own specialist analyses on functional design performance, and creative & aesthetic preferences).

Assistive design computing systems that can —based on an underlying formal representation and reasoning apparatus— generate narratives of user experience with the descriptive complexity of an architect or a user of a building are therefore needed, and can serve a good developmental benchmark. Methodologically, the objective of such systems is to provide high-level, semantic, analytical design computing capabilities aimed at assisting the architect or designer in a function-driven creative design task.

In this paper, we use the framework of narratives to report on a set of KR-driven *analytical design computing* capabilities concerned with providing decision-support to an architect or planner. Our design assistance system is able to compute narratives of user experience solely from geometric CAAD models or building master plans. Our long-term goal

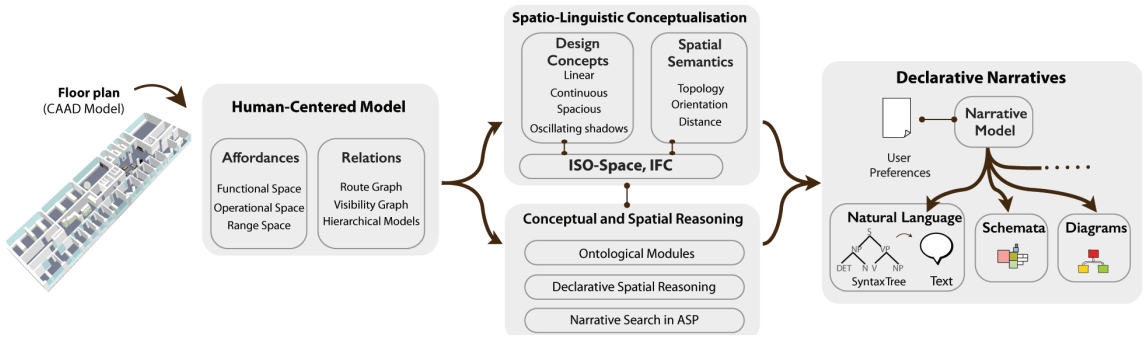


Figure 2: Reverse Engineering CAAD: Overview of Declarative Narrativisation Process

is to achieve a level of analytical accuracy, and semantic and descriptive quality and expressibility that is comparable with the performance of specialist designers, and novice users of buildings – this guides our broad-based work on computational narratives for KR-driven architectural design computing & cognition.

Computing Narratives

We adopt the conceptual framework of *narratives*, and specialise it for computing declarative narratives directly from 3D CAAD models of built-up spaces; this is akin to *cognitively-driven reverse engineering of CAAD models*. The semantic information that is obtainable from the computed narratives of user experience consist of visuo-spatial, locomotive, and affordance-based analytical content conforming to qualitative, spatio-linguistic conceptualisations pertaining to built-up spaces in the minds of people – both specialist architects, as well as potential users of a building.

Declarative Narrativisation Descriptions of user experience in buildings, e.g., in the form of a linguistic narrative, may be human-generated, or they could be generated by a system or suite of algorithms:

1 *Human-Generated Narratives*. Listing 1 (Fig. 4) depicts an example of the kind of natural language description that a human user may generate. The descriptions are representative of a user experience recording in an existing public building. We refer to the descriptions in Listing 1 as "*narratives of user experience*".

2 *System Generated Narratives*. Listings 2 (Fig. 4) is an example of computationally generated narratives of user experience — these have been generated solely on the basis of an elaborate 3D geometric CAAD model (of a real museum building). We refer to the formal knowledge structures and models from which such (linguistic or other) analytical descriptions of user experience can be generated as "*declarative narratives of user experience*". We refer to the process of computationally generating the formally characterised declarative narratives as *declarative narrativisation*.

Why Declarative? The declarative-ness (of narrativisation) signifies the existence of models that can be reasoned and queried upon, e.g., within a traditional declarative KR framework such as logic programming, constraint logic programming, description logic based reasoning, answer-set programming, and even other commonsense reasoners based on expressive action description languages.

The Declarative Narrativisation Framework

Figure 2 presents a conceptual overview of the narrativisation framework, system components, and the steps leading up to the computation of the declarative narratives, and the generation of language as a way to materialise the analytical information contained in the declarative narratives. Core components (NF1–NF3) are elaborated in the following:

NF1. People-Centered Design Typology Affordance spaces are spatial entities whose geometric extensions in space are derivable from the other (explicitly defined) material objects (e.g., door, wall, furniture; Fig. 3(b)), or their attributes, e.g., the opening space of a door can be computed from basic parameters of the door itself. Affordance spaces are derived from the abstracted shape geometries of physical objects chiefly using (a) polygonal boolean operations (intersection, difference, union) (Lauther, 1988), (b) line-of-sight calculations (Asano et al., 1985), and Minkowski sum (Kaul, O'Connor, and Srinivasan, 1992):

A1. *Movement Spaces*. Union of navigable surfaces (e.g. slabs) subtracted by obstacles such as walls (Fig. 3(f)).

A2. *Range Spaces*. Point-visibility polygons (isovist) restricted to the sensor's angular field of view and focus distance (Fig. 3(c)).

A3. *Operational Spaces*. Sweeping, extruding, translating, rotating, and scaling parts of the physical geometry of the reference object (e.g. sweeping a door panel; Fig. 3(d)).

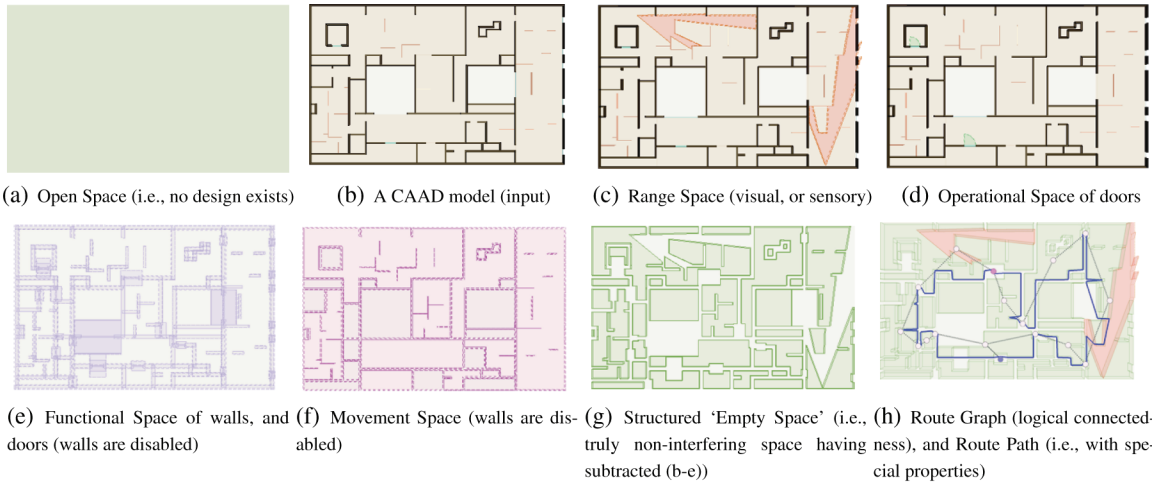
A4. *Functional Spaces*. Buffer of the physical geometry of the reference object subtracted by obstacles (Fig. 3(e)).

A5. *Empty Spaces*. Union of movement spaces subtracted by other affordance spaces such as functional and range spaces spaces (Fig. 3(g)).

A6. *Route Graph*. Connectivity relations are intersections between movement spaces and waypoints (e.g. doors). Actual route paths provide precise co-ordinates of motion along a route (Fig. 3(h)).

A7. *Visibility Graph*. Visibility relations are intersections between visibility spaces and objects (Fig. 6(c))

The spatial design typology in (A1–A7) provides a basis to further analyse and "make sense" of the "*shape of the*



(a) Open Space (i.e., no design exists) (b) A CAAD model (input) (c) Range Space (visual, or sensory) (d) Operational Space of doors

(e) Functional Space of walls, and doors (walls are disabled) (f) Movement Space (walls are disabled) (g) Structured 'Empty Space' (i.e., truly non-interfering space having subtracted (b-e)) (h) Route Graph (logical connectedness), and Route Path (i.e., with special properties)

Figure 3: Structuring Empty Space. Illustrated using the derivations computed by our design analysis engine. Floor plan: *Museum Calouste Gulbenkian, Lisbon, Portugal*.

Listing 1. Human Generated User Experience (Gaizauskas et al., 2012): (1) The main entrance to the building is located in a corner under an overhang, which does not allow it to be visible to patients easily. (5) The immediate interior area around the entrance feels reassuring because it is open and airy. (11) Most of the windows in the consultation rooms overlook the courtyards. (18) [The] space behind the sculpture could be used for outdoor seating in the summer and passers-by would be able to see that there was a cafe available in the area.

Listing 2. System Computed Specialist Analysis: The layout and spatial organisation of the museum maintains 'continuity' between locations. The overall plan follows a circular structure, starting at the front lobby, passing through Rooms A, B, C, D, and via the North Door of Room E. The rooms flow linearly, and maintain visibility with the external environment (except during the segment between Room C and Room D). By removing Wall Y in Room X, the circular ring structure can be converted to a hierarchical structure with Room Z as the central hub. Direct sunlight exposure is achieved in approximately 85% of the floor plan. Region X never receives any sunlight at anytime during the year.

Figure 4: Extracts of (1) human narratives about buildings (Gaizauskas et al., 2012), and (2) System Computed Narratives from a CAAD model of the Museum Calouste Gulbenkian, Lisbon, Portugal

empty" (Bhatt, Schultz, and Huang, 2012) that results from a configuration of a designed structure as available within a CAAD model. This is done by establishing a formal link between low-level physical design structure, and high-level conceptualisation of design function as identifiable by the spatio-linguistic conceptualisation of architects, i.e., by formalising modelling and reasoning about design semantics.

NF2. Design Semantics and Ontology Whereas the philosophy of *form and function* in design is a well-studied topic, the formal modelling of *structural form* and resulting *artefactual function* within design assistance systems remains elusive. We make these definitions explicit by ontologically modelling architectural domain entities, and their properties and related constraints. We interpret "(structural) form" and "(artefactual) function" by specifying modular ontologies and their interplay for the architectural design domain.

Multi-Perspective Semantics and Modularity. Consider the illustration in Fig. 5: an abstraction such as a Room or Wall may be identified semantically by its placement within an ontological hierarchy and its relationships with other conceptual categories. These different categories are used by a designer during the initial design conceptualisation phase. However, when a design is implemented in a CAAD tool, the same concepts acquire a new perspective, i.e., the designer has to interpret design concepts in terms of points, line-segments, polygons and other geometric primitives available by the design tool. Given the lack of design semantics within contemporary design tools, no solution is available for a knowledge-based system to make inferences about the conceptual design and its geometric interpretation within a CAAD model in a unified manner. Our approach is to model these perspectives and their respective spatial semantics by using modular ontologies that individually comply with one of the perspectives. Presently, we represent conceptual, qualitative, and quantitative ontological modules using the Web Ontology Language (OWL). These modularly specified ontologies not only reflect the thematically different perspectives in a more adequate way but they also provide a clear ontological representation of the interplay and exchange between form and function, and underlying spatial constraints. This also directly supports modular conceptual and spatial reasoning (M1-M3):

M1. *Conceptual Space* This ontological module reflects terminological information of architectural entities. Here, the entities are regarded as such, i.e., they are defined according to their properties without taking into account the

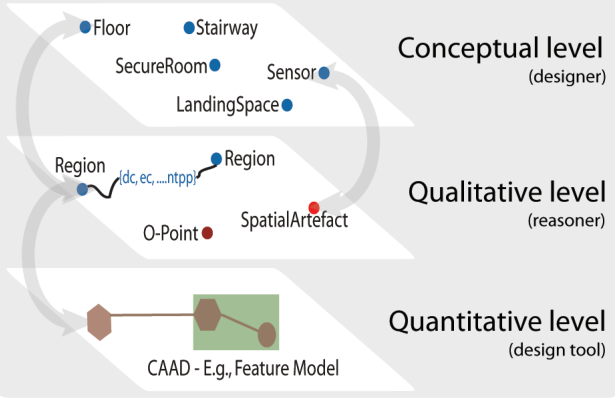


Figure 5: Multi-Perspective Semantics and Ontological Modularity

context in which they are put. The concept space is based on DOLCE (Masolo et al., 2003), in particular, on the OWL version DOLCE-Lite[3]. For instance, the class construction below specifies that types of buildings, e.g., courthouse, museum or university, are non-physical endurants that depend on an actual (physical) building, which provides the building's type or function:

$$\text{BuildingType} \sqsubseteq \text{dolceLite:non-physical-endurant} \sqcap \exists\, \text{dolceLite:generically-dependent-on} \,.\, \text{PhysicalBuilding}$$

Module M1 defines other similar constructs that occur in the design domain, such as VisualSensor, SlidingDoor, LandingSpace, OpenFloor.

M2. *Quality Space* This ontological module reflects qualitative spatial information of architectural entities. In particular, the module uses relations as provided by the topological spatial calculus RCC-8 (Randell, Cui, and Cohn, 1992), and calculi pertaining to other aspects of space. Qualitative spatial relations provide an abstraction method for specifying high-level design knowledge, e.g., a Room is necessarily a *proper part* of a Building. For instance, the category OpSpaceDoor specifies the region that is composed by a door and its opening angle, the category OpSpaceWall specifies the region that is composed by the external boundary of a wall, and the category OpSpaceWindow specifies the region that is composed by a windows and its opening angle.

$$\begin{aligned} \text{OpSpaceDoor} &= \langle \text{has\_operational\_space} \rangle^{M2}\, \text{Door} \\ \text{OpSpaceWall} &= \langle \textit{has\_operational\_space} \rangle^{M2}\, \text{Wall} \\ \text{OpSpaceWindow} &= \langle \textit{has\_operational\_space} \rangle^{M2}\, \text{Window} \end{aligned}$$

The following constraint encodes the requirement that all doors have to be externally connected (EC) with doors, walls, or windows:

$$EC(\text{OpSpaceDoor},\ \sqcup\ \text{OpSpaceWall} \sqcup \text{OpSpaceWindow})$$

[3]DOLCE-Lite. http://www.loa-cnr.it/ontologies/DOLCE-Lite.owl

M3. *Quantity Space* This ontological module reflects geometric information of architectural entities. It is closely related to the Industry Foundation Class (IFC) data model (Liebich et al., 2006) and partially mirrors the IFC classes; IFC consists of a non-proprietary data model and exchange format to foster interoperability in the building industry.[4] For example, Door, Wall, and Window are characterized together with their properties length, orientation, placement.

$$\begin{aligned} &\text{Door} \sqsubseteq \text{StructuralBuildingElement} \sqcap = 1\ \text{openingAngle} \,.\, \text{float} \\ &\sqcap = 1\ \text{doorknobType} \,.\, \text{ENTITY} \sqcap = 1\ \text{height} \,.\, \text{float} \\ &\sqcap = 1\ \text{length} \,.\, \text{float} \sqcap = 1\ \text{width} \,.\, \text{float} \end{aligned}$$

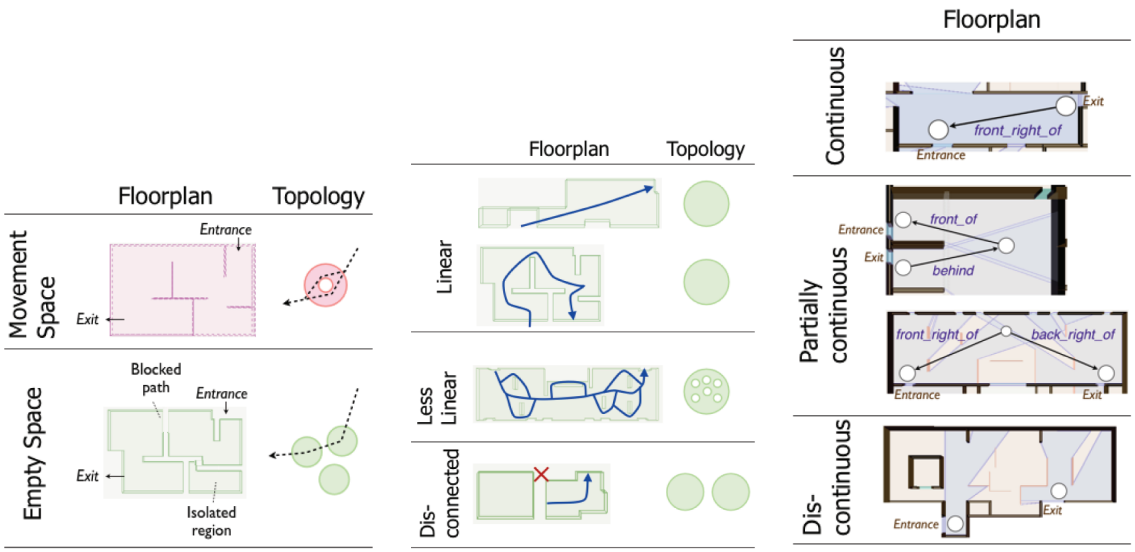
Data provided by IFC for a concrete building model can then be instantiated with respect to the quantity space ontological module.

Integrated Representation The connection of the three different modules (M1–M3) results in formalizing *link relations* across modules.[5] For example, an instance of Wall from the quantity space is related to an instance of Wall from the quality space, or as a physical endurant in the conceptual space. In the quantitative module, it is described by its length and position, while its counterpart in the qualitative module defines region-based relations to other walls and relations to rooms it constitutes.

Reasoning with Intergated Representation An important conceptual reasoning task in the design context is to determine whether or not a set of conceptual requirement constraints specified by a designer may possibly have a model. This form of reasoning is useful to check if a given set of design requirements are mutually consistent from the viewpoint of a conceptual specification; this kind of reasoning is directly possible with state of the art *Description Logic* (DL) (Baader et al., 2003) based reasoning systems such as RACER (Haarslev, Möller, and Wessel, 2004), PELLET (Sirin et al., 2007). With respect to the semantics of geometric and qualitative spatial knowledge, we have developed the CLP(QS), which is a declarative spatial reasoning system capable of modelling and reasoning about qualitative spatial relations pertaining to multiple spatial domains, i.e., one or more aspects of space such as topology, and intrinsic and extrinsic orientation, distance (Bhatt, Lee, and Schultz, 2011; Schultz and Bhatt, 2012). CLP(QS) also offers mixed geometric-qualitative spatial reasoning capabilities, and in its current form, a limited of quantification support offering

[4]IFC is based on object classes, e.g., *IfcWall* or *IfcWindow*, and their relationships containing metrical data as properties. We apply the stable release IFC2x3 TC1 (Liebich et al., 2006) defining 653 *building entities* (e.g., *IfcWall*) and additionally, several *defined types*, *enumerations*, and *select types* for specifying their properties and relationships.

[5]Bhatt, Hois, and Kutz (2012) provides technical details concerning the implementation of ontological modules using the framework of $\mathcal{E}$-Connections (Kutz et al., 2004). For the purposes of this paper, it suffices to think of link relations as predicates that link categories from respective modules; these links are utilised in a logic programming environment in order to assimilate knowledge from different perspectives within the body of inference rules in the logic program.



(a) Affordance path analysis can identify design malfunctions.

(b) Empty space topology indicates the linearity of a space.

(c) Visibility of landmarks indicated visual continuity

Figure 6: Affordance Paths, and Navigation Linearity and Visual Continuity

the means to go back from qualitative relations to the domain of precise quantitative information. CLP(QS) is implemented as a general library within the context of Constraint Logic Programming (CLP).

NF3. Spatio-Linguistic Conceptualisation & Reasoning Conceptual reasoning is hierarchical, deriving high-level design concepts from qualitative spatial relations based on low-level geometric primitives. Qualitative spatial representation and reasoning (QSR) provides a commonsensical interface to abstract and reason about quantitative spatial information. Qualitative spatial calculi are relational-algebraic systems pertaining to *topology, orientation, direction, size* (Cohn and Renz, 2007). Within computing for spatial design, the use of formal qualitative spatial calculi and conceptual design requirements serve as a link between the *structural form* of a design and the differing *functional capabilities* that it affords or leads to, e.g.:

C1. The FunctionalSpace of the Door of every Office should overlap with the RangeSpace of one or more Camera or MotionSensor.

C2. A MonitoredArea that are connected by doors and/or passages should not have any security blind spots whilst people transition from one room to another.

C3. Figure illustrates an inconsistency with circulation: a physical path exists between two locations if MovementSpace topology is connected, however, when taking FunctionalSpaces of objects into account we find that there is no affordance path; the EmptySpace topology is disconnected.

Examples in (C1–C3) constitute the basic case; more complex requirements are driven by high-level spatio-linguistic patterns and features, e.g., OpenPath, Around, Along etc. The semantics of such spatio-linguistic conceptual information is grounded with respect to the ISO-Space standard, an emerging initiative concerned with formalising the semantics of the representation of motion as it is expressed in language, e.g., *path predicates* and *manner-of-motion-predicates*.[6]. Finally, concepts most closely related to the design concept of an expert architect often involved a mixture of spatial prepositions, and high-level spatial patterns. We present some examples of such design concepts by way of **(A–C)**:

A. *Linearity*. Navigation linearity is associated with the number of path decisions that a person needs to make as they move through a space (Stephen Bitgood, 1992). A room is qualitatively *more linear* than another room if it has a less complex empty space topology, e.g., defined as having fewer regions with fewer holes (assuming that the rooms have the same number of entry and exit points). (Fig. 6(b))

B. *Visual Continuity*. Visual continuity through an environment is maintained if, as visitors move through the environment, they have visual contact with important waypoints (such as entrances, exits, or signage). (Fig. 6(c))

```
visual_continuity(discontinuous, id(LandmarkA), id(LandmarkB)):-
        visibility_space(id(LandmarkA),VisGeometryA),
        visibility_space(id(LandmarkB),VisGeometryB),
        topology(rcc8(dc), VisGeometryA, VisGeometryB).

visual_continuity(partial, id(LandmarkA), id(LandmarkB)) :-
        visibility_space(id(LandmarkA), VisGeometryA),
        visibility_space(id(LandmarkB), VisGeometryB),
        topology(rcc8(po), VisGeometryA, VisGeometryB).

visual_continuity(continuous, id(LandmarkA), id(LandmarkB)):-
        visibility_space(id(LandmarkA),VisGeometryA),
        physical_space(id(LandmarkB),PhyGeometryB),
        mereology(rcc5(pp), PhyGeometryB, VisGeometryA).
```

[6]The formal semantics underlying the ISO-Space framework is in Dynamic Interval Temporal Logic (DITL) and qualitative spatial formalisms as conceived within the QSTR community (Mani and Pustejovsky, 2012)

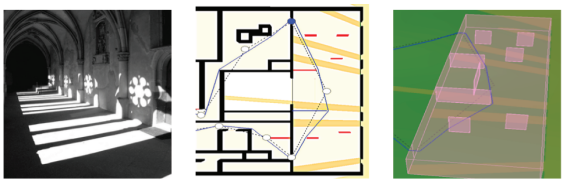


Figure 7: A Route through an Oscillating Shadow Pattern

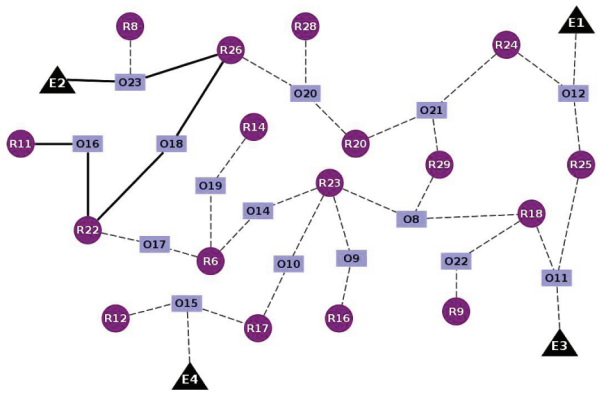


Figure 8: Route Computed by an ASP Solver

```
#const start = r11.
#const end = fireEx.

%%%%Domain of Rooms
objectClass(r0, cDsMovementSpace).
objectClass(r1, cDsMovementSpace).
objectClass(r2, cDsMovementSpace).
objectClass(r3, cDsMovementSpace). ...

%%%%Domain of Doors, Fire-Exits
objectClass(o0, dsHorizontalPassage).
objectClass(o1, dsHorizontalPassage).
objectClass(o2, dsHorizontalPassage).
objectClass(o3, dsHorizontalPassage). ...

connection(r6, o14). connection(r8, o23).
connection(r9, o22). connection(r6, o19).
connection(r6, o17). ...

edge(X, Y) :-
      connection(X, Y),
      objectClass(X, cDsMovementSpace),
      objectClass(Y, dsHorizontalPassage).
edge(X, end) :- connection(X, end).
edge(Y, X) :- edge(X, Y).

adj(X, Y) :- edge(X, Y).
adj(X, Y) :- adj(X, Z), adj(Z, Y), edge(X, Z).
:- not adj(start, end).
selected(X, Y) :- adj(X, Y), edge(X, Y), connection(X, Y).

#minimize[selected(X, Y)].
#hide.
#show selected/2.
```

Figure 9: Basic encoding for graph search as an answer set program (minimal example serving illustration purpose)

C. *Oscillating Patterns of Sunlight and Shadows.*

The influence of sunlight (and resulting shadows) in evoking aesthetic impressions is well known in design (Seidler, 1959). For instance, one may identify an oscillating pattern of sunlight and shadows at a given time and place by determining whether the shadows in a room have a particular topology (Fig. 7):[7]

```
sunlight(oscillating_pattern, id(Room), id(Sun)) :-
        movement_space(id(Room),MvGeometry),
        sunlight(id(Sun),SunGeometry),
        SunPattern=intersection(MvGeometry,SunGeometry),
        enumerate(disconnected_regions, SunPattern, N), {N > 3}.
```

NF4. From Declarative Narratives, to Narratives of User Experience NF1–NF3 focussed on the semantic and spatial abstraction and reasoning methods that are used to generate people-cognition centred visuo-locomotive analytical information. The computed and inferentially derived analytical information is available declaratively, i.e., it is possible to query the analytical information using a combination of:

- conceptual reasoning with a DL reasoner such as Racer (Haarslev, Möller, and Wessel, 2004), Pellet (Sirin et al., 2007) (e.g., retrieval tasks)
- (system level) integration of conceptual reasoning with constraint logic based spatial reasoning and question-answering (in Prolog)
- application of answer set programming (Fig. 9) to search over the available analytical information, e.g., wrt. a graph search encoding, thereby computing stable models that satisfy certain design constraints fulfilling an anticipated "visuo-locomotive experience". For instance, a range of routes that fulfil particular user experience criteria may be computed: e.g., routes to a fire-exit (Fig. 8), nearest room from where sunset or sunrise are visible, routes that meet certain experiential constraints (e.g., visibility of at least one Bathroom, visibility to external environment)

The analytical information, in order to be useful to designers, has to be communicated using human-centred modalities; toward this, possible approaches are:

- *simulation*: diagnostic pin-pointing using virtual reality (VR) or similar visual techniques (e.g., 2D and 3D simulation)
- *schematisation*: generation of abstract schematisations, dynamic graph based complex spatio-temporal data visualisations, diagrams (e.g., similar to the timeline in Fig. 1)
- *stories in natural language and speech*: language generation based on the declarative analytical information, i.e., narratives of user experience

For the discussion in this paper, we focus on the use of the analytical declarative knowledge to generate spatio-linguistically grounded narratives of user experience (e.g., Listing 2 in Fig. 4). Template syntax trees for describing particular design concepts (e.g. referring to *linearity* as "the space flows linearly") are encoded, and instantiated with respect to the declarative narrative model.[8]

[7] It is straightforward to determine the sun location based on a given time, or the time that corresponds to a given geospatial location.

[8] We utilise the open source tools available via the Stanford Natural Language Processing Group: http://nlp.stanford.edu/

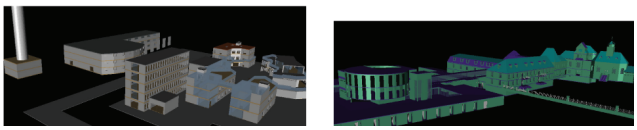


Figure 10: Select CAAD models used for empirical analyses – An Academic Interchange, and the Schloss Dagstuhl Palace

Empirical Performance Evaluation

The reported design analysis system and narrativisation framework have been fully implemented, and have also been used as a workbench for conducting experiments using real, industry-scale building data from CAAD models produced with commercial CAAD tools such as ArchiCAD and Autodesk Revit.

Datasets The dataset consists of 10 CAAD models of large-scale public environments of different sizes (e.g., an international academic interchange building (Germany), parts of the campus redevelopment plan at the University of Arkansa (USA), the Schloss Dagstuhl building(s), model of the Museum Calouste Gulbenkian (Portugal) etc (e.g., see Fig. 10).

Analysis The empirical analysis of the implemented system is performed for the following aspects (E1–E2):

E1. *Design Typology Computation*: Geometric computations are needed to compute the various affordance spaces as per the people-centred design typology.[9] The design typology computation experiments (Fig. 11) were conducted on a MacBookPro, OS X 10.6.8, 2.66 GHz with 4GB RAM. As can be seen Fig. 11, typology computations have been fine-tuned to suit practical applications demanding scalability for real-time performance; see (Schultz and Bhatt, 2013).

E2. *Narrative Generation via a Search Process in ASP*: The derivation of the narrative of user experience that has been implemented using answer set programming is evaluated. The ASP based experiments (Table 1) were conducted on an iMac machine with 2.7GHz, with 8 GB RAM. At this stage, fine-tuning the domain encodings in ASP has not been a principal concern. However, we are presently investigating effective encodings in ASP that specifically target graph structures and dataset sizes that are typical in the computer aided design domain.

Conclusions

We have presented a system for declarative narrativisation of user experience in spatial design. In contrast to the strictly geometric-centric view of building information, our framework explicitly incorporates human-centered perceptual models in the form of affordances and visuo-spatial abstraction. We have conducted experiments using real, industry scale CAAD models using a full implemention of our framework. In the experiments, all human-centered affordance spaces were computed within a relatively short time, thus demonstrating the practicality of the narrativisation framework in real-world settings. In the future we plan to extend the affordances to include a broader range of design features, and to formally benchmark the system-generated narratives wrt. human-generated narratives.

[9]Computing the *isovist* was done using the VisiLibity package (Obermeyer and Contributors, 2008). Polygon operations were performed using General-Polygon-Clipper., www.cs.man.ac.uk/~toby/alan/software

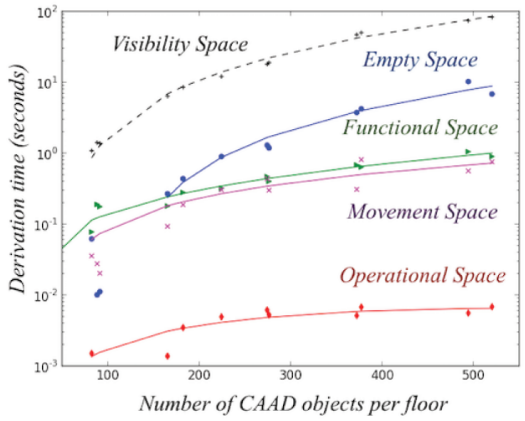


Figure 11: Time taken for Design Typology Computation

CAAD Model	Edges	Nodes	Grounding	Search
Gulbenkian	31	31	0.00	0.01
Flynn Interchange	32	30	0.00	0.01
Acad. Interchange (I)	90	86	0.03	0.36
Cartesium	150	148	0.13	9.68
AC11-Institute	161	161	0.15	4.58
Arkansas: Dining	188	179	0.19	82.13
Acad. Interchange (II)	215	208	0.16	18.17
Arkansas: Basketball	231	208	0.30	481.32
Arkansas: Football	243	227	0.39	315.93
Parkland Hospital	400	393	1.16	377.37

Table 1: *The average time (30 experiments per design) taken to compute shortest graph paths to simulate narrative generation*

References

Asano, T.; Asano, T.; Guibas, L.; Hershberger, J.; and Imai, H. 1985. Visibility-polygon search and euclidean shortest paths. In *Foundations of Computer Science, 1985., 26th Annual Symposium on*, 155–164. IEEE.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. 2003. *The Description Logic Handbook*. Cambridge University Press.

Baldwin, C. 2007. Steps toward a science of design. In *NSF Principal Investigators Conference on the Science of Design*.

Baral, C. 2003. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. New York, NY, USA: Cambridge University Press.

Bayazit, N. 2004. Investigating design: A review of forty years of design research. *Design Issues* 20(1).

Bechtel, R., and Churchman, A. 2002. *Handbook of Environmental Psychology*. Hoboken, NJ: John Wiley and Sons.

Bhatt, M.; Guesgen, H.; Wölfl, S.; and Hazarika, S. 2011. Qualitative Spatial and Temporal Reasoning: Emerging Applications, Trends, and Directions. *Spatial Cognition & Computation* 11(1):1–14.

Bhatt, M.; Hois, J.; and Kutz, O. 2012. Ontological Modelling of Form and Function for Architectural Design. *Applied Ontology* 7(3):233–267.

Bhatt, M.; Lee, J. H.; and Schultz, C. 2011. CLP(QS): A Declarative Spatial Reasoning Framework. In *Proceedings of the 10th international conference on Spatial information theory*, COSIT'11, 210–230. Berlin, Heidelberg: Springer-Verlag.

Bhatt, M.; Schultz, C.; and Freksa, C. 2013. The 'Space' in Spatial Assistance Systems: Conception, Formalisation and Computation. In Tenbrink, T.; Wiener, J.; and Claramunt, C., eds., *Representing space in cognition: Interrelations of behavior, language, and formal models. Series: Explorations in Language and Space*. 978-0-19-967991-1, Oxford University Press.

Bhatt, M.; Schultz, C.; and Huang, M. 2012. The Shape of Empty Space: Human-Centred Cognitive Foundations in Computing for Spatial Design. In *VL/HCC 2012: IEEE Symposium on Visual Languages and Human-Centric Computing*, 33–40.

Brown, D. C. 2007. AI EDAM at 20. *Artif. Intell. Eng. Des. Anal. Manuf.* 21(1):1–2.

Cohn, A. G., and Renz, J. 2007. Qualitative spatial reasoning. In van Harmelen, F.; Lifschitz, V.; and Porter, B., eds., *Handbook of Knowledge Representation*. Elsevier.

Eastman, C.; Teicholz, P.; Sacks, R.; and Liston, K. 2008. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. Frontiers in Artificial Intelligence and Applications. Wiley.

Finlayson, M. A.; Fisseni, B.; Löwe, B.; and Meister, J. C., eds. 2013. *2013 Workshop on Computational Models of Narrative, CMN 2013, August 4-6, 2013, Hamburg, Germany*, volume 32 of *OASICS*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Fisher, W. R. 1987. *Human communication as narration: Toward a philosophy of reason, value, and action*. Columbia, SC: University of South Carolina Press.

Froese, T.; Fischer, M.; Grobler, F.; Ritzenthaler, J.; Yu, K.; Sutherland, S.; Staub, S.; Akinci, B.; Akbas, R.; Koo, B.; Barron, A.; and Kunz, J. 1999. Industry foundation classes for project management - a trial implementation. *ITCon* 4:17–36.

Gaizauskas, R.; Barker, E.; Chang, C.; Derczynski, L.; Phiri, M.; and Peng, C. 2012. Applying ISO-Space to Healthcare Facility Design Evaluation Reports. In *Proceedings of the Joint ISA-7, SRSL-3 and I2MRT Workshop on Semantic Annotation and the Integration and Interoperability of Multimodal Resources and Tools*.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R. A., and Bowen, K. A., eds., *ICLP/SLP*, 1070–1080. MIT Press.

Gero, J. S. 2007. AI EDAM at 20: Artificial intelligence in designing. *AI EDAM* 21(1):17–18.

Haarslev, V.; Möller, R.; and Wessel, M. 2004. Querying the semantic web with Racer + nRQL. In *Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL'04)*.

Herman, D.; Jahn, M.; and Ryan, M.-L. 2005. *Routledge Encyclopedia Of Narrative Theory*. Routledge.

Kaul, A.; O'Connor, M.; and Srinivasan, V. 1992. Computing minkowski sums of regular polygons. In *Proc. 3rd Canadian Conference on Computational Geometry*, 74–77.

Kutz, O.; Lutz, C.; Wolter, F.; and Zakharyaschev, M. 2004. $\mathcal{E}$-Connections of Abstract Description Systems. *Artificial Intelligence* 156(1):1–73.

Lauther, U. 1988. An o (n log n) algorithm for boolean mask operations. In *Papers on Twenty-five years of electronic design automation*, 233–240. ACM.

Liebich, T.; Adachi, Y.; Forester, J.; Hyvarinen, J.; Karstila, K.; and Wix, J. 2006. *Industry Foundation Classes: IFC2x Edition 3 TC1*. International Alliance for Interoperability (Model Support Group).

Mani, I., and Pustejovsky, J. 2012. *Interpreting Motion - Grounded Representations for Spatial Language*, volume 5 of *Explorations in language and space*. Oxford University Press.

Mani, I. 2012. Computational modeling of narrative. *Synthesis Lectures on Human Language Technologies* 5(3):1–142.

Masolo, C.; Borgo, S.; Gangemi, A.; Guarino, N.; and Oltramari, A. 2003. Ontologies library. WonderWeb Deliverable D18, ISTC-CNR.

Obermeyer, K. J., and Contributors. 2008. The VisiLibity library. http://www.VisiLibity.org. R-1.

Randell, D. A.; Cui, Z.; and Cohn, A. G. 1992. A spatial logic based on regions and connection. In *Proc. of the 3rd International Conference on Knowledge Representation and Reasoning (KR'92)*, 165–176. Morgan Kaufmann, San Mateo.

Schultz, C., and Bhatt, M. 2012. Towards a Declarative Spatial Reasoning System. In *ECAI 2012 - 20th European Conference on Artificial Intelligence., Montpellier, France*.

Schultz, C., and Bhatt, M. 2013. Inspace3d: A middleware for built environment data access and analytics. In *ICCS*, 80–89.

Seidler, H. 1959. Sunlight and architecture. *Architectural Science Review* 2.

Simon, H. 1969. *The Sciences of the Artificial [The Karl Taylor Compton lectures.]. Chapter 3: The Science of Design*. Cambridge, MA, USA: MIT Press.

Simon, H. 1996. *The sciences of the artificial (3rd ed.)*. Cambridge, MA, USA: MIT Press.

Sirin, E.; Parsia, B.; Grau, B.; Kalyanpur, A.; and Katz, Y. 2007. Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web* 5(2):51–53.

Stephen Bitgood, J. Hines, W. H. 1992. Visitor circulation through a changing exhibits gallery. *Visitor Studies: Theory, Research, and Practice* 4:102–111.

SmartPM: An Adaptive Process Management System Through Situation Calculus, IndiGolog, and Classical Planning

Andrea Marrella and **Massimo Mecella**
Sapienza - Universitá di Roma, Rome, Italy
{*marrella—mecella*}*@dis.uniroma1.it*

Sebastian Sardina
RMIT University, Melbourne, Australia
sebastian.sardina@rmit.edu.au

Abstract

In this paper we present SmartPM, a model and a prototype Process Management System featuring a set of techniques providing support for automated adaptation of knowledge-intensive processes at run-time. Such techniques are able to automatically adapt process instances without explicitly defining policies to recover from exceptions and without the intervention of domain experts at run-time, aiming at reducing error-prone and costly manual ad-hoc changes, and thus at relieving users from complex adaptations tasks. To accomplish this, we make use of well-established techniques and frameworks from Artificial Intelligence, such as situation calculus, IndiGolog and classical planning.

1 Introduction

This paper is concerned with the use of of three well-established KR&R techniques—reasoning about actions, high-level programming, and automated planning—for the *automated adaptation* of knowledge-intensive processes that execute in highly dynamic settings. The work falls within the scope of *Business Process Management* (BPM) (van der Aalst, ter Hofstede, and Weske 2003), an active area of research that is highly relevant from a practical point of view while offering many technical challenges.

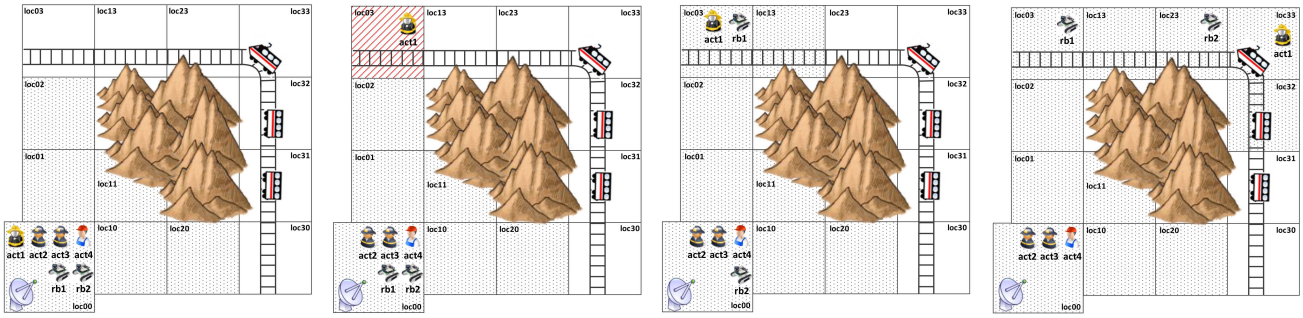
BPM is based on the observation that each product and/or service that a company provides to the market is the outcome of a number of activities performed. *Business processes* are the key instruments for organizing such activities and improving the understanding of their interrelationships. Examples of traditional business processes include insurance claim processing, order handling, and personnel recruitment. In order to support the design and automation of business processes, a new generation of information systems, called *Process Management Systems* (PMSs) have become increasingly popular during the last decade (Weske 2012). A PMS is a software system that manages and executes business processes involving people, applications, and information sources on the basis of *process models* (Dumas, van der Aalst, and ter Hofstede 2005). The basic constituents of a process model are *tasks*, describing the various activities to be performed by process participants (i.e., software applications, agents, or humans). The procedural rules to control such tasks, described by so-called "routing" constructs such as sequences, loops, parallel and alternative branches, define the *control flow* of the process. A PMS, then, takes a process model (containing the process' tasks and control flow) and manages the process routing by deciding which tasks are enabled for execution. Once a task is ready for execution, the PMS assigns it to those participants capable of carrying it on.

Current maturity of process management methodologies has led to the application of process-oriented approaches in new rich challenging scenarios, such as healthcare (Lenz and Reichert 2007), emergency management (Marrella, Russo, and Mecella 2012), and home automation (Helal et al. 2005). In those settings, processes generally reflect "preferred work practices", and the control flow is influenced by user decision making and coupled with contextual data and knowledge production. Such processes are known as *knowledge-intensive* processes (KiPs) (Di Ciccio, Marrella, and Russo 2012)—genuinely knowledge and data centric—and require the integration of the data dimension with the traditional control flow dimension.

During the enactment of KiPs, variations or divergence from structured reference models are common due to exceptional circumstances arising (e.g., autonomous user decisions or contextual changes), thus requiring the ability to properly *adapt* the process behavior (Sadiq, Sadiq, and Orlowska 2001). In knowledge-intensive scenarios, the fact is that the number of possible exceptions is often too large, and traditional manual implementation of exception handlers at design-time is not feasible (Reichert and Weber 2012). In fact, the designer often lacks the required knowledge to model all the possible exceptions at the outset.

To tackle this issue, we develop in this paper an approach, together with an actual implementation, to *automatically adapt KiPs at run-time* when unanticipated exceptions occur, thus requiring no specification of recovery policies at design-time. To that end, we shall resort to three popular Artificial Intelligence (AI) "technologies": situation calculus (Reiter 2001), IndiGolog (De Giacomo et al. 2009), and classical planning (Nau, Ghallab, and Traverso 2004; Geffner and Bonet 2013). We use the situation calculus formalism to model the domain in which KiPs are to be executed, including available tasks, contextual properties, tasks'

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



(a) Initial configuration. (b) Actor *act1* gets to *loc03*. (c) Robot *rb1* re-establishes network connectivity to actor *act1*. (d) Robot *rb2* gets to *loc23* so that *act1* is connected once reached the destination.

Figure 1: A train derailment situation; area and context of the intervention.

preconditions and effects, and the initial state. On top of such a logic-based model, we use the IndiGolog high-level agent programming language for the specification of the structure and control flow of KiPs. We customize IndiGolog to monitor the online execution of KiPs and detect potential mismatches between the model and the actual execution. If an exception invalidates the enactment of the KiP being executed, an external state-of-the-art planner is invoked to synthesise a recovery procedure for adapting the faulty process instance. We refer to this adaptive framework for KiPs as SmartPM (Smart Process Management), described in Section 4.

Besides providing the conceptual framework, we validated the approach with a case study based on real KiPs coming from an emergency management domain (Section 5). To do so, we have implemented SmartPM by relying on an existing IndiGolog interpreter (De Giacomo et al. 2009) and the state-of-the-art planning system LPG-td (Gerevini, Saetti, and Serina 2003). In Section 6, we compare SmartPM with other works that exploit AI techniques for enhanced process adaptation. We conclude, in Section 7, by drawing conclusions and discussing limitations and future work.

Before jumping to the technical proposal, let us first start with an overview of our case study and some preliminary notions necessary to understand the rest of the paper.

2 Case Study

Our case study and evaluation involves a disaster management inspired by the WORKPAD project.[1] In particular, it concerns the emergency management scenario described in Figure 1(a), in which a train derailment situation is depicted in a grid-type map. For the sake of simplicity, the train is composed of a locomotive (located at *loc33*) and two coaches (located at *loc32* and *loc31*, resp.).

During the management of complex emergency scenarios, teams of *first responders* act in disaster locations to achieve specific goals. In our train derailment situation, the goal of an incident response plan is to evacuate people from the coaches and take pictures for evaluating possible damages to the locomotive. To that end, a response team is sent to the derailment scene. The team is composed of four first responders, called *actors*, and two *robots*, initially all located at location cell *loc00*. It is assumed that actors are equipped with mobile devices for picking up and executing tasks, and that each provide specific capabilities. For example, actor *act1* is able to extinguish fire and take pictures, while *act2* and *act3* can evacuate people from train coaches. The two robots, in turn, are designed to remove debris from specific locations. When the battery of a robot is discharged, actor *act4* can charge it.

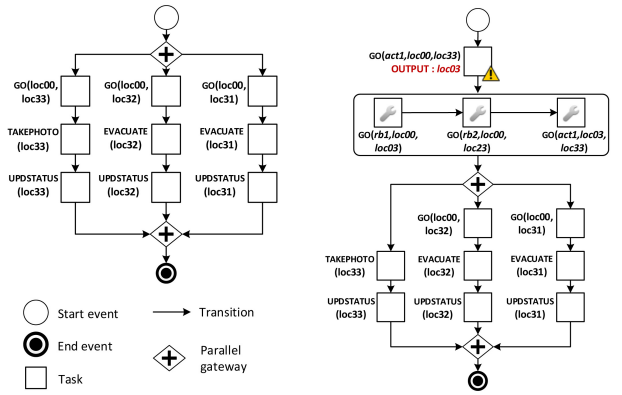
In order to carry on the response plan, all actors and robots ought to be continually inter-connected. The connection between mobile devices is supported by a fixed antenna located at *loc00*, whose range is limited to the dotted squares in Figure 1(a). Such a coverage can be extended by robots *rb1* and *rb2*, which have their own independent (from antenna) connectivity to the network and can act as wireless routers to provide network connection in all adjacent locations.

An incident response plan is defined by a set of activities that are meant to be executed on the field by first responders, and are predicated on specific contexts. Therefore, the information collected on-the-fly is used for defining and configuring at run-time the incident response plan at hand. A possible concrete realization of an incident response plan for our scenario is shown in Figure 2(a), using the Business Process Model and Notation (BPMN).[2] The process is composed of three parallel branches with tasks instructing first responders to act for evacuating people from train coaches, taking pictures of the locomotive, and assessing the gravity of the accident. Due to the high dynamism of the environment, there are a wide range of exceptions that can ensue. Because of that, there is not a clear anticipated correlation between a change in the context and a change in the process.

So, suppose for instance that actor *act1* is sent to the locomotive's location, by assigning to it the task GO(*loc00,loc33*) in the first parallel branch. Unfortunately, however, the actor happens to reach location *loc03* instead. The actor is now located at a different position than the de-

[1] See http://www.dis.uniroma1.it/~workpad/

[2] See www.omg.org/spec/BPMN/



(a) Main process (b) Adapted process

Figure 2: An emergency response plan and its adaptation.

sired one, and most seriously, is out of the network connectivity range (Figure 1(b)). Since all participants need to be continually inter-connected to execute the process, the PMS has to first find a recovery procedure to bring back full connectivity, and then find a way to re-align the process. To that end, provided robots have enough battery charge, the PMS may first instruct the first robot to move to cell *loc03* (Figure 1(c)) in order to re-establish network connection to actor *act1*, and then instruct the second robot to reach location *loc23* in order to extend the network range to cover the locomotive's location *loc33*. Finally, the task GO(*loc03,loc33*) is (re)assigned to actor *act1* (Figure 1(d)). The corresponding updated process is shown in Figure 2(b), with the encircled section being the recovery (adaptation) procedure. Notice that after the recovery procedure, the enactment of the original process can be resumed to its normal flow.

The point is that it is not adequate to assume that the process designer can pre-define all possible recovery activities for dealing with exceptions in domains that are knowledge-intensive as the one just described: the recovery procedure will depend on the actual context (e.g., the positions of participants, the range of the main network, robot's battery levels, whether a location has become dangerous to get it, etc.) and there are too many of them to be considered.

3 Preliminaries

Situation Calculus The *situation calculus* is a logical language designed for representing and reasoning about dynamic domains (Reiter 2001). The dynamic world is seen as progressing through a series of situations as a result of various *actions* being performed. A *situation* s is a first-order term denoting the sequence of actions performed so far. The special constant S_0 stands for the initial situation, where no action has yet occurred, whereas a special binary function symbol $do(a, s)$ denotes the situation resulting from the performance of action a in situation s. Features of the world whose truth value may change from situation to situation are modeled by means of so-called *fluents*. Technically, fluents are predicates taking a situation term as their last argument. For example, fluent $Holding(x, y, s)$ may state that entity x is holding object y at situation s. A special predicate $Poss(a, s)$ is used to state that action a is executable in situation s. We write $\phi(\vec{x})$ to denote a formula whose free variables are among variables $\vec{x}$. A fluent-formula is one whose only situation term mentioned is situation variable s.

Within this language, one can formulate action theories describing how the world changes as the result of the available actions. A *basic action theory* (BAT) (Reiter 2001) $\mathcal{D} = \Sigma \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{poss} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una}$ includes domain-independent foundational axioms to describe the structure of situations (Σ), one successor state axiom per fluent capturing the effects and non-effects of actions ($\mathcal{D}_{ss}$), one precondition axiom per action specifying when the action is executable ($\mathcal{D}_{poss}$), unique name axioms for actions ($\mathcal{D}_{una}$) and initial state axioms describing what is true initially in S_0 ($\mathcal{D}_{S_0}$). In particular, the *successor state axiom* for a fluent $F(\vec{x}, s)$ is an axiom of the form $F(\vec{x}, do(a, s)) \equiv \Psi_F(\vec{x}, a, s)$,[3] where $\Psi_F(\vec{x}, a, s)$ is a fluent-formula characterizing the dynamics of fluent $F(\vec{x}, s)$. Importantly, $\Psi_F(\vec{x}, a, s)$ can accommodate Reiter (2001)'s solution to the frame problem. In addition, precondition axioms are of the form $Poss(a(\vec{x}), s) \equiv \Pi_a(\vec{x}, s)$, where $\Pi_a(\vec{x}, s)$ is a fluent-formula defining the conditions under which action a can be legally executed in situation s. Finally, $\mathcal{D}_{S_0}$ is a collection of first-order sentences whose only situation term mentioned is S_0.

IndiGolog On top of situation calculus action theories, logic-based programming languages can be defined, which, in addition to the primitive actions, allow the definition of complex actions. In particular, we focus on IndiGolog (De Giacomo et al. 2009), the latest in the Golog-like family of programming languages for autonomous agents providing a formal account of interleaved action, sensing, and planning. IndiGolog programs are meant to be executed *online*, in that, at every step, a legal next action is selected for execution, performed in the world, and its sensing outcome gathered. To account for planning, a special look-ahead construct $\Sigma(\delta)$—the search operator—is provided to encode the need for solving (i.e., finding a complete execution) program δ offline.

IndiGolog allows us to define every well-structured process as defined in (van der Aalst et al. 2003); it is equipped with all standard imperative constructs (e.g., sequence, conditional, iteration, etc.) to be used on top of situation calculus primitives actions. An IndiGolog program is meant to run relative to an action theory, providing meaning to primitive actions and conditions in the program. Here we concentrate on the fragment defined by the following constructs:

a	atomic action
$\phi?$	test for a condition
$\delta_1; \delta_2$	sequence
$\pi x.\delta(x)$	nondeterministic choice of argument
δ^*	nondeterministic iteration
if ϕ **then** δ_1 **else** δ_2 **endIf**	conditional
while ϕ **do** δ **endWhile**	while loop
proc $P(\vec{x})$ **do** $\delta(x)$ **endProc**	procedure

[3] Free variables are assumed to be universally quantified.

$\delta_1 \| \delta_2$	concurrency
$\delta_1 \rangle\!\rangle \delta_2$	prioritized concurrency
$\langle \phi \rightarrow \delta \rangle$	interrupt
$\Sigma(\delta)$	lookahead search

Test program $\phi?$ can be executed if condition ϕ holds true, whereas program $\pi x.\delta(x)$ executes program $\delta(x)$ for *some* nondeterministic choice of a binding for variable x, and δ^* executes δ zero, one, or more times. The interleaved concurrent execution of two programs is represented with constructs $\delta_1 \| \delta_2$ and $\delta_1 \rangle\!\rangle \delta_2$; the latter considering δ_1 at higher priority level (i.e., δ_2 can perform a step only if δ_1 is blocked or completed). Finally, interrupt $\langle \phi \rightarrow \delta \rangle$ states that program δ ought to be executed to completion if ϕ happens to become true, whereas $\Sigma(\delta)$ finds and executes a plan—a sequence of actions—that is guaranteed to fully execute δ.

By properly combining prioritized concurrency and interrupts, together with IndiGolog's default online execution style, it is possible to design processes that are sufficiently open and reactive to dynamic environments. Furthermore, by resorting to the search operator, one can specify local places in programs where lookahead reasoning is required. Both aspects will end up being fundamental for our adaptive process management framework in the next section.

Classical Planning Planning systems are problem-solving algorithms that operate on explicit representations of states and actions (Nau, Ghallab, and Traverso 2004; Geffner and Bonet 2013). PDDL (Edelkamp and Hoffmann 2004) is the standard planning representation language; it allows one to formulate a *planning problem* $\mathcal{P} = \langle I, G, \mathcal{P}_D \rangle$, where I is the initial state, G is the goal state, and $\mathcal{P}_D$ is the planning domain. In turn, a planning domain $\mathcal{P}_D$ is built from a set of *propositions* describing the state of the world (a state is characterized by the set of propositions that are true) and a set of *operators* (i.e., actions) that can be executed in the domain. Each operator is characterized by its preconditions and effects, stated in terms of the domain propositions.

There exist several forms of planning in the AI literature. In this paper, we focus on *classical planning*, characterized by fully observable, static, and deterministic domains. A solution for a classical planning problem $\mathcal{P}$ is a sequence of operators—a plan—whose execution transforms the initial state I into a state satisfying the goal G. Such a plan is computed in advance and then carried out (unconditionally). The field of classical planning has experimented huge advances in the last twenty years, leading to a variety of concrete solvers (i.e., planning systems) that are able to create plans with thousands of actions for problems containing hundreds of propositions. In this work, we represent planning domains and planning problems using PDDL 2.2 (Edelkamp and Hoffmann 2004), which includes operators with disjunctive preconditions and derived predicates.

4 The SmartPM Approach

We adopt a *service-based* approach to process management. Thus, *tasks are executed by services*, such as software applications, human actors, robots, etc. Each task can be seen as a single step consuming input data and producing output data.

In this section, we show how one can put together the three AI frameworks described above to build a PMS, which we shall name SmartPM, that is able to not only enact KiPs, but also to *automatically* adapt processes in case of unanticipated exceptions. Intuitively, situation calculus theories will be used to model the contextual information in which the process is meant to run, IndiGolog programs will encode the KiP to be carried out, and planning systems will be used to support the automated adaptation of a process when needed.

SmartPM Basic Action Theory

A BAT for a SmartPM application specifies:

1. the tasks and services of the domain of concern;
2. the support framework for managing the task life-cycle;
3. the contextual setting in which processes operate; and
4. the support framework for the monitoring of processes.

Let us start by describing the first three ones. So, to encode tasks and services, we use five non-fluent predicates:

- *Service*(srv): *srv* is a service (i.e., a process participant);
- *Task*(t): t is a task (e.g., GO(l_1, l_2) or TAKEPHOTO(l) may denote the tasks of navigating or taking pictures);
- *Capability*(c): c is a capability;
- *Provides*(srv, c): service *srv* provides capability c; and
- *Requires*(t, c): task t requires the capability c.

Observe all these predicates are rigid: they do not depend on a situation term and are hence static. A service *srv* is able to perform certain task t iff *srv* provides all capabilities required by the task t. This is captured formally using the following abbreviation:

$$\mathit{Capable}(srv, t) \stackrel{\text{def}}{=} \forall c.\mathit{Requires}(t, c) \supset \mathit{Provides}(srv, c).$$

To talk about concrete runs of tasks, we associate them with unique identifiers. A *task instance* is then a tuple $t : id$, where t is a task and *id* is an identifier.

The life-cycle of tasks involves the execution of four primitive actions executed by the PMS and two exogenous actions arising from services. More concretely, the protocol for a successful execution of a task t goes as follows:

1. First, the PMS assigns task instance $t : id$ to a service *srv* by performing primitive action ASSIGN$(srv, id, t, \vec{o_e})$, where $\vec{o_e}$ is the *expected* (sensing result) output.
2. When a service is ready for task execution, it generates the exogenous action READYTOSTART(srv, id, t).
3. Next, the PMS performs primitive action START(srv, id, t) to authorize the service in question that is formally allowed to start carrying out the task instance.
4. When the service completes the task, it generates the exogenous action FINISHED$(srv, id, t, \vec{o_r})$, with $\vec{o_r}$ representing the physical *actual* outcome returned by the task execution (we use ϵ to denote the empty output).
5. At this point, the PMS updates the properties (i.e., the fluents) to reflect the effects of the task just completed.

6. Finally, the PMS acknowledges the completion of the task and releases the service from the task via primitive actions $\text{ACKCOMPL}(\mathit{srv}, \mathit{id}, t)$ and $\text{RELEASE}(\mathit{srv}, \mathit{id}, t)$.

The intended meaning of all the above actions is captured by means of a set of domain-independent fluents that are used to keep track of the life-cycle of tasks as well as the resource perspective of a process. For example, fluent $\mathit{Free}(\mathit{srv}, s)$ denotes whether a service *srv* is available for task assignments in situation s, whereas $\mathit{ExOut}(\mathit{id}, t, s)$ denotes the expected output of the task; their dynamics are captured via the following successor state axioms (which include Reiter (2001)'s solution to the frame problem):

$$\begin{aligned}&\mathit{Free}(\mathit{srv}, do(a, s)) \equiv\\ &\quad (\exists t, \mathit{id})a = \text{RELEASE}(\mathit{srv}, \mathit{id}, t) \vee\\ &\quad [\mathit{Free}(\mathit{srv}, s) \wedge (\forall t, \mathit{id}, \vec{o_e})a \neq \text{ASSIGN}(\mathit{srv}, \mathit{id}, t, \vec{o_e})];\\ &\mathit{ExOut}(\mathit{id}, t, do(a, s)) = \vec{o} \equiv\\ &\quad (\exists \mathit{srv}, \vec{o_e})a = \text{ASSIGN}(\mathit{srv}, \mathit{id}, t, \vec{o_e}) \wedge \vec{o} = \vec{o_e} \vee\\ &\qquad \mathit{ExOut}(\mathit{id}, t, s) = \vec{o}.\end{aligned}$$

That is, a service is free (for task assignment) after the execution of an action a iff a releases the service from some task assignment, or it was free before the execution of a and a does not assign a task instance to it. Similarly, the expected output of a task instance is determined by the assignment step (and never changes). Initial expected outcomes are initialized to "no output" via an axiom $(\forall t, \mathit{id}).\mathit{ExOut}(\mathit{id}, t, S_0) = \epsilon$ in $\mathcal{D}_{S_0}$. Other similar fluents are used to manage the life-cycle of tasks.

The BAT shall also contain a set of domain-dependent fluents, together with their corresponding precondition and successor state axioms, capturing the contextual scenario in which the process is meant to to be executed. We call them *data fluents*. In general, such fluents will be affected upon the release of an assignment task, that is, whenever a task is considered fully executed. In addition, the actual outcome result of a task is used to define the fluent in question.

Example 1 Consider the following successor state axiom for functional fluent $\mathit{At}(x, s)$ to keep track of the location of entities (e.g., actors and robots) in our emergency scenario:

$$\begin{aligned}&\mathit{At}(x, do(a, s)) = l \equiv\\ &\quad (\exists \mathit{id}, l_s, l_d)a = \text{FINISHED}(x, \mathit{id}, \text{GO}(l_s, l_d), l) \vee\\ &\quad [\mathit{At}(x, s) = l \wedge\\ &\quad (\neg\exists \mathit{id}, l')a = \text{FINISHED}(x, \mathit{id}, \text{GO}(l_s, l_d), l') \wedge l' \neq l].\end{aligned}$$

In words, service x is in location l if x was just released of a task GO whose actual physical outcome result was l. Observe that l, the new location of x, may happen to be different to the expected (destination) location l_d. ■

An important property of the domain in question is that one stating whether a service is connected to the network. Rather than defining a new data fluent, an abbreviation will be enough.

Example 2 The abbreviation $\mathit{Connected}(x, s)$ denotes that service x is within network connectivity range and is defined as follows:

$$\begin{aligned}&\mathit{Connected}(x, s) \stackrel{\text{def}}{=}\\ &\quad \mathit{Robot}(x) \vee \mathit{Covered}(\mathit{At}(x, s)) \vee\\ &\qquad \exists r.\mathit{Robot}(r) \wedge \mathit{Neigh}(\mathit{At}(x, s), \mathit{At}(r, s)).\end{aligned}$$

That is, service x is connected to the network iff x is actually robot (robots have their own connectivity), or x's location is covered by the main base network, or x is adjacent to a robot providing network connectivity in its surroundings. ■

Now, in KiPs, data fluents and abbreviations will be often used for defining the *preconditions* of domain tasks. By doing so, the PMS can reason, at run-time, about the active process instance relative to the current context.

Example 3 The following precondition axiom defines when the PMS can assign a navigation task to a service:

$$\begin{aligned}&\mathit{Poss}(\text{ASSIGN}(\mathit{srv}, \mathit{id}, \text{GO}(l_s, l_d), l_e), s) \equiv\\ &\quad (\mathit{At}(\mathit{srv}, s) = l_s) \wedge \mathit{Connected}(\mathit{srv}, s) \wedge\\ &\quad (l_e = l_d) \wedge \mathit{Capable}(\mathit{srv}, \text{GO}(l_s, l_d)).\end{aligned}$$

That is, the service (e.g., human actor, robot, etc.) ought to be at the source location l_s, connected to the network, and capable of locomotion. Moreover, the expected outcome of the task needs to be the destination location l_d. ■

This concludes the exposition of the first three aspects of a SmartPM action theory, as listed above.

Exception Monitoring We now turn our attention to the mechanism for automatically detecting failures/exceptions. To that end, we leverage on De Giacomo, Reiter, and Soutchanski (1998)'s technique of adaptation from the field of agent-oriented programming, by specializing it to our KiP setting. We consider adaptation as *reducing the gap* between the *expected reality*, the (idealized) model of reality, and the *physical reality*, the real world with the actual conditions and outcomes. A misalignment of the two realities stems from errors or exceptions in the tasks' outcomes, and may require explicit intervention.

The physical reality is captured using data fluents (together with their corresponding successor state axioms) as described above (e.g., fluent $\mathit{At}(x, s)$). The expected reality, in turn, is captured by a set of automatically generated fluents from the data fluents. Technically, for every fluent $F(\vec{x}, s)$, a new fluent $F_{\text{exp}}(\vec{x}, s)$ (F-expected) is used. Intuitively, $F_{\text{exp}}(\vec{x}, s)$ represents the value of $F(\vec{x}, s)$ in the "expected" (or "desired") execution. If $F(\vec{x}, do(a, s)) \equiv \Psi_F(\vec{x}, a, s)$ is F's successor state axiom, its expected version $F_{\text{exp}}(\vec{x}, s)$ is defined as follows:

$$\begin{aligned}&F_{\text{exp}}(\vec{x}, do(a, s)) \equiv\\ &\quad a = \text{ALIGN} \supset F(\vec{x}, s) \vee\\ &\quad [(\exists \mathit{srv}, \mathit{id}, t, \vec{o_r})a = \text{FINISHED}(\mathit{srv}, \mathit{id}, t, \vec{o_r}) \supset\\ &\qquad \Psi^*_F(\vec{x}, \text{FINISHED}(\mathit{srv}, \mathit{id}, t, \mathit{ExOut}(\mathit{id}, t, s)), s)] \vee\\ &\quad [a \neq \text{ALIGN} \wedge (\forall \vec{y})a \neq \text{FINISHED}(\vec{y}) \supset \Psi^*_F(\vec{x}, a, s)].\end{aligned}$$

where $\Psi^*_F(\vec{x}, a, s)$ is obtained by replacing each fluent X in $\Psi_F(\vec{x}, a, s)$ (the right-hand-side formula of F's successor state axioms) with its expected version X_{exp}.

The first disjunct states that the special action ALIGN assigns the actual value of the fluent to its expected value, thus providing a synchronization mechanism between the expected and physical realities. The remaining two disjuncts state together that the dynamics of F_{exp} is that of F *assuming all task outcomes turn out to be the ones expected*. Technically, this is achieved by replacing all data fluents in Ψ_F

with their expected version and the actual outcomes $\vec{o_r}$ of finished tasks with the expected ones (term $ExOut(id, t, s)$).

Example 4 The corresponding (syntactically simplified) expected version for fluent $At_{\text{exp}}(x, s)$ is as follows:

$$\begin{array}{l} At_{\text{exp}}(x, do(a, s)) = l \equiv \\ \quad a = \text{ALIGN} \supset (At(x, s) = l) \vee \\ \quad [(\exists id, l_s, l_d, l_r) a = \text{FINISHED}(x, id, \text{GO}(l_s, l_d), l_r) \supset \\ \qquad l = ExOut(id, \text{GO}(l_s, l_d), s)] \vee \\ \quad [At_{\text{exp}}(x, s) = l \wedge a \neq \text{ALIGN} \wedge \\ \qquad (\forall id, l_s, l_d, l_r) a \neq \text{FINISHED}(x, id, \text{GO}(l_s, l_d), l_r)]. \end{array}$$

Observe that if the action a denotes the completion of a GO task for x with some actual outcome l_r, the new location of x is that expected (denoted by $ExOut(id, \text{GO}(l_s, l_d), s)$). ■

Similarly, for each abbreviation $A(s) \stackrel{\text{def}}{=} \Psi(s)$, one can define corresponding abbreviation $A_{\text{exp}}(s) \stackrel{\text{def}}{=} \Psi^*(s)$, where $\Psi^*(s)$ is obtained by replacing each fluent (or abbreviation) X in $\Psi(s)$ with its expected version X_{exp} and all actual task outcomes $\vec{o_r}$ with their expected ones $ExOut(id, t, s)$. Of course, one can also define special expected fluent or abbreviations that can encode particular expectations. For example, by simply taking $Connected_{\text{exp}}(x, s) \stackrel{\text{def}}{=} \texttt{true}$, we are able to encode the constraint that actors are expected to always be connected.

So, using the data fluents and their expected versions, a misalignment can be recognized. However, it may only be important to check for mismatches among *some* properties of the world. So, we assume that the designer specifies distinguished abbreviation $Misaligned(s)$ to characterize misalignment situations requiring process adaptation. The general form of this abbreviation is as follows:

$$\begin{array}{l} Misaligned(s) \stackrel{\text{def}}{=} \\ \quad \exists \vec{x_1}.\Phi_{X^1}(\vec{x_1}, s) \supset \neg[X^1(\vec{x_1}, s) \equiv X^1_{\text{exp}}(\vec{x_1}, s)] \vee \\ \qquad\qquad \vdots \\ \quad \exists \vec{x_n}.\Phi_{X^n}(\vec{x_n}, s) \supset \neg[X^n(\vec{x_n}, s) \equiv X^n_{\text{exp}}(\vec{x_n}, s)], \end{array}$$

where $X^i(\vec{x_i}, s)$, with $i \in \{1, \ldots, n\}$, are all the data fluents and abbreviations used in the SmartPM application, and $\Phi_{X^i}(\vec{x_i}, s)$ states when fluent/abbreviation $X^i(\vec{x_i}, s)$ needs to be monitored for misalignment between its physical and expected values. We use $\Phi_{X^i}(\vec{x_i}, s) = \texttt{true}$ and $\Phi_{X^i}(\vec{x_i}, s) = \texttt{false}$ to specify permanent and no monitoring, respectively, for the corresponding fluent/abbreviation.

Example 5 Assuming we are only interested in monitoring the location of actors and their connectivity, we would specify the following definition:

$$\begin{array}{l} Misaligned(s) \stackrel{\text{def}}{=} \\ \quad \exists x_1.Actor(x_1) \supset \neg[At(x_1, s) \equiv At_{\text{exp}}(x_1, s)] \vee \\ \quad \exists x_1.Actor(x_1) \supset \\ \qquad \neg[Connected(\vec{x_1}, s) \equiv Connected_{\text{exp}}(\vec{x_1}, s)]. \end{array}$$

Observe the definition is not concerned about exceptions on the location of robots or their connectivity, for example. ■

This concludes the explanation on what type of situation calculus BAT we shall use in a SmartPM application. Let us call this theory $\mathcal{D}_{\text{SmartPM}}$.

SmartPM High-Level Program

Algorithm 1 shows the IndiGolog program for the SmartPM system. The program, as any high-level program, is meant to be executed relative to the basic action theory $\mathcal{D}_{\text{SmartPM}}$ as developed above, which shall give meaning to conditions and primitive statements in the program (i.e., actions).

The top-level part of PMS involves four interrupts running at different priorities, as long as the domain process is yet not finished. The highest two priority programs deal with automated process adaptation. First, if the system has just been adapted, then the two realities—expected and physical—must be aligned, as a new repair plan has been found and a new synchronization point has been reached. Second, the system checks for a misalignment between the two realities, as explained above. If a mismatch is recognized, the adaptation procedure is triggered (see below).

At medium priority, the PMS runs the IndiGolog program reflecting the actual KiP, represented by procedure **Process**() and corresponds to the process depicted in Figure 1(a). Finally, at the lowest priority (when the process cannot advance a step further) the PMS just waits for an exogenous action to arrive from one of the services (e.g., FINISHED signaling the completion of a task). While waiting, the (human) process designer could also manually intervene (for example, by adding new services or updating the capabilities of existing services).

Managing the life-cycle of a task instance—procedure **ManageExecution**()—involves selecting a free service capable of carrying it out, assigning the task to the chosen service, allowing the start of the service, acknowledging its completion and fully releasing the service from the task. Notice the use of the non-deterministic choice of argument $\pi srv.\delta(srv)$ to select an appropriate service.

The most interesting part of the procedure involves procedure **Adapt**(). An adaptation is very simple: *find a sequence of actions that will resolve the misalignment*. This is exactly what the code inside the search operator Σ does: *pick n actions (and execute them) zero, one, or more times such that abbreviation Misaligned*(s) *becomes false.* The action SETMUSTALIGN at the front of the search construct will just make $MustAlign(s)$ true, which will trigger (provided an actual adaptation plan is found) the top-priority program in the main procedure to force an alignment of the two realities. Observe that because the adaptation mechanism runs at higher priority than the actual process, the recovery plan found will be run *before* whatever part of the domain process remains to be executed.

While this specification of the automated adaptation procedure turns out to be extremely clean and simple, the direct use of the native search operator provided by the IndiGolog architecture (De Giacomo et al. 2009) poses serious problems in terms of efficiency. Indeed, the search operator provided by IndiGolog performs basic blind search and as a result is not able to cope with extremely easy adaptation tasks. The important observation to make is that while the search operator is meant to handle *any* IndiGolog high-level program (including ones containing nested search operators), our SmartPM system uses a specific program that encodes a (classical) planning problem. So, leveraging on

Algorithm 1: IndiGolog high-level program for PMS.

Proc **SmartPM**()
  $\langle \neg \mathit{Finished} \wedge \mathit{MustAlign} \rightarrow \text{ALIGN} \rangle \rangle$
  $\langle \neg \mathit{Finished} \wedge \mathit{Misaligned} \rightarrow \mathbf{Adapt}() \rangle \rangle$
  $\langle \neg \mathit{Finished} \rightarrow [\mathbf{Process}(); \text{FINISH}] \rangle \rangle$
  $\langle \neg \mathit{Finished} \rightarrow [\text{WAIT}] \rangle$.
Proc **Adapt**()
  $\Sigma_{\text{LPG}}[\text{SETMUSTALIGN}; (\pi a.a)^*; \neg \mathit{Misaligned}?]$.
Proc **ManageExecution**(*Task*, *id*, *ExpOut*)
  $(\pi\ srv)$.
    $(\mathit{Capable}(srv, \mathit{Task}) \wedge \mathit{Free}(srv))?$;
    ASSIGN(*srv*, *id*, *Task*, *ExpOut*);
    START(*srv*, *id*, *Task*);
    ACKCOMPL(*srv*, *id*, *Task*);
    RELEASE(*srv*, *id*, *Task*)

Proc **Process**()
  [**Branch1**()|| **Branch2**()|| **Branch3**()].
Proc **Branch1**()
  ManageExecution(GO($loc00, loc33$), $id_1, loc33$);
  ManageExecution(TAKEPHOTO($loc33$), id_2, ok);
  ManageExecution(UPDATESTATUS($loc33$), id_3, ok)
Proc **Branch2**()
  ManageExecution(GO($loc00, loc32$), $id_4, loc32$);
  ManageExecution(EVACUATE($loc32$), id_5, ok);
  ManageExecution(UPDATESTATUS($loc32$), id_6, ok)
Proc **Branch3**()
  ManageExecution(GO($loc00, loc31$), $id_7, loc31$);
  ManageExecution(EVACUATE($loc31$), id_8, ok);
  ManageExecution(UPDATESTATUS($loc31$), id_9, ok)

the recent progress of classical planning systems, we implement the search operator call in procedure **Adapt**(), by using an off-the-shelf planner to synthesise the recovery plan, and then fit such a plan back into the IndiGolog framework. In this work, we used the LPG-td system (Gerevini, Saetti, and Serina 2003), one of the many state-of-the-art planning systems available. The basic search scheme of LPG-td is inspired by Walksat (Selman, Kautz, and Cohen 1994), an efficient procedure for solving SAT-problems; as expected, it outperforms the blind search operator by several orders of magnitudes. Nonetheless, our approach is orthogonal to other planning systems.

We do not go over all the details on how the interface between IndiGolog and LPG-td has been implemented but just go over the main ingredients. First of all, given a BAT for a KiP application, the corresponding PDDL planning domain is built offline and stored. Because we are not concerned with the exogenous actions generated by services to acknowledge the start and termination of assigned processes (i.e., actions READYTOSTART and FINISHED), we do not model the full PMS assign-start-acknowledge-release task life-cycle, but just encapsulates them all in the actual name of the task being handled (e.g., GO). By doing that, we assume that the life-cycle of a task instance will follow its normal evolution. The SmartPM BAT will define a form of tasks and services repository, which may include entities not used in the current running process. So, the PDDL domain for our case study will contain, among others, the following action schema:

```
(:action go
 :parameters (?x - srv ?from - loc
              ?to - loc)
 :precondition (and (free ?x)
                (provides ?x movement)
                (at ?x ?from) (connected ?x))
 :effect (and
          (not (at ?x ?from)) (at ?x ?to)))
```

Note that the task-action in the planning system will contain the service in charge and its expected effects.

Then, whenever a search operator adaptation program is called in procedure **Adapt**(), the current situation of the IndiGolog system is encoded as a planning initial state (as the set of fluents that are true) and formula $\mathit{Misaligned}(s)$ is encoded as a goal state (by taking the collection of relevant fluents to be as their expected versions). Those two states, together with the planning domain already pre-computed, are then passed to the LPG-td system.

Finally, if the planner finds a plan that brings about the (desired) expected reality, such a plan—built from task names only—is translated into the typical assign-start-acknowledge-release task life-cycle IndiGolog program. As stated above, the plan will run *before* the actual domain process, which shall resume then, hopefully from the expected reality. In our running example, the full IndiGolog program would encode the KiP depicted in Figure 2(b).

This concludes the complete overview of SmartPM, built from a BAT and an IndiGolog high-level program that uses a classical planner for its specific (classical planning) search operator implementation. Let us now look at some empirical evaluation of such a system.

5 Validation

In order to investigate the feasibility of the SmartPM approach, we performed experiments to study the time required for synthesizing a recovery plan for different adaptation problems. We made our tests by using the LPG-td planner (Gerevini, Saetti, and Serina 2003). We chose LPG-td as *(i)* it treats most of PDDL 2.2 features; and *(ii)* it has been developed in two versions: a version tailored to computation speed, named LPG-td.speed, which produces *sub-optimal plans* that do not prove any guarantee other than the correctness of the solution, and a version tailored for *plan quality*, named LPG-td.quality. LPG-td.quality differs from LPG-td.speed basically in that it does not stop when the first plan is found, but continues until a stopping quality criterion is met. In our experiments, the criterion for plan quality was set to minimal plan length.

The experimental setup was performed with the test case shown in our train emergency running example. We used a tasks repository containing 20 different emergency management tasks, annotated with 28 relational predicates, 2 derived predicates, and 4 functional numeric fluents. We provided 185 different planning problems of different complexity, by manipulating ad-hoc the values of the initial state and the goal in order to devise adaptation problems of growing complexity. The results are reported in Table 1. Each row summarizes the results of various planning instances ($\#I$ instances). Column L-RP indicates the length of the shortest

Table 1: Performances of LPG-td. L-RP = length of the shortest recovery plan; #I = number of problem instances; AT-SOS/AL-SOS = average time/length for sub-optimal plan; and AT-QS = average time for quality plan.

L-RP	#I	AT-SOS	AL-SOS	AT-QS
1	29	6.769	3	7.768
2	36	7.213	3	16.865
3	32	7.846	4	24.123
4	25	8.128	5	37.017
5	21	8.598	8	39.484
6	17	8.736	9	52.421
7	13	9.188	13	73.526
8	12	9.953	14	81.414

plans for all instances. Column AL-SOS indicates the average number of actions of the sub-optimal solutions found by LPG-td.speed; whereas column AT-SOS reports the time it took to find such plan solutions. Those plans will generally include more tasks than the ones strictly needed. Finally, column AT-QS indicates the average time taken by LPG-td.quality to produce a quality plan. For example, on 21 different planning problems requiring a recovery procedure of length 5, the LPG-td planner is able to find, on average, a sub-optimal plan in 8.598 seconds (with an average of three tasks more) and a quality plan (of exactly 5 tasks necessary for recovery) in 39.484 seconds. Consequently, the approach is feasible for medium-sized dynamic processes used in practice.[4]

Effectiveness of SmartPM in Adapting Processes

An important aspect to consider during the development of a PMS with adaptation features concerns its *effectiveness* in supporting process models having control flows with different structures. We define effectiveness as the *ability of a PMS to complete the execution of a process model (i.e., to execute all the tasks involved in a path from the start event to the end event) by adapting automatically its running process instance if some failure/exception arises, without the need of any manual intervention of the process designer at run-time*.

To evaluate the effectiveness of SmartPM, we developed a software module named the SmartPM *Simulator*, which is able to automatically build IndiGolog processes and corresponding $\mathcal{D}_{\text{SmartPM}}$ theories, by simulating their execution on the basis of some customizable parameters:

- *Structure of the control flow*, with tasks organized in sequence or in 3 or 5 parallel branches.
- *Tasks repository size*, equal to 25, 50, or 75 tasks.
- *Number of available services* in the initial situation for task assignment. We fixed this value to 5 services.
- *Number of task preconditions/effects*; we allowed the generation of tasks having a maximum of 5 conditions in the precondition/effect axioms.
- *Number of available fluents*; we allowed the generation of 50 relevant data fluents that may assume boolean values. For each data fluent, the SmartPM Simulator automatically builds the corresponding expected fluent for monitoring possible tasks failures.
- *Percentage of tasks failures* This parameter may assume two possible values (30% or 70%), and affects the percentage of tasks error during the process execution.
- *Percentage of capabilities coverage* (30% or 70%), that affects the ability of each available service to execute the tasks stored in the repository.

Given *(i)* a specific structure of the control flow, *(ii)* a fixed percentage of capabilities coverage and *(iii)* of tasks failures, for each possible size of the tasks repository the SmartPM *Simulator* generated 100 process models with control flows composed respectively by 5 to 25 tasks randomly picked from the tasks repository. In total, we tested 3600 process models. Test results are shown in Figure 3. Collected data are organized in 4 diagrams obtained by combining the percentage of failures (FP) with the the percentage of capabilities coverage (SC). Each bar corresponds to the enactment of 100 different process models with a fixed size of the tasks repository and a specific structure of the control flow.

The analysis of the performed tests points out some interesting aspect. For example, let us consider the diagram in Figure 3(a), that shows the effectiveness of SmartPM in executing processes with a FP equal to 30% and a SC fixed to 30%. When the size of the tasks repository is equal to 75, the effectiveness of SmartPM in executing 100 process models composed by a sequence of tasks is equal to 82%. This means that 82 processes out of 100 that were executed have been correctly enacted and completed, while for the other 18 processes the system did not find any recovery plan for dealing with an exception occurrence. We can note that the effectiveness decreases if the instances have tasks organized in three parallel branches (79%) and in five parallel branches (75%). In general, the effectiveness of the SmartPM system decreases as the number of parallel branches increases, since more services are possibly involved at the same time for tasks execution, by letting only few services available for process adaptation and recovery. Furthermore: *(i)* when the FP increases, the effectiveness of the SmartPM system decreases; *(ii)* when the SC increases, the effectiveness of the SmartPM system increases, because there are more possibilities for a task in the repository to be selected by an available service.

To sum up, the execution of 3600 process models with different structures was a valid test for measuring the effectiveness of SmartPM, that was able to complete 2537 process instances, corresponding to an effectiveness of about 70.5%.

6 Related Work

The AI community has been involved with research on process management for several decades, and AI technologies can play an important role in the construction of PMS engines that manage complex processes, while remaining robust, reactive, and adaptive in the face of both environmental and tasking changes (Myers and Berry 1998). One of the

[4] We did our tests by using an Intel U7300 CPU 1.30GHz Dual Core, 4GB RAM machine.

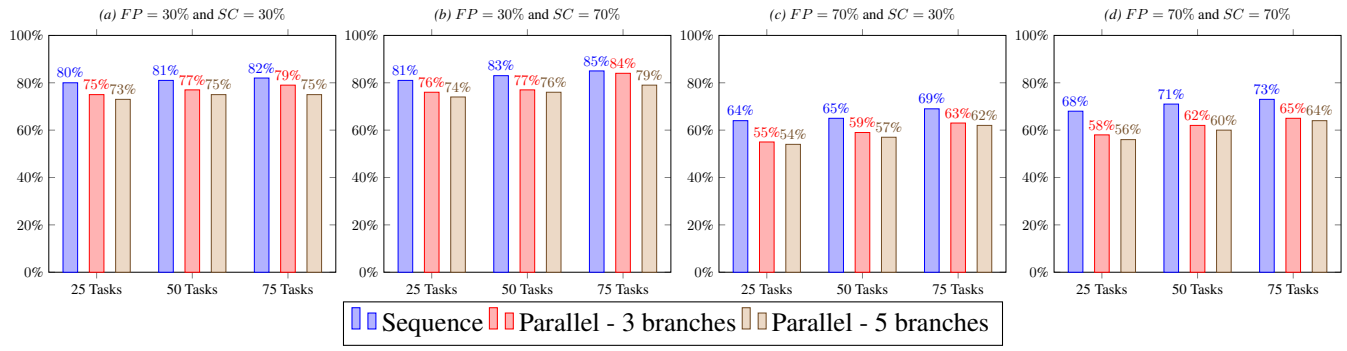


Figure 3: Analysis of the SmartPM effectiveness. The x-axis states the size of the tasks repository, while the y-axis indicates the effectiveness in executing process models with a specific structure of the control flow.

first works dealing with this research challenge is (Beckstein and Klausner 1999). It discusses at high level how the use of an intelligent assistant based on planning techniques may suggest compensation procedures or the re-execution of activities if some expected failure arises during the process execution. In (Jarvis et al. 1999) the authors describe how planning can be interleaved with process execution and plan refinement, and investigates plan patching and plan repair as means to enhance flexibility and responsiveness.

A goal-based approach for enabling automated process instance change in case of emerging exceptions is shown in (Gajewski et al. 2005). If a task failure occurs at run-time and leads to a process goal violation, a multi-step procedure is activated. It includes the termination of the failed task, the sound suspension of the process, the automatic generation (through the use of a partial-order planner) of a new complete process definition that complies with the process goal and the adequate process resumption. A similar approach is proposed in (Ferreira and Ferreira 2006). The approach is based on learning business activities as planning operators and feeding them to a planner that generates a candidate process model that is able of achieving some business goals. If an activity fails during the process execution at run-time, an alternative candidate plan is provided on the same business goals. The major issue of (Gajewski et al. 2005; Ferreira and Ferreira 2006) lies in the replanning stage used for adapting a faulty process instance. In fact, it forces to completely re-define the process specification at run-time when the process goal changes (due to some activity failure), by completely revolutionizing the work-list of tasks assigned to the process participants (that are often humans). On the contrary, our approach adapts a running process instance by modifying only those parts of the process that need to be changed/adapted and keeps other parts stable.

A work dealing with process interference is that of van Beest et al. (2014). Process interference is a situation that happens when several concurrent business processes depending on some common data are executed in a highly distributed environment. During the processes execution, it may happen that some of these data are modified causing unexpected or wrong business outcomes. To overcome this limitation, the work van Beest et al. proposes a run-time mechanism which uses *(i) Dependency Scopes* for identifying critical parts of the processes whose correct execution depends on some shared variables; and *(ii) Intervention Processes* for solving the potential inconsistencies generated from the interference, which are automatically synthesised through a domain independent planner based on constraint techniques. While closely related to van Beest's work, our account deals with changes in a more abstract and domain-independent way, by just checking misalignment between expected/physical realities. Conversely, van Beest's work requires specification of a (domain-dependent) adaptation policy, based on volatile variables and when changes to them become relevant.

7 Conclusion

As discussed in this paper, SmartPM defines a general approach, a concrete framework and a prototype PMS for automated adaptation of processes. Our purpose was to demonstrate that the combination of procedural and imperative models with declarative elements, along with the exploitation of techniques from the field of AI such as situation calculus, IndiGolog and classical planning, can increase the ability of existing PMSs of supporting and adapting KiPs.

The adaptation mechanism is based on execution monitoring for detecting failures and context changes at run-time, without requiring to predefine any specific adaptation policy or exception handler at design-time (as most of the current approaches do). The use of classical planning techniques for the synthesis of the recovery procedure has a twofold consequence. On the one hand, we can exploit the good performance of current state-of-the-art planners to solve medium-sized real-world problems as used in practice. On the other hand, classical planning imposes some restrictions for addressing more expressive problems, including incomplete information, preferences and multiple task effects.

We note that, besides task failures, the execution of a KiP could be also jeopardized by the occurrence of *exogenous events*—e.g., a sudden fire outbreak in a coach—that could change, in asynchronous manner, some *contextual* properties of the scenario in which the process is under execution, hence possibly requiring the KiP to be adapted. Future work

is required to extend our approach to deal such disruptive exogenous events.

The current prototype of SmartPM is developed to be effectively used by process designers and practitioners.[5] Users define processes in the well-known BPMN language, enriched with semantic annotations for expressing properties of tasks, which allow our interpreter to derive the IndiGolog program representing the process. Interfaces with human actors (as specific graphical user applications in Java) and software services (through Web service technologies) allow the core system to be effectively used for enacting processes. Our future work will also improve the current prototype in order to be compliant with many other technologies adopted by process practitioners, e.g., RESTful services and HTML-based user interfaces.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. The first two authors were supported by Sapienza - Universitá di Roma through the grant SUPER and by Regione Lazio and FILAS through the "Progetto Sensoristica Interconnessa per la Sicurezza". The last author was supported by the Australian Research Council under a Discovery Project DP120100332.

References

Beckstein, C., and Klausner, J. 1999. A Meta Level Architecture for Workflow Management. *Journal of Integrated Design and Process Science* 3(1):15–26.

De Giacomo, G.; Lespérance, Y.; Levesque, H.; and Sardina, S. 2009. IndiGolog: A High-Level Programming Language for Embedded Reasoning Agents. In *Multi-Agent Programming*. Springer US. 31–72.

De Giacomo, G.; Reiter, R.; and Soutchanski, M. 1998. Execution Monitoring of High-Level Robot Programs. In *6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, 453–465.

Di Ciccio, C.; Marrella, A.; and Russo, A. 2012. Knowledge-intensive Processes: An Overview of Contemporary Approaches. In *1st International Workshop on Knowledge-intensive Business Processes (KiBP'12)*, 33–47.

Dumas, M.; van der Aalst, W. M. P.; and ter Hofstede, A. H. M. 2005. *Process-aware Information Systems: Bridging People and Software Through Process Technology*. Wiley Online Library.

Edelkamp, S., and Hoffmann, J. 2004. PDDL2.2: The Language for the Classical Part of the 4th International Planning Competition. Technical report, Albert-Ludwigs-Universität Freiburg, Institut für Informatik.

Ferreira, H., and Ferreira, D. 2006. An Integrated Life Cycle for Workflow Management Based on Learning and Planning. *International Journal of Cooperative Information Systems* 15(4):485–505.

Gajewski, M.; Meyer, H.; Momotko, M.; Schuschel, H.; and Weske, M. 2005. Dynamic Failure Recovery of Generated Workflows. In *16th International Conference on Database and Expert Systems Applications (DEXA'05)*, 982–986.

Geffner, H., and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning Through Stochastic Local Search and Temporal Action Graphs in LPG. *Journal of Artificial Intelligence Research* 20:239–290.

Helal, S.; Mann, W.; El-Zabadani, H.; King, J.; Kaddoura, Y.; and Jansen, E. 2005. The Gator-Tech Smart House: A Programmable Pervasive Space. *Computer* 38(3):50–60.

Jarvis, P.; Moore, J.; Stader, J.; Macintosh, A.; du Mont, A. C.; and Chung, P. 1999. Exploiting AI Technologies to Realise Adaptive Workflow Systems. In *AAAI Workshop on Agent-Based Systems in the Business Context*, 25–34.

Lenz, R., and Reichert, M. 2007. IT support for healthcare processes - premises, challenges, perspectives. *Data Knowledge Engineering* 61(1):39–58.

Marrella, A.; Russo, A.; and Mecella, M. 2012. Planlets: Automatically Recovering Dynamic Processes in YAWL. In *20th International Conference on Cooperative Information Systems (CoopIS'12)*, 268–286.

Myers, K., and Berry, P. 1998. Workflow Management Systems: An AI Perspective. *AIC-SRI report*.

Nau, D.; Ghallab, M.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Reichert, M., and Weber, B. 2012. *Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies*. Springer.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Sadiq, S. W.; Sadiq, W.; and Orlowska, M. E. 2001. Pockets of Flexibility in Workflow Specification. In *20th International Conference on Conceptual Modeling: Conceptual Modeling (ER'01)*, 513–526.

Selman, B.; Kautz, H. A.; and Cohen, B. 1994. Noise Strategies for Improving Local Search. In *12th National Conference on Artificial Intelligence (AAAI-94)*, 337–343.

van Beest, N.; Kaldeli, E.; Bulanov, P.; Wortmann, J.; and Lazovik, A. 2014. Automated Runtime Repair of Business Processes. *Information Systems* 39:45–79.

van der Aalst, W. M. P.; ter Hofstede, A. H. M.; Kiepuszewski, B.; and Barros, A. P. 2003. Workflow patterns. *Distributed Parallel Databases* 14(1):5–51.

van der Aalst, W. M. P.; ter Hofstede, A. H. M.; and Weske, M. 2003. Business Process Management: A Survey. In *1st International Conference on Business Process Management (BPM'03)*, 1–12.

Weske, M. 2012. *Business Process Management: Concepts, Languages, Architectures*. Springer.

[5] See www.dis.uniroma1.it/~smartpm

Tweety: A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and Knowledge Representation

Matthias Thimm
Institute for Web Science and Technologies (WeST)
University of Koblenz-Landau
Germany

Abstract

This paper presents Tweety, an open source project for scientific experimentation on logical aspects of artificial intelligence and particularly knowledge representation. Tweety provides a general framework for implementing and testing knowledge representation formalisms in a way that is familiar to researchers used to logical formalizations. This framework is very general, widely applicable, and can be used to implement a variety of knowledge representation formalisms from classical logics, over logic programming and computational models for argumentation, to probabilistic modeling approaches. Tweety already contains over 15 different knowledge representation formalisms and allows easy computation of examples, comparison of algorithms and approaches, and benchmark tests. This paper gives an overview on the technical architecture of Tweety and a description of its different libraries. We also provide two case studies that show how Tweety can be used for empirical evaluation of different problems in artificial intelligence.

1 Introduction

Knowledge Representation and Reasoning (KR) [Brachman and Levesque, 2004] is an important subfield in Artificial Intelligence (AI) that deals with issues regarding formalizing knowledge in such a way that machines can read, understand, and reason with it. Nowadays, KR has a lot of applications within, e. g., the semantic web [Antoniou and van Harmelen, 2004], as a lot of work on description logics [Baader et al., 2003] and ontologies originate from this field (at least the technical or computer-science-oriented perspectives on those). Apart from that, more fundamental work in KR deals with issues regarding uncertainty of beliefs, dynamics of belief, and defeasible reasoning. Many branches of research in knowledge representation and reasoning is theoretical in nature and researchers usually do not put effort in implementation and empirical evaluation. To address this issue we present in this field study the Tweety libraries for logical aspects of artificial intelligence and knowledge representation.

Approaches to knowledge representation follow almost always a specific pattern. Starting from a formal syntax one can build formulas which are collected in knowledge bases. Using knowledge bases one can derive new information using either the underlying semantics of the language or a specific reasoner. For example, propositional logic is the most basic form for knowledge representation. Given some set of propositions (or atoms) one can build complex formulas using disjunction, conjunction, or negation. A set of propositional formulas, i. e., a knowledge base, can be used to derive new propositional formulas as conclusions. For instance, this can be done using the standard model-theoretic semantics of propositional logic or more sophisticated reasoning techniques such as paraconsistent reasoning. Most logical approaches to knowledge representation such as first-order logic, description logics, defeasible logics, default logics, probabilistic logics, fuzzy logics, etc. follow this pattern. Moreover, many other formalisms which are not so obviously rooted in logic such as abstract argumentation or Bayes nets can also be cast into this framework. For example, for abstract argumentation frameworks [Dung, 1995], a knowledge base is given by a conjunction of attack statements between arguments and different kinds of semantics such as grounded or stable semantics determine how sets of arguments can be derived from a knowledge base.

The Tweety libraries support the implementation of such approaches by providing a couple of abstract classes and interfaces for components such as `Formula`, `BeliefBase`, and `Reasoner`. Furthermore, many strictly logic-based approaches to knowledge representation can also utilize further classes such as `Predicate`, `Atom`, and `Variable`, to name just a few. Currently, Tweety already contains implementations of over 15 different approaches to knowledge representation such as propositional logic, first-order logic, several approaches to probabilistic logics, and several approaches to computational models of argumentation.

In this paper, besides giving an overview on the technical details of Tweety and its libraries, we also report on two case studies that use Tweety as a framework for experimentation and empirical evaluation. The first study is on inconsistency measurement for probabilistic logics [Thimm, 2011; 2013b]. In general, probabilistic logics are concerned with using quantitative uncertainty for non-monotonic reasoning. Naturally, these approaches are computationally hard and not easy to understand, as the underlying reasoning mechanisms are quite complicated. Consequently, implementa-

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

tions serve well to understand examples and to (in-)validate conjectures. Our second case study is about strategic argumentation in multi-agent systems [Thimm and Garcia, 2010; Rienstra, Thimm, and Oren, 2013]. Similarly, when defining agent models and negotiation strategies in such an environment, effects that occur on a larger scale are hard to predict by hand. Moreover, just an analytical evaluation of different negotiation strategies is often also simply too weak to provide meaningful insights [Rienstra, Thimm, and Oren, 2013]. For that reason Tweety can also be used as a tool for empirical evaluation as it has been done in [Rienstra, Thimm, and Oren, 2013] to provide average performance results on a series of experiment runs in random settings. Further works that use Tweety for implementing knowledge representation formalisms or for empirical evaluation are, e. g., [Thimm and Kern-Isberner, 2008; Thimm and Garcia, 2010; Krümpelmann et al., 2011; Thimm, 2011; Kern-Isberner and Thimm, 2012; Thimm, 2012; 2013b; Rienstra, Thimm, and Oren, 2013; Thimm, 2013a; Krümpelmann and Kern-Isberner, 2012].

The rest of this paper is organized as follows. Section 2 gives an overview on the architecture of Tweety and Section 3 presents some technical details on its different libraries. In Section 4 two case studies are presented that show how Tweety can be used for evaluation in scientific research. Section 5 concludes with a summary and pointers to future work.

2 Technical Overview

Tweety is organized as a modular collection of Java libraries with a clear dependence structure. The programming language Java has been chosen as it is easy to understand, commonly used, and platform-independent. Each knowledge representation formalism has a dedicated Tweety library (ranging from a library on propositional logic to libraries on computational models of argumentation) which provides implementations for both syntactic and semantic constructs of the given formalism as well as reasoning capabilities. Several libraries provide basic functionalities that can be used in other projects. Among those is the *Tweety Core* library which contains abstract classes and interfaces for all kinds of knowledge representation formalisms. Furthermore, the library *Math* contains classes for dealing with mathematical sub-problems that often occur, in particular, in probabilistic approaches to reasoning. Most other Tweety projects deal with specific approaches to knowledge representation. In the next section, we have a closer look on the individual libraries.

Each Tweety library is organized as a Maven[1] project (Maven is a tool for organizing dependencies between projects, building, and deploying). Most libraries can be used right away as they only have dependencies to other Tweety libraries. Some libraries provide bridges to third-party libraries such as numerical optimization solvers which are not automatically found by Maven and have to be installed beforehand. However, all necessary third-party libraries can be installed by executing a single install file located within the Tweety distribution.

In order to use and develop with Tweety we recommend using the Eclipse IDE[2] and its Maven plugin[3]. As all Tweety libraries are organized as Maven projects they can all be easily imported and used for other projects within Eclipse. Furthermore, pre-compiled JARs for every library can be downloaded from the Tweety homepage[4] and directly used in other projects. A third way of using the functionalities of Tweety is by using its *Command Line Interface* which is currently under development.

Currently, Tweety contains 161 Java packages which themselves contain 794 Java classes in overall 87421 lines of code. The average cyclomatic complexity number per method (CCN) [McCabe, 1976] is 2.87. This means, that every method roughly contains two to three if-else statements thus reducing complexity on a method-basis and emphasizing the modular nature of Tweety. Furthermore, the average code-to-comment ratio in Tweety is 2.13, meaning that for roughly every two lines of code one line of comment is given.

3 Libraries

In the following we give a detailed description of the currently available libraries within Tweety. An overview of these libraries is given in Table 1 which provides both the name of a library and its Java root package name. Furthermore, the final column lists references to original literature and the implemented reasoning mechanisms and solvers. There, a dagger (†) indicates that a the particular reasoning mechanism has been directly implemented from the original literature, a double dagger (‡) means that a wrapper for the existing original implementation is provided, and an asterisk (*) refers to related literature.

General Libraries

The *General* libraries of Tweety provide basic functionalities and utility classes for all other Tweety classes.

Tweety Core The *Tweety Core* library contains abstract classes and interfaces for various knowledge representation concepts. Among the most important ones are

`Formula` A formula of a representation formalism.

`BeliefBase` Some structure containing beliefs.

`BeliefSet` A set of beliefs, i. e., a set of formulas, it is the most commonly used class derived from `BeliefBase`. Please note that we follow the Java guideline for naming a class containing a set of beliefs a *belief set* (it contains a finite unordered set of elements), opposed to the naming convention in belief dynamics where a *belief set* is usually deductively closed. In terms of belief dynamics research the class `BeliefSet` actually represents a *belief base*.

`Signature` The signature of a representation formalism.

[1] http://maven.apache.org

[2] http://www.eclipse.org

[3] http://maven.apache.org/eclipse-plugin.html

[4] http://www.mthimm.de/projects/tweety/

Library	Project root package	Notes
General Libraries		
Tweety Core	`net.sf.tweety`	
Command Line Interface	`n.s.t.cli`	
Plugin	`n.s.t.plugin`	*JSPF[a]
Math	`n.s.t.math`	‡lpsolve[b], ‡OpenOpt[c], ‡Apache simplex[d], ‡Choco[e], †Gradient descent
Graphs	`n.s.t.graphs`	
Logic Libraries		
Logic Commons	`n.s.t.logics.commons`	†[Grant and Hunter, 2006], †[Hunter and Konieczny, 2006]
Propositional Logic	`n.s.t.logics.pl`	‡Sat4j[f]
First-Order Logic	`n.s.t.logics.fol`	*[Brachman and Levesque, 2004]
Conditional Logic	`n.s.t.logics.cl`	†[Pearl, 1990], †[Kern-Isberner, 2001]
Relational Conditional Logic	`n.s.t.logics.rcl`	†[Kern-Isberner and Thimm, 2012]
Probabilistic Conditional Logic	`n.s.t.logics.pcl`	†[Rödder, 2000], †[Thimm, 2013b], †[Potyka, 2014]
Relational Prob. Conditional Logic	`n.s.t.logics.rpcl`	†[Kern-Isberner and Thimm, 2010]
Markov Logic	`n.s.t.logics.ml`	†[Richardson and Domingos, 2006] ‡Alchemy[g]
Epistemic Logic	`n.s.t.logics.el`	*[Fagin et al., 1995]
Description Logic	`n.s.t.logics.dl`	*[Baader et al., 2003]
Logic Translators	`n.s.t.logics.translators`	
Logic Programming Libraries		
Answer Set Programming	`n.s.t.lp.asp`	‡Clingo[h], ‡DLV[i]
Dynamics in Answer Set Programming	`n.s.t.lp.asp.beliefdynamics`	†[Krümpelmann and Kern-Isberner, 2012]
Nested Logic Programming	`n.s.t.lp.nlp`	*[Lifschitz, Tang, and Turner, 1999]
Argumentation Libraries		
Abstract Argumentation	`n.s.t.arg.dung`	†[Dung, 1995], †[Baroni, Caminada, and Giacomin, 2011]
Deductive Argumentation	`n.s.t.arg.deductive`	†[Besnard and Hunter, 2001], †[Besnard and Hunter, 2006]
Structured Argumentation Frameworks	`n.s.t.arg.saf`	†[Thimm and Garcia, 2010]
Defeasible Logic Programming	`n.s.t.arg.delp`	†[Garcia and Simari, 2004], †[Stolzenburg et al., 2003]
Logic Programming Argumentation	`n.s.t.arg.lp`	†[Schweimeier and Schroeder, 2003]
Probabilistic Argumentation	`n.s.t.arg.prob`	†[Thimm, 2012]
Agent Libraries		
Agents	`n.s.t.agents`	*[Weiss, 2013]
Dialogues	`n.s.t.agents.dialogues`	†[Thimm and Garcia, 2010], †[Rienstra, Thimm, and Oren, 2013]
Other Libraries		
Action and Change	`n.s.t.action`	†[Gelfond and Lifschitz, 1998]
Belief Dynamics	`n.s.t.beliefdynamics`	†[Hansson, 2001], †[Fermé and Hansson, 1999], †[Krümpelmann et al., 2011]
Machine Learning	`n.s.t.machinelearning`	‡LIBSVM[j]
Preferences	`n.s.t.preferences`	*[Walsh, 2007], †[Thimm, 2013a]

Table 1: Overview on the Tweety libraries (the prefix `n.s.t` stands for `net.sf.tweety`)

[a] https://code.google.com/p/jspf/
[b] http://lpsolve.sourceforge.net
[c] http://openopt.org
[d] http://commons.apache.org/math
[e] http://www.emn.fr/z-info/choco-solver/
[f] http://www.sat4j.org
[g] http://alchemy.cs.washington.edu
[h] http://potassco.sourceforge.net
[i] http://www.dlvsystem.com
[j] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

`Interpretation` An interpretation that evaluates the truth of formulas.

`Reasoner` Implements a specific reasoning strategy to answer queries for a representation formalism.

`Parser, Writer` For reading/writing formulas and belief sets.

Most other Tweety libraries provide specific implementations of the above abstract classes and interfaces for their specific representation formalisms. For example, the library *Propositional Logic* implements `Formula` by `PropositionalFormula` (which is recursively defined using conjunction, disjunction, and negation) and `Interpretation` by `PossibleWorld`. In this way, the classical approach to formally define a logical language via syntax and semantics has a one-to-one correspondence with its implementation in Tweety.

Besides the above mentioned abstract classes and interfaces, *Tweety Core* provides abstract implementations of several other knowledge representation concepts and several utility classes for working with sets, subsets, vectors, and general rules.

Plugin The *Plugin* library provides classes for implementing Tweety plugins that can be used by, e. g., the *Command Line Interface*. This library makes use of the *Java Simple Plugin Framework (JSPF)*[5]. Using these classes one can encapsulate the functionalities of a specific knowledge representation formalism and expose them in the same way to user interfaces. The most important class is the abstract class `AbstractTweetyPlugin` which is the basis for developing plugins. Please note that the *Plugin* library is currently in an experimental phase.

Command Line Interface All Tweety libraries can be accessed programmatically in Java through their API (*Application Programming Interface*). However, for non-programmers this way of utilizing the libraries is not very convenient. Using the *Plugin* library the *Command Line Interface* library provides a general command line interface for many Tweety libraries. Every library can expose its functionality through a Tweety plugin that can be plugged into the command line interface and accessed in a uniform way. Please note that the *Command Line Interface* library is currently in an experimental phase.

Math Many algorithms for knowledge representation and reasoning are based on mathematical methods such as optimization techniques. The *Math* library encapsulates those mathematical methods and exposes them through simple interfaces to other libraries for realizing these algorithms. At the core, the *Math* library contains classes for representing mathematical terms (such as `Constant`, `Variable`, `Product`, `Logarithm`) and statements (such as `Equation`). Using these constructs one can represent, e. g., constraint satisfaction problems (`ConstraintSatisfactionProblem`) and optimization problems (`OptimizationProblem`). Through the `Solver` interface the *Math* library provides bridges to several third-party solvers such as the *ApacheCommons* Simplex-algorithm[6], the *OpenOpt* solvers[7], or Choco[8].

Graphs The *Graphs* library contains a simple graph implementation with utility functions as it can be used, e. g., to represent abstract argumentation frameworks (see *Abstract Argumentation* library).

Logic Libraries

The *Tweety Logic* libraries (located under the package `net.sf.tweety.logics`) provide implementations for various knowledge representation formalisms based on classical logics (propositional logic and first-order logic) and non-classical logics such as conditional logic, probabilistic logics, epistemic logics, or description logic. Each library follows a strict approach in defining the formalism by implementing the abstract classes and interfaces `Formula`, `BeliefBase`, `Interpretation`,... from the *Tweety Core* library. Each library contains a sub-package `syntax` which contains the elements to construct formulas of the formalism and a sub-package `semantics` which contains elements for realizing the semantics of the formalism. Besides these two common sub-packages many libraries also contain parsers for reading formulas from file and reasoner that implement a specific reasoning approach.

Logic Commons The *Logic Commons* library contains abstract classes and interfaces which further refine the general `Formula` interface from the *Tweety Core* library. Among these refinements are several concepts that are shared among a great number of knowledge representation formalism such as `Predicate`, `Variable` or `Atom`.

Propositional Logic The *Propositional Logic* library provides an implementation of classical propositional logic. Propositional formulas can be constructed using, e. g., classes `Conjunction` or `Disjunction` and propositional formulas can be put into a knowledge base of type `PlBeliefSet`. Currently, the *Propositional Logic* library supports two different reasoners. The first is a simple brute force approach that directly follows the definition of classical entailment, i. e., in order to prove a given propositional formula wrt. a given set of propositional formulas all possible worlds are enumerated and tested. Obviously, this reasoner only works for small examples but is useful when one is interested in all models of a knowledge base. The second supported reasoner incorporates the Sat4j reasoner[9]. Other SAT-solvers can be added in a straightforward way.

First-Order Logic This library contains an implementation of first-order logic as a knowledge representation formalism. Both the *Propositional Logic* library and the *First-Order Logic* library are used by many other libraries of knowledge representation formalisms.

[5] https://code.google.com/p/jspf/

[6] http://commons.apache.org/math

[7] http://openopt.org

[8] http://www.emn.fr/z-info/choco-solver/

[9] http://www.sat4j.org

Conditional Logic The *Conditional Logic* library extends the *Propositional Logic* library by *conditionals*, i. e., non-classical rules of the form $(B \mid A)$ ("A usually implies B"), cf. [Nute and Cross, 2002]. In the literature, several different semantics and reasoning approaches for conditional logics have been proposed and this library can be used to easily compare their reasoning behavior. Currently, the *Conditional Logic* library implements interpretations in the form of ranking functions [Spohn, 1988] and conditional structures [Kern-Isberner, 2001], and provides reasoner based on z-ranking [Goldszmidt and Pearl, 1996] and c-representations [Kern-Isberner, 2001].

Relational Conditional Logic Similar to the *Conditional Logic* library the *Relational Conditional Logic* extends the *First-Order Logic* libraries with relational conditionals (i. e. conditionals that may contain first-order formulas), cf. [Delgrande, 1998; Kern-Isberner and Thimm, 2012]. Currently, this library contains an implementation of the relational c-representation reasoning approach of [Kern-Isberner and Thimm, 2012].

Probabilistic Conditional Logic This library further extends the *Conditional Logic* library by extending conditionals to *probabilistic conditionals* of the form $(B \mid A)[p]$ ("A usually implies B with probability p"), cf. [Rödder, 2000]. Besides a naive implementation of probabilistic reasoning based on the principle of maximum entropy [Paris, 1994] this library also contains several classes for analyzing and repairing inconsistent sets of probabilistic conditionals, cf. [Thimm, 2011; 2013b]. We will discuss this package in more detail in Section 4.

Relational Probabilistic Conditional Logic By combining both the *Relational Conditional Logic* and *Probabilistic Conditional Logic* libraries the *Relational Probabilistic Conditional Logic* library introduces relational conditionals with probabilities, cf. [Kern-Isberner and Thimm, 2010]. It implements both the *averaging* and *aggregating* semantics from [Kern-Isberner and Thimm, 2010] and also allows for lifted inference as proposed in [Thimm, 2011].

Markov Logic This library builds on the *First-Order Logic* library to implement *Markov Logic*, an extension of first-order logic with weights to allow for probabilistic reasoning, cf. [Richardson and Domingos, 2006]. It provides several propriety sampling-based reasoner and a bridge to the Alchemy reasoner[10].

Epistemic Logic This library extends the *Propositional Logic* library with modal operators for epistemic logic and its semantics with accessibility relations and Kripke models. Please note that the *Epistemic Logic* library is currently in an experimental phase.

Description Logic The *Description Logic* library provides a general description logic implementation [Baader et al., 2003] based on the *First-Order Logic* library. Please note that the *Description Logic* library is currently in an experimental phase.

[10] http://alchemy.cs.washington.edu

Logic Translators This library provides the abstract class `Translator` that provides basic functionalities to implement translators between different knowledge representation formalisms. Currently, the *Logic Translators* library contains translators between first-order logic and answer set programming, between nested logic programming and answer set programming, and between propositional logic and first-order logic.

Logic Programming Libraries

The *Logic Programming* libraries (located under the package `net.sf.tweety.lp`) provide implementations of knowledge representation formalisms based on logic programming.

Answer Set Programming The *Answer Set Programming* library provides classes for representing extended logic programs [Gelfond and Leone, 2002]. Answer set programs are logic programs of the form $A \leftarrow B_1, \ldots, B_m$ with first-order literals $A, B_1, \ldots, B_m$ where the body literals $B_1, \ldots, B_m$ may also have a default negation not. This library provides bridges to several established solvers such as DLV[11], DLV Complex[12], and Clingo[13].

Dynamics in Answer Set Programming This library extends the *Answer Set Programming* library by introducing revision and update approaches. The library contains implementations of the approaches introduced in [Krümpelmann and Kern-Isberner, 2012; Delgrande, Schaub, and Tompits, 2007] and also revision approaches based on argumentation.

Nested Logic Programming This library contains an implementation of nested logic programs which allow for complex first-order formulas to appear in logic programming rules [Lifschitz, Tang, and Turner, 1999].

Argumentation Libraries

The argumentation libraries (located under the package `net.sf.tweety.arg`) are one of the most mature libraries of Tweety and contain a wide variety of implementations of different approaches to computational argumentation.

Abstract Argumentation This library implements abstract argumentation as proposed in [Dung, 1995]. An abstract argumentation framework is a directed graph (A, Att) where A is interpreted as a set of arguments and an edge $(A, A') \in Att$ is an attack of A on A'. The library provides implementations of the mostly used semantics and their corresponding reasoner, both in terms of extensions (an extension is a set of arguments that is regarded as accepted by a semantics) and labelings (a labeling is a function with a three-valued truth assignment to each argument). Several utility classes for generating random argumentation frameworks complement this library.

[11] http://www.dlvsystem.com

[12] https://www.mat.unical.it/dlv-complex

[13] http://potassco.sourceforge.net

Deductive Argumentation The *Deductive Argumentation* library provides an implementation of the approach proposed in [Besnard and Hunter, 2001]. In deductive argumentation, an argument is composed of a set of propositional formulas that derive the claim of the argument. Attack between arguments is derived from classical unsatisfiability.

Structured Argumentation Frameworks This library implements the approach of structured argumentation frameworks as proposed in [Thimm and Garcia, 2010]. In structured argumentation frameworks arguments are composed of subarguments and a conclusion.

Defeasible Logic Programming This library provides an implementation of Defeasible Logic Programming (DeLP) [Garcia and Simari, 2004]. In DeLP knowledge bases contain strict and defeasible rules and facts, similar to knowledge representation formalisms for logic programming. Defeasible rules can be collected in arguments and compared by generalized specificity [Stolzenburg et al., 2003].

Logic Programming Argumentation This library provides an implementation of the argumentation approach of [Schweimeier and Schroeder, 2003] which is also based on logic programming techniques.

Probabilistic Argumentation The *Probabilistic Argumentation* library extends the *Abstract Argumentation* library with non-classical semantics based on probabilistic assessments [Thimm, 2012].

Agent Libraries

The agent libraries (located under the package `net.sf.tweety.agents`) provide a framework for analyzing and simulating interactions between agents.

Agents This general library contains an abstract formalization of agents and multi-agent systems. Classes such as `Agent`, `Environment`, `MultiAgentSystem`, and `Protocol` can be used to set up and simulate a system of agents within an environment. This library has a specific focus on the simulation aspect and provides classes such as `MultiAgentSystemGenerator` and `GameSimulator` that allow the automatic generation of test scenarios and their evaluation.

Dialogues The library *Dialogues* extends the *Agents* library with the capability of simulating dialogues between agents, as they are investigated in the context of argumentation in multi-agent systems [Karunatillake et al., 2009]. It also provides an implementation of agents with an opponent model as proposed in [Rienstra, Thimm, and Oren, 2013]. We will discuss this package in more detail in Section 4.

Other Libraries

The above discussed libraries constitute the core of Tweety by providing implementations of several knowledge representation formalisms. This collection is complemented by some further libraries that relate either to topics that do not strictly belong to the field of knowledge representation (such as the *Machine Learning* library) or can be applied across several different knowledge representation formalisms (such as the *Belief Dynamics* library).

Action and Change The *Action and Change* library implements several action languages and their dynamics from [Gelfond and Lifschitz, 1998].

Belief Dynamics This library provides a general implementation for various approaches to belief (base) revision and update [Hansson, 2001]. It provides interfaces and several implementations of many concepts used in belief dynamics such as `BaseRevisionOperator`, `BaseContractionOperator`, `IncisionFunction`, and `LeviBaseRevisionOperator`. Those classes are defined in such a general way that they can be used not only to implement belief dynamics for propositional logic but also for other knowledge representation formalisms implementing the corresponding Tweety interfaces. This library contains also specific revision approaches such as selective revision [Fermé and Hansson, 1999] and argumentative selective revision [Krümpelmann et al., 2011].

Machine Learning The *Machine Learning* library provides several abstract concepts that can be used in a machine learning context such as `Observation`, `Classifier`, and `CrossValidator`. It contains also an implementation of support vector machines utilizing LIBSVM[14].

Preferences This library contains classes for representing preference orders and approaches for aggregating them [Walsh, 2007]. It also contains an implementation of the dynamic preference aggregation approach proposed in [Thimm, 2013a].

4 Case Studies and Evaluation

In this section we discuss two case studies that make use of Tweety as a platform for experimentation and empirical evaluation. The first case study is on inconsistency handling for probabilistic logics [Thimm, 2011; 2013b] while the second study is on strategic argumentation in multi-agent systems [Thimm and Garcia, 2010; Rienstra, Thimm, and Oren, 2013].

Inconsistency Handling for Probabilistic Logics

In order to motivate the work described in this section we give a brief introduction into probabilistic conditional logic and its inconsistency measures, cf. [Thimm, 2011; 2013b].

For propositional formulas ϕ, ψ and a real-value $p \in [0, 1]$ we call $(\phi \mid \psi)[p]$ a *probabilistic conditional*. A probabilistic conditional $(\phi \mid \psi)[p]$ represents a specific form of defeasible rule and has the intuitive meaning "if ψ is true then ϕ is true with probability p". A (probabilistic conditional) *knowledge base* $\mathcal{K}$ is a set of probabilistic conditionals. Semantics are given to probabilistic conditionals by probability functions $P : \Omega \rightarrow [0, 1]$ with Ω being the set of interpretations (possible worlds) of the underlying propositional logic (We assume that the set of propositions is finite and so is the

[14] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

set of possible worlds). A probability function P *satisfies* a conditional $(\phi \mid \psi)[p]$ if and only if $P(\phi \wedge \psi) = pP(\psi)$ (the probability of a formula is defined to be the sum of the probabilities of all possible worlds satisfying it). Note that this follows the definition of conditional probability ($P(\phi \mid \psi) = P(\phi \wedge \phi)/P(\psi) = p$) as long as $P(\psi) \neq 0$. In order to avoid a case differentiation for $P(\psi) = 0$ we use the above definition, cf. [Paris, 1994]. A probability function P satisfies a knowledge base $\mathcal{K}$ if and only if it satisfies all its probabilistic conditionals. A knowledge base is consistent if such a probability function exists.

Example 1 *Consider* $\mathcal{K} = \{(f \mid b)[0.9], (b \mid p)[1], (f \mid p)[0.01]\}$ *with the intuitive meaning that birds* (b) *usually (with probability* 0.9*) fly* (f)*, that penguins* (p) *are always birds, and that penguins usually do not fly (only with probability* 0.01*). The knowledge base* $\mathcal{K}$ *is consistent as a probability function satisfying it can easily be constructed, cf. [Thimm, 2013b]. Note that, e. g., the knowledge base* $\mathcal{K} = \{(x \mid y)[0.9], (y \mid \top)[0.9], (x \mid \top)[0.2]\}$ *is inconsistent (*$\top$ *is a logical tautology): considering just the conditionals* $(x \mid y)[0.9]$ *and* $(y \mid \top)[0.9]$ *we obtain that* x *has to be at the least probability* 0.81 *which is inconsistent with stating that* x *has probability* 0.2*.*

In order to deal with inconsistent knowledge bases the work [Thimm, 2013b] proposes inconsistency measures as a tool for analyzing inconsistencies. An inconsistency measure is a function $\mathcal{I}$ that takes a knowledge base $\mathcal{K}$ and computes an inconsistency value $\mathcal{I}(\mathcal{K}) \in [0, \infty)$ with the intuitive meaning that a larger value indicates a more severe inconsistency (and $\mathcal{I}(\mathcal{K}) = 0$ means that $\mathcal{K}$ is consistent). See [Thimm, 2013b] for more details, some rationality postulates on inconsistency measurement, and specific approaches.

The inconsistency measurement framework for probabilistic logics has been implemented in the *Probabilistic Conditional Logic* library of Tweety (sub-package `net.sf.tweety.logics.pcl.analysis`). However, as inconsistency measurement is a broader topic that can also be used in other knowledge representation formalisms such as classical logics [Grant and Hunter, 2006], the concept *inconsistency measure* is already implemented in the *Logic Commons* library (sub-package `net.sf.tweety.logics.commons.analysis`) as a very general interface (only a simplified version is shown):

```
public interface InconsistencyMeasure
            <T extends BeliefBase> {
  public Double inconsistencyMeasure
                        (T beliefBase);
}
```

The interface above is parametrized by the specific type of belief base using Java Generics. All types of belief bases used within Tweety, such as propositional belief sets (`PlBeliefSet`) or probabilistic conditional knowledge bases (`PclBeliefSet`), are derived from `BeliefBase`. The package `net.sf.tweety.logics.commons.analysis` provides several generally applicable implementations of the above interface such as (only a simplified version is shown):

```
public class MiInconsistencyMeasure
  <S extends Formula,T extends BeliefSet<S>>
  implements InconsistencyMeasure<T> {

  private BeliefSetConsistencyTester<S,T>
      consTester;

  public MiInconsistencyMeasure(
      BeliefSetConsistencyTester<S,T>
      consTester){
    this.consTester = consTester;
  }

  @Override
  public Double inconsistencyMeasure
    (T beliefSet) {
      return new Double(this.consTester.
          minimalInconsistentSubsets(
          beliefSet).size());
  }
}
```

The above measure is an implementation of the MI-inconsistency measure [Grant and Hunter, 2006] and is applicable for all kinds of logics that provide an implementation of an `BeliefSetConsistencyTester`. This measure takes the number of minimal inconsistent subsets of a knowledge as an assessment of its inconsistency. Another example, particularly for the case of probabilistic conditional logic, is the `DistanceMinimizationInconsistencyMeasure` that makes use of the *Math* library. This measure assesses the grade of inconsistency by measuring how much the probabilities of the conditionals have to be modified in order to obtain a consistent knowledge base, cf. [Thimm, 2013b]. This problem is solved by optimization techniques that can be found in the *Math* library.

For probabilistic conditional logic, determining whether a knowledge base is inconsistent and assessing its inconsistency value is not easily done by hand. Using the implementations of various inconsistency measures in Tweety, we were able to compute and compare inconsistency values for various knowledge bases, cf. [Thimm, 2011]. Furthermore, as the interfaces and abstract classes provided by Tweety are very general and force the programmer to work as abstract as possible, even the implementation of such specific concepts such as inconsistency measures yield very generally applicable classes that are easily adapted to other approaches.

Strategic Argumentation

Our second case study is about strategic argumentation in multi-agent systems. In the works [Thimm and Garcia, 2010; Rienstra, Thimm, and Oren, 2013] we investigated systems of agents that are engaged in dialogues and aim at resolving contradiction by exchange of arguments. We give a brief introduction into the topic now, but simplify the formalization for the sake of readability.

We consider two agents PRO (proponent) and OPP (opponent) engaged in a dialogue about a specific argument A

(we use the terminology of abstract argumentation frameworks as mentioned earlier). The proponent has the goal to establish that A is acceptable and the opponent has the goal to establish that A is not acceptable. Both agents have only access to a subset of all available arguments and are, in general, ignorant or uncertain about the arguments the other agent has access to. Both agents take turn in forwarding a set of arguments. In [Rienstra, Thimm, and Oren, 2013] several different belief states with *opponent models* were proposed and discussed that help an agent to act strategically in these kinds of dialogues. The first type T_1 of belief state is a tuple (B, E) where B is the set of arguments a particular agent (either PRO or OPP) has access to, and E is the opponent model which is itself a belief state of type T_1[15]. This type of belief state therefore models what an agent thinks another agent beliefs, etc.. The second T_2 and third T_3 types of belief state extend the first type by introducing uncertainty on the set of arguments believed by the other agent and uncertainty about the arguments themselves. The second type of belief state T_2 is a tuple (B, P) where B is again the set of arguments a particular agent has access to and P is a probability distribution over some set $\{K_1, \ldots, K_n\}$ where each K_i $(i = 1, \ldots, n)$ is again a belief state of type T_2. For a formalization of the belief state of type T_3 see [Rienstra, Thimm, and Oren, 2013]. In [Rienstra, Thimm, and Oren, 2013] it has been analytically shown that the expressiveness of the three models is increasing from T_1 to T_3. However, in order to understand the differences between the three models examples have to be created and computed with different belief states. In the setting of strategic argumentation, this is a hard task to do by hand. In a system with at least two agents where both agents are equipped with a non-trivial belief state that changes with every action, running through a complete example by hand is a tedious task.

The complete setting of [Rienstra, Thimm, and Oren, 2013] has been implemented in the *Dialogues* library which makes heavy use of the general agent classes from the *Agents* library and, of course, the knowledge representation formalism from the *Abstract Argumentation* library. The central class of the implementation is the `ArguingAgent` class (we only show an excerpt):

```
public class ArguingAgent extends Agent {
  private BeliefState beliefState;
  private AgentFaction faction;

  @Override
  public Executable next(Collection<?
      extends Perceivable> percepts) {
    // [env = the environment object]
    this.beliefState.update(env.
        getDialogueTrace());
    return this.beliefState.move(env);
  }
}
```

The central attributes of an arguing agent are its belief state and its faction, e. g., either PRO or OPP. The method `next(...)` (derived from the super-class `Agent`) determines the agent's behavior on receiving some perception from the environment and returns some action (of type `Executable`). Here, the agent first updates its belief state with the current dialogue trace (a sequence of sets of arguments advanced so far) and then returns its own move (a set of arguments). The three different belief state types have been implemented in the classes `T1BeliefState`, `T2BeliefState`, and `T3BeliefState`. Arguing agents are organized in a `GroundedGameSystem` which is of type `MultiAgentSystem<ArguingAgent>` and models an argumentation dialogue ("grounded" refers to the grounded semantics used for this type of game). On top of this implementation of the actual dialogue system a simulation framework was implemented that allows the (random) generation of the above multi-agent systems and measures the performance of the individual agents over a series of runs. The central class for the simulation framework is the `GroundedGameGenerator` which implements the interface `MultiAgentSystemGenerator` and is able to generate (random) multi-agent systems of arguing agents.

In [Rienstra, Thimm, and Oren, 2013], for evaluating performance we generated a random abstract argumentation theory with 10 arguments, ensuring that the argument under consideration is in its grounded extension, i. e., under perfect information the proponent should win the dialogue. However, from these 10 arguments only 50 % are known by the proponent but 90 % by the opponent. We used a proponent without opponent model and generated an belief state of type T_3 for the opponent. From this T_3 belief state we derived T_2 and T_1 belief states by ignoring the added expressivity. For each belief state we simulated a dialogue against the same opponent and counted the number of wins. We repeated the experiment 5000 times, Figure 1 shows our results, cf. [Rienstra, Thimm, and Oren, 2013]. There, it can be seen that increasing the complexity of the belief state yields better overall performance (thus confirming the analytical evaluation). However, this empirical evaluation sheds also more light on the importance of the added expressivity for strategic argumentation. While T_2 is significantly better that T_1, the difference between T_3 and T_2 is nearly marginal. These kinds of nuances are very difficult to discover when considering only analytical evaluation.

5 Summary and Future Work

In this paper we presented Tweety, a comprehensive collection of Java libraries for logical aspects of artificial intelligence and knowledge representation. We gave an overview on the technical aspects and provided details on its individual packages. Finally, we presented two case studies that make use of Tweety as a framework for experimentation and empirical evaluation.

Tweety is an open source project[16,17] and can therefore be used and extended by everyone. In particular, instantiating the abstract Tweety classes for a particular formalism is sim-

[15] Note that this model has originally been proposed in [Oren and Norman, 2010]

[16] http://www.mthimm.de/projects/tweety/

[17] The source code of Tweety is hosted at SourceForge: http://tweety.svn.sourceforge.net.

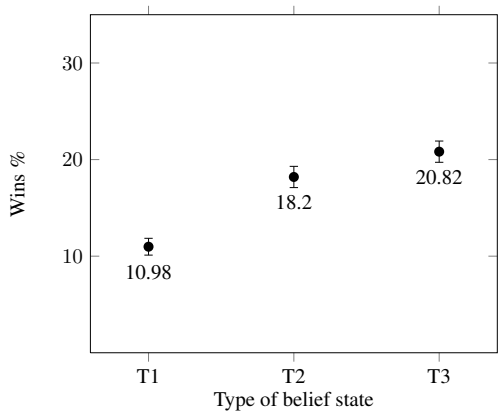


Figure 1: Average performance of T_1, T_2, and T_3 belief state models after 5000 simulation runs (with Binomial proportion confidence intervals)

ple. Although Tweety is implemented in a object-oriented programming language it follows a strict declarative formal way to define concepts from theoretical knowledge representation research. Tweety is available under the GNU General Public License version 3.0. In order to contribute to the main Tweety repository contact the author.

To the best of our knowledge, Tweety is the first attempt to provide a general-purpose framework for a broad variety of knowledge representation formalisms. However, there exist also more specialized frameworks for specific approaches or areas, such as the OWLAPI[18] for working with OWL ontologies, KReator[19] for relational probabilistic knowledge representation, or bcontractor[20] for belief dynamics.

Current and future work on Tweety is mainly concerned with extending the general infrastructure and improving usability. In particular, current work is about implementation of the plugin architecture for all libraries, a command line interface, and a web front-end. The ultimate goal there is to have several standardized user interfaces that are apt to work with any kind of knowledge representation mechanism and thus remove the burden of designing and implementing user interfaces from the researcher.

Acknowledgments

Tweety is being collaboratively developed by several contributors. Thanks go to Sebastian Homann, Tim Janus, Patrick Krümpelmann, Nico Potyka, Tjitze Rienstra, Stefan Tittel, Thomas Vengels, and Bastian Wolf. The research reported here was partially supported by the FP7 project SocialSensor (EC contract number 287975).

References

Antoniou, G., and van Harmelen, F. 2004. *A Semantic Web Primer*. Cambridge, MA, USA: MIT Press.

[18]http://owlapi.sourceforge.net

[19]http://kreator-ide.sourceforge.net

[20]https://code.google.com/p/bcontractor/

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press.

Baroni, P.; Caminada, M.; and Giacomin, M. 2011. An introduction to argumentation semantics. *The Knowledge Engineering Review* 26(4):365–410.

Besnard, P., and Hunter, A. 2001. A logic-based theory of deductive arguments. *Artificial Intelligence* 128(1-2):203–235.

Besnard, P., and Hunter, A. 2006. Knowledgebase compilation for efficient logical argumentation. In *Proceedings of the 10th International Conference on Knowledge Representation (KR'06)*, 123–133. AAAI Press.

Brachman, R. J., and Levesque, H. J. 2004. *Knowledge Representation and Reasoning*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann Publishers.

Delgrande, J. P.; Schaub, T.; and Tompits, H. 2007. A preference-based framework for updating logic programs. In *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*.

Delgrande, J. P. 1998. On first-order conditional logics. *Artificial Intelligence* 105(1-2):105–137.

Dung, P. M. 1995. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence* 77(2):321–358.

Fagin, R.; Halpern, J.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning about Knowledge*. MIT Press.

Fermé, E., and Hansson, S. O. 1999. Selective revision. *Studia Logica* 63(3):331–342.

Garcia, A., and Simari, G. R. 2004. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming* 4(1–2):95–138.

Gelfond, M., and Leone, N. 2002. Logic programming and knowledge representation - the a-prolog perspective. *Artificial Intelligence* 138(1-2):3–38.

Gelfond, M., and Lifschitz, V. 1998. Action languages. *Electronic Transactions on AI* 2:193–210.

Goldszmidt, M., and Pearl, J. 1996. Qualitative probabilities for default reasoning, belief revision, and causal modeling. *Artificial Intelligence* 84:57–112.

Grant, J., and Hunter, A. 2006. Measuring inconsistency in knowledgebases. *Journal of Intelligent Information Systems* 27:159–184.

Hansson, S. O. 2001. *A Textbook of Belief Dynamics*. Norwell, MA, USA: Kluwer Academic Publishers.

Hunter, A., and Konieczny, S. 2006. Shapley inconsistency values. In *Proceedings of the 10th International Conference on Knowledge Representation (KR'06)*, 249–259. AAAI Press.

Karunatillake, N. C.; Jennings, N. R.; Rahwan, I.; and McBurney, P. 2009. Dialogue games that agents play within a society. *Artificial Intelligence* 173(9-10):935–981.

Kern-Isberner, G., and Thimm, M. 2010. Novel semantical approaches to relational probabilistic conditionals. In Lin, F.; Sattler, U.; and Truszczyński, M., eds., *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR'10)*, 382–392. AAAI Press.

Kern-Isberner, G., and Thimm, M. 2012. A ranking semantics for first-order conditionals. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12)*.

Kern-Isberner, G. 2001. *Conditionals in Nonmonotonic Reasoning and Belief Revision*. Number 2087 in Lecture Notes in Computer Science. Springer-Verlag.

Krümpelmann, P., and Kern-Isberner, G. 2012. Belief base change operations for answer set programming. In Cerro, L. F.; Herzig, A.; and Mengin, J., eds., *Proceedings of the 13th European conference on Logics in Artificial Intelligence (JELIA'12)*, 294–306. Springer.

Krümpelmann, P.; Thimm, M.; Falappa, M. A.; Garcia, A. J.; Kern-Isberner, G.; and Simari, G. R. 2011. Selective revision by deductive argumentation. In Modgil, S.; Oren, N.; and Toni, F., eds., *Theory and Applications of Formal Argumentation, Proceedings of the First International Workshop (TAFA'11, revised selected papers)*, volume 7132 of *Lecture Notes in Artificial Intelligence*, 147–162. Springer.

Lifschitz, V.; Tang, L. R.; and Turner, H. 1999. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence* 25:369–389.

McCabe, T. J. 1976. A complexity measure. *IEEE Transactions on Software Engineering* 308–320.

Nute, D., and Cross, C. 2002. Conditional logic. In Gabbay, D., and Guenther, F., eds., *Handbook of Philosophical Logic*, volume 4. Kluwer Academic Publishers, second edition edition. 1–98.

Oren, N., and Norman, T. J. 2010. Arguing using opponent models. In *Proceedings of the 6th international conference on Argumentation in Multi-Agent Systems*, ArgMAS'09, 160–174. Berlin, Heidelberg: Springer-Verlag.

Paris, J. B. 1994. *The Uncertain Reasoner's Companion – A Mathematical Perspective*. Cambridge University Press.

Pearl, J. 1990. System Z: a natural ordering of defaults with tractable applications to default reasoning. In *Proceedings of the 3rd conference on Theoretical aspects of reasoning about knowledge (TARK'90)*, 121–135. Morgan Kaufmann Publishers.

Potyka, N. 2014. Linear programs for measuring inconsistency in probabilistic logics. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14)*.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1–2):107–136.

Rienstra, T.; Thimm, M.; and Oren, N. 2013. Opponent models with uncertainty for strategic argumentation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*.

Rödder, W. 2000. Conditional logic and the principle of entropy. *Artificial Intelligence* 117:83–106.

Schweimeier, R., and Schroeder, M. 2003. A parameterised hierarchy of argumentation semantics for extended logic programming and its application to the well-founded semantics. *Theory and Practice of Logic Programming* 5(1–2):207–242.

Spohn, W. 1988. Ordinal conditional functions: a dynamic theory of epistemic states. In Harper, W., and Skyrms, B., eds., *Causation in Decision, Belief Change, and Statistics*, volume 2. Kluwer Academic Publishers. 105–134.

Stolzenburg, F.; Garcia, A.; Chesnevar, C. I.; and Simari, G. R. 2003. Computing generalized specificity. *Journal of Non-Classical Logics* 13(1):87–113.

Thimm, M., and Garcia, A. J. 2010. Classification and Strategical Issues of Argumentation Games on Structured Argumentation Frameworks. In van der Hoek, W.; Kaminka, G. A.; Lespérance, Y.; Luck, M.; and Sen, S., eds., *Proceedings of the Ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems 2010 (AAMAS'10)*.

Thimm, M., and Kern-Isberner, G. 2008. A distributed argumentation framework using defeasible logic programming. In Besnard, P.; Doutre, S.; and Hunter, A., eds., *Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA'08)*, number 172 in Frontiers in Artificial Intelligence and Applications, 381–392. IOS Press.

Thimm, M. 2011. *Probabilistic Reasoning with Incomplete and Inconsistent Beliefs*, volume 331 of *Dissertations in Artificial Intelligence*. IOS Press.

Thimm, M. 2012. A probabilistic semantics for abstract argumentation. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12)*.

Thimm, M. 2013a. Dynamic preference aggregation under preference changes. In *Proceedings of the Fourth Workshop on Dynamics of Knowledge and Belief (DKB'13)*.

Thimm, M. 2013b. Inconsistency measures for probabilistic logics. *Artificial Intelligence* 197:1–24.

Walsh, T. 2007. Representing and reasoning with preferences. *AI Magazine* 28(4):59–70.

Weiss, G., ed. 2013. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, MA, USA: MIT Press, second edition.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Reasoning with Uncertain Inputs in Possibilistic Networks

Salem Benferhat , Karim Tabia
Univ Lille Nord de France, F-59000 Lille, France.
UArtois, CRIL UMR CNRS 8188, F-62300 Lens, France.
{benferhat, tabia}@cril.univ-artois.fr

Abstract

Graphical belief models are compact and powerful tools for representing and reasoning under uncertainty. Possibilistic networks are graphical belief models based on possibility theory. In this paper, we address reasoning under uncertain inputs in both quantitative and qualitative possibilistic networks. More precisely, we first provide possibilistic counterparts of Pearl's methods of virtual evidence then compare them with the possibilistic counterparts of Jeffrey's rule of conditioning. As in the probabilistic setting, the two methods are shown to be equivalent in the quantitative setting regarding the existence and uniqueness of the solution. However in the qualitative setting, Pearl's method of virtual evidence which applies directly on graphical models disagrees with Jeffrey's rule and the virtual evidence method. The paper provides the precise situations where the methods are not equivalent. Finally, the paper addresses related issues like transformations from one method to another and commutativity.

Introduction

Belief revision and more generally belief dynamics is a fundamental task in artificial intelligence. Indeed, rational agents often need to revise their beliefs in order to take into account new information. In uncertainty frameworks, this task is often referred to as belief revision or reasoning with uncertain inputs. Belief revision has received a lot of attention in artificial intelligence especially in logic-based and some uncertainty frameworks (Benferhat et al. 2010)(Dubois and Prade 1994). In spite of the power of graphical belief models for representing and reasoning with uncertain information, belief revision and reasoning with uncertain inputs in such models is addressed only in few works mostly in the context of Bayesian networks (Chan and Darwiche 2005)(Vomlel 2004).

In this paper, we compare two methods for revising the beliefs encoded in a possibilistic framework when new and uncertain information is available. The two methods compared here are Jeffrey's rule of conditioning (Jeffrey 1965) and the virtual evidence method (Pearl 1988). They were originally proposed and studied in a probabilistic setting where they are shown to be equivalent and differ only in the way they specify the inputs (Chan and Darwiche 2005).

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In the possibilistic setting, the counterparts of Jeffrey's rule are proposed in (Dubois and Prade 1997)(Dubois and Prade 1993). In (Benferhat, Tabia, and Sedki 2011), we studied the existence and the uniqueness of the solution in both the quantitative and qualitative possibilistic settings. The possibilistic counterpart of Jeffrey's rule is investigated for belief revision in possibilistic knowledge bases in (Benferhat et al. 2010) where it is claimed that this rule can successfully recover most of the belief revision kinds such as the natural belief revision, drastic belief revision, reinforcement, etc. In (Benferhat, Da Costa Pereira, and Tettamanzi 2013), a syntactic version is proposed for the possibilistic counterpart of Jeffrey's rule. In this paper, we address revising the beliefs encoded by means of possibilistic networks with uncertain inputs. More precisely, the paper provides

- Possibilistic counterparts of Pearl's method of virtual evidence and its generalization named the virtual evidence method in both the quantitative and qualitative settings. Unlike the probabilistic and quantitative possibilistic settings, the inputs for the qualitative counterparts of Pearl's methods should be possibility degrees because of the definition of the qualitative conditioning.
- An analysis of the existence and uniqueness of the solutions using the proposed possibilistic counterparts of Pearl's methods.
- Transformations from Jeffrey's rule to the virtual evidence method and vice versa and comparisons of these methods in both the quantitative and qualitative settings. As in the probabilistic setting, the two methods are shown to be equivalent in the quantitative setting regarding the existence and uniqueness of the solution. However in the qualitative setting, Pearl's method of virtual evidence is not equivalent to Jeffrey's rule since it is impossible using this method to increase the possibility degree of an event but its generalization is shown equivalent to Jeffrey's rule.

Possibility Theory and Possibilistic networks

Let us first fix the notations used in the rest of this paper. V=$\{X, Y, A_1, A_2, ..\}$ denotes a set of variables (in capital letters and indexed when necessary). D_{A_i}=$\{a_1, a_2, .., a_m\}$ denotes the domain of a variable A_i (note that D_{A_i} is assumed a finite domain). a_i denotes an instance (value) of variable A_i, namely $a_i \in D_{A_i}$. Ω=$\times_{A_i \in V} D_{A_i}$ denotes the

universe of discourse (all possible states of the world). It is the cartesian product of all the variable domains involved in V. A tuple w=$(a_1, a_2, .., a_n)$ which is an instance of Ω represents a possible state of the world (also called a model or interpretation). ϕ, φ, λ denote subsets of Ω called events.

Possibility theory

Possibility theory is an alternative uncertainty theory suited for representing and reasoning with uncertain and incomplete information (Dubois and Prade 1988a; Yager 1983). The concept of possibility distribution π is an important building block of possibility theory: It is a mapping from the universe of discourse Ω to the unit scale $[0, 1]$ which can be either quantitative or qualitative (ordinal). In both these settings, a possibility degree $\pi(w_i)$ expresses to what extent it is consistent that w_i can be the actual state of the world. In particular, $\pi(w_i)$=1 means that w_i is totally plausible and $\pi(w_i)$=0 denotes an impossible event. The relation $\pi(w_i)>\pi(w_j)$ means that w_i is more plausible than w_j. A possibility distribution π is said to be normalized if $\max_{w_i \in \Omega}(\pi(w_i))$=1. The second important concept in possibility theory is the one of possibility measure denoted $\Pi(\phi)$ and computing the possibility degree relative to an event $\phi \subseteq \Omega$. It evaluates to what extent ϕ is consistent with the current knowledge encoded by the possibility distribution π on Ω. It is defined as follows:

$$\Pi(\phi) = \max_{w_i \in \phi}(\pi(w_i)). \quad (1)$$

The term $\Pi(\phi)$ denotes the possibility degree of having one of the events involved in ϕ as the actual state of the world.
The necessity measure is the dual of possibility measure and evaluates the certainty implied by the current knowledge of the world. Namely, $N(\phi)$=$1 - \Pi(\overline{\phi})$ where $\overline{\phi}$ denotes the complement of ϕ.
According to the interpretation underlying the possibilistic scale [0,1], there are two variants of possibility theory:

- **Qualitative (or min-based) possibility theory**: In this case, the possibility distribution is a mapping from the universe of discourse Ω to an *ordinal* scale where only the "ordering" of the values is important.
- **Quantitative (or product-based) possibility theory**: In this case, the possibilistic scale [0,1] is numerical and possibility degrees are like numeric values that can be manipulated by arithmetic operators. One of the possible interpretations of quantitative possibility distributions is viewing $\pi(w_i)$ as a degree of surprise as in Spohn's ordinal conditional functions (Spohn 1988).

The other fundamental notion in possibility theory is the one of conditioning concerned with updating the current knowledge encoded by the possibility distribution π when a completely sure event (evidence) is observed. Note that there are several definitions of the possibilistic conditioning (Hisdal 1978)(L.M. De Campos and Moral 1995)(Dubois and Prade 1988b) (Fonck 1997). In the quantitative setting, the product-based conditioning (also known as Dempster rule of conditioning (Shafer 1976)) is defined as follows:

$$\pi(w_i|_p\phi) = \begin{cases} \frac{\pi(w_i)}{\Pi(\phi)} & \text{if } w_i \in \phi; \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Conditioning in the qualitative setting is defined as follows (Hisdal 1978):

$$\pi(w_i|_m\phi) = \begin{cases} 1 & \text{if } \pi(w_i)=\Pi(\phi) \text{ and } w_i \in \phi; \\ \pi(w_i) & \text{if } \pi(w_i)< \Pi(\phi) \text{ and } w_i \in \phi; \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

While there are several similarities between the quantitative possibilistic and the probabilistic frameworks (conditioning is defined in the same way), the qualitative one is significantly different. Note that the two definitions of conditioning satisfy the condition: $\forall \omega \in \phi$, $\pi(\omega)=\pi(\omega|\phi) \otimes \Pi(\phi)$ where $\otimes$ is either the product or min-based operator.

Possibilistic networks

A possibilistic network G=$<\mathcal{G},\Theta>$ is specified by:

i) *A graphical component* $\mathcal{G}$ consisting in a directed acyclic graph (DAG) where vertices represent variables of interest and edges represent direct *dependence* relationships between these variables.

ii) *A quantitative component* Θ allowing to quantify the uncertainty of the relationships between domain variables using local possibility tables (CPTs). The quantitative component or G's parameters consist in a set of local possibility tables Θ_i=$\{\theta_{a_i|u_i}\}$ where $a_i \in D_i$ and u_i is an instance of U_i denoting the parent variables of A_i in $\mathcal{G}$.

Note that all the local possibility distributions Θ_i must be normalized, namely $\forall i$=1..n, $\forall u_i \in D_{U_i}$, $\max_{a_i \in D_i}(\theta_{a_i|u_i})$=1. The structure of G encodes a set of conditional independence relationships I=$\{I(A_i, U_i, Y)\}$ where Y is a subset of variables non descendent from A_i. For example, in the network of Figure 1, variable C is independent of B in the context of A.

Example 1. Figure 1 gives an example of a possibilistic network over four binary variables A, B, C and D.

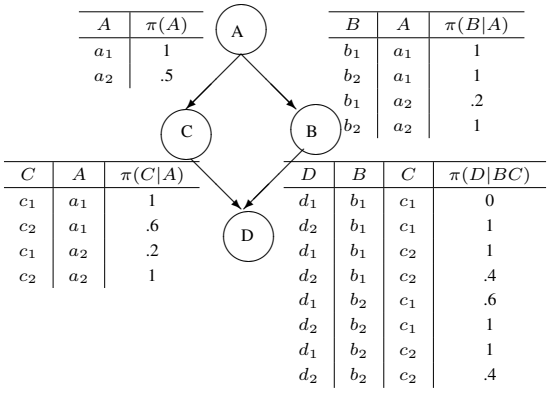


Figure 1: Example of a possibilistic network

In the possibilistic setting, the joint possibility distribution is factorized using the chain rule defined as follows:

$$\pi(a_1, a_2, .., a_n) = \otimes_{i=1}^{n}(\pi(a_i|u_i)), \quad (4)$$

where $\otimes$ denotes the product-based (resp. min-based) operator used in the quantitative (resp. qualitative) setting.

Example 2. In the min-based setting, the joint distribution encoded by the network of Figure 1 is derived as follows:

$$\pi(A, B, C, D) = \min(\pi(A), \pi(C|A), \pi(B|A), \pi(D|BC)).$$

Reasoning with uncertain inputs in the probabilistic setting

In the probabilistic framework, there are two main methods for revising beliefs represented using probability distributions or probabilistic models by uncertain information: Jeffrey's rule (Jeffrey 1965) and the virtual evidence methods (Pearl 1988). Let us first focus on the notions of beliefs and uncertain inputs.

Beliefs and uncertain inputs

The concept of belief used in this paper allows an agent to encode at which extent a given event is believed to be or become the actual state of the world. Generally, beliefs are specified over a universe of discourse Ω using belief measures like probability or possibility measures[1]. Then belief degrees are associated with each singleton event $\omega \in \Omega$ in the form of a belief distribution. According to the chosen setting, belief measures allow to assess the belief of any arbitrary event $\phi \subseteq \Omega$. Now given a set of initial beliefs (also called prior beliefs), an agent may have new information which can be in the form of evidence (also called hard evidence and corresponding for instance to a sure observation of the value of a variable) or in the form of uncertain or soft evidence (e.g. unreliable input) or simply new beliefs regarding some events[2]. In the uncertainty literature, belief change dealing with hard evidence is known as belief update and it is generally based on conditioning while it is known as belief revision in case of uncertain inputs.
In Jeffrey's rule and the virtual evidence methods, the uncertainty bears on an exhaustive and mutually exclusive set of events $\lambda_1,..,\lambda_n$ (namely, $\forall \lambda_i \subseteq \Omega$ and $\forall \lambda_j \subseteq \Omega$ with $i \neq j$, we have $\lambda_i \cap \lambda_j = \emptyset$ and $\lambda_1 \cup \lambda_2 \cup .. \cup \lambda_n = \Omega$). However, the new information is expressed differently:

- In Jeffrey's rule, the new beliefs are a probability distribution over $\lambda_1,..,\lambda_n$ and must consequently sum up to 1. The new information is expressed in the form of (λ_i, α_i) such that $P'(\lambda_i) = \alpha_i$ where p' denotes the revised probability distribution fully accepting the new beliefs.
- In Pearl's methods, the new information is expressed by specifying the amount of increase or decrease of the belief on each event λ_i moving from p to p'. This amount is called in (Darwiche 2009) the Bayes factor and corresponds to the ratio $\frac{P'(\lambda_i)}{P(\lambda_i)}$. For example, a ratio regarding an event λ_i of 2 means that the new belief regarding λ_i is twice as it was before receiving this new information.

[1]The beliefs of an agent can be encoded using other formalisms like belief bases (e.g. probabilistic or possibilistic knowledge bases), graphical belief models, etc.

[2]On the different meanings of hard, soft and uncertain evidence, see (Ma and Liu 2011)(Pan, Peng, and Ding 2006)(Bilmes 2004).

Jeffrey's rule of conditioning

Jeffrey's rule (Jeffrey 1965) is an extension of the probabilistic conditioning to the case where the evidence is uncertain. This method involves a way for:

1. **Specifying the uncertain evidence:** The uncertainty is of the form (λ_i, α_i) with $\alpha_i = P'(\lambda_i)$ meaning that after the revision operation, the posterior probability of each event λ_i must be equal to α_i (namely, $P'(\lambda_i) = \alpha_i$). The uncertain inputs are seen as a constraint or an effect once the new information is fully accepted.
2. **Computing the revised probability distributions:** Jeffrey's method assumes that although there is a disagreement about the events λ_i in the old distribution p and the new one p', the conditional probability of any event $\phi \subseteq \Omega$ given any uncertain event λ_i remains the same in the original and the revised distributions. Namely,

$$\forall \lambda_i \in \Omega, \forall \phi \subseteq \Omega, P(\phi|\lambda_i) = P'(\phi|\lambda_i). \quad (5)$$

The underlying interpretation of the revision implied by constraint of Equation 5 is that the revised probability distribution p' must not change the conditional probability degrees of any event ϕ given the uncertain events λ_i. To revise the probability degree of any event $\phi \subseteq \Omega$, the following formula is used:

$$P'(\phi) = \sum_{\lambda_i} \alpha_i * \frac{P(\phi, \lambda_i)}{P(\lambda_i)}. \quad (6)$$

The revised distribution p' obtained using Jeffrey's rule always exists and it is unique (Chan and Darwiche 2005). In the following, we first present Pearl's method of virtual evidence applying directly on Bayesian networks then its generalization named virtual evidence method applying directly on probability distributions as in Jeffrey's rule.

Pearl's method of virtual evidence

This method is proposed in (Pearl 1988) in the framework of Bayesian networks. The main idea of this method is to cast the uncertainty relative to the uncertain evidence E on some $virtual$ sure event η: the uncertainty regarding E is specified as the likelihood of η in the context of E. In Pearl's method of virtual evidence the beliefs are encoded with a Bayesian network over a set of variables $\{A_1, .., A_n\}$. Assume that the observation regarding a variable A_i is uncertain (for instance, because of a sensor unreliability). Pearl's virtual evidence method deals with this issue by adding for each uncertain observation variable A_i a variable Z_i with an arc from A_i to Z_i. The uncertainty relative to A_i is then cast as the likelihoods of $Z_i = z_i$ in the context of A_i. Then the uncertain inputs are taken into account by observing the sure evidence $Z_i = z_i$. Doing this way, it is clear that the conditional probability of any event ϕ given A_i is the same in the old and revised distribution, namely $\forall \phi \subseteq \Omega$, $p(\phi|A_i) = p'(\phi|A_i)$. It is the d-separation[3] criterion that ensures this property. In this method, the uncertainty bears on a set of exhaustive and mutually exclusive events $a_1,..,a_n$ (forming the domain of variable A_i). Let $\gamma_1 :..: \gamma_n$ denote the

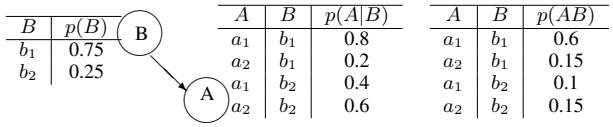
[3]The d-separation property states that two disjoint variable subsets X and Y are d-separated if there exists a third variable sub-set Z such that X and Y are independent given Z.

likelihood ratios encoding the new inputs. Such ratios should satisfy the following condition:

$$\gamma_1 : .. : \gamma_n = \frac{P'(a_1)}{P(a_1)} : .. : \frac{P'(a_n)}{P(a_n)} \quad (7)$$

Note that there are many solutions for the values of $\gamma_1, .. , \gamma_n$ satisfying the condition of Equation 7 (one possible solution for encoding the inputs within the network is to set $p(z|a_i)$ to γ_i=$\frac{p'(a_i)}{p(a_i)}$). It is worth to mention that contrary to Jeffrey's rule where the inputs α_1,..,α_n are the revised belief degrees once the revision performed, in Pearl's methods, the inputs are likelihood ratios γ_1,..,γ_n satisfying Equation 7 and they don't form a probability distribution.

Example 3. *Assume that the current beliefs about a given problem are encoded by the Bayesian network G of Figure 2 over two binary variables A and B. The joint probability distribution encoded by this network is given by the joint probability distribution $p(AB)$ of Figure 2.*



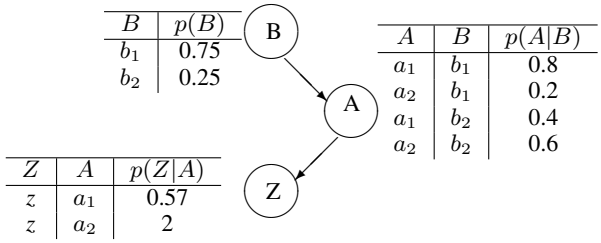
B	$p(B)$
b_1	0.75
b_2	0.25

A	B	$p(A\|B)$
a_1	b_1	0.8
a_2	b_1	0.2
a_1	b_2	0.4
a_2	b_2	0.6

A	B	$p(AB)$
a_1	b_1	0.6
a_2	b_1	0.15
a_1	b_2	0.1
a_2	b_2	0.15

Figure 2: Example of an initial Bayesian network G and the joint distribution $p(AB)$ encoded by G.

Assume now that we have new inputs γ_{a_1}=.57 and γ_{a_2}= 2. Following Pearl's method of virtual evidence, this is handled by adding a variable Z as a child of A as in Figure 3.



B	$p(B)$
b_1	0.75
b_2	0.25

A	B	$p(A\|B)$
a_1	b_1	0.8
a_2	b_1	0.2
a_1	b_2	0.4
a_2	b_2	0.6

Z	A	$p(Z\|A)$
z	a_1	0.57
z	a_2	2

Figure 3: Bayesian network G': Bayesian network G of Figure 2 augmented with node Z to encode the new inputs.

Let us mention that the conditional probability table of node Z in order to encode the new inputs don't need to be normalized (it can easily be normalized but this is not needed to revise the old beliefs encoded by the initial network). Another solution satisfying Equation 7 is γ_{a_1}:γ_{a_2} = .2:.7 (since $\frac{.57}{2}$=$\frac{.2}{.7}$). The revised beliefs are given in Table 1.

A	B	$p(AB\|z)$
a_1	b_1	0.34
a_2	b_1	0.3
a_1	b_2	0.06
a_2	b_2	0.3

Table 1: The conditional distribution $p_{G'}(.|z)$ representing the revised distribution encoded by the network of Figure 2.

It is easy to check that the revised distribution p'=$p_{G'}(.|z)$ fully integrates the inputs.

Virtual evidence method

The virtual evidence method generalizes Pearl's method of virtual evidence and applies directly on joint probability distributions as in Jeffrey's rule.

1. **Specifying the uncertain inputs:** The new information is in the form of a set of likelihood ratios γ_1,..,γ_n such that γ_i=$P(\eta|\lambda_i)$ and
$$\gamma_1 : .. : \gamma_n = \frac{P'(\lambda_1)}{P(\lambda_1)} : .. : \frac{P'(\lambda_n)}{P(\lambda_n)},$$
where λ_1,..,λ_n denote the exhaustive and mutually exclusive set of events on which bears the uncertainty. Moreover, as a consequence of the d-separation criterion in Bayesian networks, we have the following property:
$$\forall \phi \subseteq \Omega, \forall i = 1..n, P'(\eta|\lambda_i, \phi) = P'(\eta|\lambda_i),$$
where η denotes the virtual event.
2. **Computing the revised beliefs:** The revised probability distribution p' is simply equivalent to $p(.|\eta)$ and it is computed as follows (Chan and Darwiche 2005):
$$\forall \phi \subseteq \Omega, P'(\phi) = P(\phi|\eta) = \frac{\sum_{i=1}^{n}(\gamma_i * P(\lambda_i, \phi))}{\sum_{j=1}^{n}(\gamma_j * P(\lambda_j))}. \quad (8)$$

Example 4. *Let us reuse the joint probability distribution of the example of Figure 2. Let also the likelihood ratios be γ_1=$\frac{P'(a_1)}{P(a_1)}$=.57 and γ_2=$\frac{P'(a_2)}{P(a_2)}$=2. The revised distribution p' is computed using Equation 8.*

A	B	$p(AB)$
a_1	b_1	0.6
a_2	b_1	0.15
a_1	b_2	0.1
a_2	b_2	0.15

A	B	$p(AB\|\eta)$
a_1	b_1	0.34
a_2	b_1	0.3
a_1	b_2	0.06
a_2	b_2	0.3

Table 2: Example of initial probability distribution p and the revised distribution $p(.|\eta)$.

From the results of Table 1 and Table 2, it is clear that the revised distributions are equivalent.

Jeffrey's rule and Pearl's methods differ only in the way they specify the inputs and the way the revised beliefs are computed (Chan and Darwiche 2005). In Jeffrey's rule, the inputs are seen as the result or the effect of the revision operation while in the virtual evidence method, the inputs only denote the relative difference between the old beliefs and the revised ones specified in terms of likelihood ratios.
In the following, we compare the two methods presented in this section in a possibilistic framework.

Reasoning with uncertain inputs in the quantitative possibilistic setting

Jeffrey's rule of conditioning in the quantitative possibilistic setting

In the possibilistic setting, given the initial beliefs encoded by a possibility distribution π and a set of inputs in the form of (α_i, λ_i) such that $\Pi'(\lambda_i)$=α_i and α_i∈[0, 1] meaning that

after revising π, the new possibility degree of λ_i is α_i. The revised possibility distribution π' according to Jeffrey's rule must satisfy the following conditions:

C1: $\forall\lambda_i, \Pi'(\lambda_i)=\alpha_i$.

C2: $\forall\lambda_i\subset\Omega, \forall\phi\subseteq\Omega, \Pi'(\phi|\lambda_i)=\Pi(\phi|\lambda_i)$.

As in the probabilistic setting, revising a possibility distribution π into π' according to the possibilistic counterpart of Jeffrey's rule must fully accept the inputs (condition C1) and preserve the fact that the uncertainty about the events λ_i must not alter the conditional possibility degree of any event $\phi\subseteq\Omega$ given any uncertain event λ_i (condition C2). The revision based on the possibilistic counterpart of Jeffrey's rule in the product-based possibilistic setting is performed as follows (Dubois and Prade 1997):

Definition 1. *Let π be a possibility distribution and (λ_1, α_1),..,(λ_n, α_n) be a set of exhaustive and mutually exclusive events where the uncertainty is of the form $\Pi'(\lambda_i)=\alpha_i$ for i=1..n. The revised possibility degree of any arbitrary event $\phi\subseteq\Omega$ is computed as follows (we assume that $\Pi(\phi)>0$):*

$$\forall\phi\subseteq\Omega, \Pi'(\phi)=\max_{\lambda_i}(\alpha_i * \frac{\Pi(\phi,\lambda_i)}{\Pi(\lambda_i)}). \quad (9)$$

It follows from Equation 9 that the revised possibility degree of any interpretation $\omega_j\in\Omega$ is computed as follows:

$$\forall\omega_j\in\lambda_i, \pi'(w_j)=\alpha_i * \frac{\pi(w_j)}{\Pi(\lambda_i)}.$$

It is shown in (Benferhat, Tabia, and Sedki 2011) that the revised possibility distribution π' computed according to Definition 1 always exists and it is unique.

Example 5. *In this example, we assume that we have beliefs over two binary variables A and B. The possibility distribution $\pi(AB)$ encodes the current beliefs. Table 3 gives the distribution π, the marginal distribution of A (namely, $\pi(A)$), the one of B (namely, $\pi(B)$) and the conditional distribution of B given A (namely, $\pi(B|A)$).*

A	B	$\pi(AB)$
a_1	b_1	1
a_2	b_1	0.4
a_1	b_2	0.1
a_2	b_2	0.4

A	$\pi(A)$
a_1	1
a_2	0.4

B	$\pi(B)$
b_1	1
b_2	0.4

A	B	$\pi(B\|A)$
a_1	b_1	1
a_2	b_1	1
a_1	b_2	0.1
a_2	b_2	1

Table 3: Example of initial possibility distribution π and the underlying marginal and conditional distributions.

Now assume that we have new beliefs in the form (a_1, .4) and (a_2, 1). The revised distribution using Jeffrey's rule of Equation 9 is given by π' of Table 4.

According to Tables 3 and 4, it is clear that the input beliefs are fully accepted (see the marginal distribution $\pi'(A)$) and that $\forall a_i\in D_A$, $\forall b_j\in D_B$, $\Pi(b_j|a_i)=\Pi'(b_j|a_i)$.

Pearl's method of virtual evidence in the quantitative possibilistic setting

In Pearl's virtual evidence method, the new information is a set of likelihood ratios γ_1,..,γ_n and satisfies the following condition:

A	B	$\pi'(AB)$
a_1	b_1	0.4
a_2	b_1	1
a_1	b_2	0.04
a_2	b_2	1

A	$\pi'(A)$
a_1	**0.4**
a_2	**1**

B	$\pi'(B)$
b_1	1
b_2	1

A	B	$\pi'(B\|A)$
a_1	b_1	1
a_2	b_1	1
a_1	b_2	0.1
a_2	b_2	1

Table 4: Revised beliefs of the initial distribution given in Table 3 using Jeffrey's rule of Equation 9.

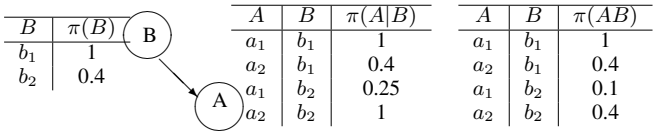
C3: γ_1:..:γ_n= $\Pi(\eta|\lambda_1)$:..:$\Pi(\eta|\lambda_n)$=$\frac{\Pi(\lambda_1|\eta)}{\Pi(\lambda_1)}$:..:$\frac{\Pi(\lambda_n|\eta)}{\Pi(\lambda_n)}$.

Pearl's virtual evidence method guarantees that the uncertainty bears only on the events λ_1,..,λ_n and does not concern the other events. Formally,

C4: $\forall\phi\subseteq\Omega, \Pi(\eta|\lambda_i,\phi)=\Pi(\eta|\lambda_i)$.

Pearl's method of virtual evidence applies in a quite straightforward way for quantitative possibilistic networks. Indeed, once the new inputs specified, they are integrated into the network G encoding the current beliefs in the form of a new node Z with a conditional possibility table designed in such a way that conditioning on the node Z, the conditional distribution $\pi_G(.|z)$ provides the revised joint distribution.

Example 6. *Let G be a possibilistic network over two binary variables A and B. The network G encodes the same possibility distribution as the distribution π of Table 3.*



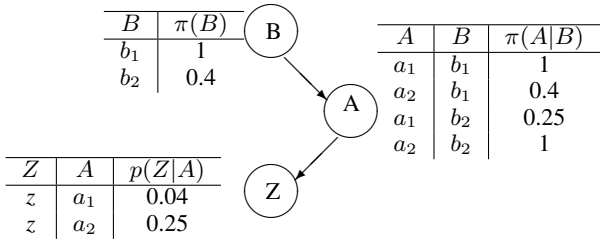
B	$\pi(B)$
b_1	1
b_2	0.4

A	B	$\pi(A\|B)$
a_1	b_1	1
a_2	b_1	0.4
a_1	b_2	0.25
a_2	b_2	1

A	B	$\pi(AB)$
a_1	b_1	1
a_2	b_1	0.4
a_1	b_2	0.1
a_2	b_2	0.4

Figure 4: Example of a possibilistic network G and the joint distribution $\pi(AB)$ encoded by G.

Let us assume now that new information says that γ_{a_1}:γ_{a_2}=.4:2.5. One solution satisfying this ratio is γ_{a_1}=.04 and γ_{a_2}=.25. Let us then add a new node Z to integrate γ_{a_1} and γ_{a_2}.



B	$\pi(B)$
b_1	1
b_2	0.4

A	B	$\pi(A\|B)$
a_1	b_1	1
a_2	b_1	0.4
a_1	b_2	0.25
a_2	b_2	1

Z	A	$p(Z\|A)$
z	a_1	0.04
z	a_2	0.25

Figure 5: G': The possibilistic network G of Figure 4 augmented with the node Z.

The revised beliefs are given in Table 5.

One can easily check that the revised distribution of Table 5 using Pearl's method of virtual evidence is exactly the same as the distribution π' obtained using Jeffrey's rule given in Table 4. It is also easy to check that conditions C3 and C4 are satisfied.

A	B	$\pi(AB\|z)$
a_1	b_1	0.4
a_2	b_1	1
a_1	b_2	0.04
a_2	b_2	1

Table 5: The conditional possibility distribution $\pi_{G'}(.|z)$ representing the revised distribution of the initial beliefs encoded by the network of Figure 4.

After addressing Pearl's method of virtual evidence in the quantitative possibilistic setting, let us see its generalization.

Virtual evidence method in the quantitative possibilistic setting

Here, the virtual evidence method applies on any possibility distribution exactly as Jeffrey's rule. The revised beliefs are computed according to the following definition.

Definition 2. *Let the initial beliefs be encoded by π and the new inputs be $\gamma_1,..,\gamma_n$. The revised possibility degree $\Pi'(\phi)$ of any event $\phi \subseteq \Omega$ is computed as follows:*

$$\forall \phi \subseteq \Omega, \Pi'(\phi) = \Pi(\phi|\eta) = \frac{\max_{i=1}^{n} \gamma_i * \Pi(\phi, \lambda_i)}{\max_{j=1}^{n} \gamma_j * \Pi(\lambda_j)}. \quad (10)$$

It is straightforward that revising the possibility degree of individual events $\omega_k \in \Omega$ is done as follows:

$$\forall \omega_k \in \lambda_i, \pi'(\omega_k) = \pi(\omega_k|\eta) = \frac{\gamma_i * \pi(\omega_k)}{\max_{j=1}^{n} \gamma_j * \Pi(\lambda_j)}. \quad (11)$$

Example 7. *Let the initial beliefs be encoded by the possibility distribution π of Table 6. Let also the likelihood ratios be $\gamma_1 = \frac{\Pi'(a_1)}{\Pi(a_1)} = .4$ and $\gamma_2 = \frac{\Pi'(a_2)}{\Pi(a_2)} = 2.5$ as in the example of Table 5. The revised distribution π' is computed using Equation 10.*

A	B	$\pi(AB)$
a_1	b_1	1
a_2	b_1	0.4
a_1	b_2	0.1
a_2	b_2	0.4

A	B	$\pi(AB\|\eta)$
a_1	b_1	0.4
a_2	b_1	1
a_1	b_2	0.04
a_2	b_2	1

Table 6: Example of initial possibility distribution π and the revised distribution $\pi(.|\eta)$.

The distribution π' computed using Equation 11 always exists and it is unique according the following proposition.

Proposition 1. *Let π be the possibility distribution encoding the initial beliefs. Let also $\gamma_1,..,\gamma_n$ be the likelihood ratios corresponding to the new inputs regarding the exhaustive and mutually exclusive set of events $\lambda_1,..,\lambda_n$. Then the revised possibility distribution π' computed using the formula of Equation 10 always exists and it is unique.*

Proof. Let π be the possibility distribution encoding the initial beliefs and let $\gamma_1,..,\gamma_n$ be the likelihood ratios regarding the events $\lambda_1,..,\lambda_n$.

1. Let us first show that the revised possibility distribution π' computed using the formula of Equation 10 satisfies the conditions **(C3)** and **(C4)**. Let us start proving that condition **(C3)** is satisfied.

$\Pi(\eta|\lambda_1) :..: \Pi(\eta|\lambda_n) = \frac{\Pi(\eta,\lambda_1)}{\Pi(\lambda_1)} :..: \frac{\Pi(\eta,\lambda_n)}{\Pi(\lambda_n)}$

$= \frac{\Pi(\lambda_1|\eta)*\Pi(\eta)}{\Pi(\lambda_n)} :..: \frac{\Pi(\lambda_n|\eta)*\Pi(\eta)}{\Pi(\lambda_n)}$

$= \frac{\frac{\gamma_1*\Pi(\lambda_1)}{\max_j(\gamma_j*\Pi(\lambda_j))}*\Pi(\eta)}{\Pi(\lambda_n)} :..: \frac{\frac{\gamma_n*\Pi(\lambda_n)}{\max_j(\gamma_j*\Pi(\lambda_j))}*\Pi(\eta)}{\Pi(\lambda_n)}$

$= \gamma_1 * \frac{\Pi(\eta)}{\max_j(\gamma_j*\Pi(\lambda_j))} :..: \gamma_n * \frac{\Pi(\eta)}{\max_j(\gamma_j*\Pi(\lambda_j))}$

$= \gamma_1 :..: \gamma_n$ □

Let us now prove that condition **(C4)** is satisfied.

$\forall \phi \subseteq \Omega, \Pi(\eta|\lambda_i, \phi) = \frac{\Pi(\eta,\lambda_i,\phi)}{\Pi(\lambda_i,\phi)}$

$= \frac{\Pi(\lambda_i,\phi|\eta)*\Pi(\eta)}{\Pi(\lambda_i,\phi)} = \frac{\frac{\gamma_i*\Pi(\lambda_i,\phi)}{\max_j(\gamma_j*\Pi(\lambda_j))}*\Pi(\eta)}{\Pi(\lambda_i,\phi)}$

$= \frac{\gamma_i*\Pi(\eta)}{\max_j(\gamma_j*\Pi(\lambda_j))} = \frac{\gamma_i*\frac{\Pi(\lambda_i,\eta)}{\Pi(\lambda_i|\eta)}}{\max_j(\gamma_j*\Pi(\lambda_j))}$

$= \frac{\gamma_i*\frac{\Pi(\lambda_i,\eta)}{\gamma_i*\Pi(\lambda_i)}}{\frac{\max_j(\gamma_j*\Pi(\lambda_j))}{\max_j(\gamma_j*\Pi(\lambda_j))}}$

$= \frac{\Pi(\lambda_i,\eta)}{\Pi(\lambda_i)} = \Pi(\eta|\lambda_i)$ □

2. Now let us provide the proof that if a distribution π' satisfies the conditions **(C3)** and **(C4)** then π' is computed using Equation 10.

$\forall \phi \subseteq \Omega, \Pi'(\phi) = \Pi(\phi|\eta) = \frac{\Pi(\phi,\eta)}{\Pi(\eta)}$

$= \frac{\max_{i=1}^{n}(\Pi(\phi,\lambda_i,\eta))}{\max_{j=1}^{n}(\Pi(\lambda_j,\eta))}$

$= \frac{\max_{i=1}^{n}(\Pi(\eta|\phi,\lambda_i)*\Pi(\phi,\lambda_i))}{\max_{j=1}^{n}(\Pi(\eta|\lambda_j)*\Pi(\lambda_j))}$

$= \frac{\max_{i=1}^{n}(\Pi(\eta|\lambda_i)*\Pi(\phi,\lambda_i))}{\max_{j=1}^{n}(\Pi(\eta|\lambda_j)*\Pi(\lambda_j))}$

$= \frac{\max_{i=1}^{n}(\gamma_i*\Pi(\phi,\lambda_i))}{\max_{j=1}^{n}(\gamma_j*\Pi(\lambda_j))}$ □

In the following, we provide the transformations from Jeffrey's rule to the virtual evidence method and vice versa.

From Jeffrey's rule to the virtual evidence method in a quantitative possibilistic setting

The following transformations are the possibilistic counterparts of the corresponding ones proposed in the probabilistic framework in (Chan and Darwiche 2005):

Proposition 2. *Let π be a possibility distribution encoding the initial beliefs and let also $\lambda_1,..,\lambda_n$ be an exhaustive and mutually exclusive set of events and new information in the form of (α_i, λ_i) such that for $i=1..n$, $\Pi'(\lambda_i)=\alpha_i$. Let $\gamma_1,..,\gamma_n$ be likelihood ratios such that*

$$\gamma_1 : .. : \gamma_n = \frac{\alpha_1}{\Pi(\lambda_1)} : .. : \frac{\alpha_n}{\Pi(\lambda_n)},$$

then the revised possibility distribution π'_J computed using Jeffrey's rule of Equation 9 and the revised possibility distribution π'_P computed using the virtual evidence method of Equation 10 are equivalent. Namely, $\forall\omega\in\Omega$, $\pi'_J(\omega)=\pi'_P(\omega)$.

Proof sketch. The proof is direct. Just set in the virtual evidence method of Equation 10 $\gamma_i=\frac{\alpha_i}{\Pi(\lambda_i)}$ for i=1..n, and the obtained distribution π'_P satisfies conditions C1 and C2 of Jeffrey's rule and since the revised distribution with Jeffrey's rule is unique then π'_P equals π'_J. □

From the virtual evidence method to Jeffrey's rule in a quantitative possibilistic setting

We show now how to obtain the new beliefs α_1,..,α_n needed in Jeffrey's rule from the available set of likelihood ratios:

Proposition 3. *Let π be a possibility distribution encoding the initial beliefs. Let also λ_1,..,λ_n be an exhaustive and mutually exclusive set of events and new information in the form of likelihood ratios γ_1,..,γ_n. For i=1..n, let $\alpha_i=\gamma_i*\Pi(\lambda_i)$. Then the revised possibility distribution π'_J computed using Jeffrey's rule of Equation 9 and the revised possibility distribution π'_P computed using the virtual evidence method of Equation 10 are equivalent. Namely, $\forall\omega\in\Omega$, $\pi'_J(\omega)=\pi'_P(\omega)$.*

Proof sketch. The proof is similar to the proof of Proposition 2. Using Jeffrey's rule of Equation 9 with the inputs (λ_i, α_i) for i=1..n such that $\alpha_i=\gamma_i*\Pi(\lambda_i)$ and the obtained distribution π'_J satisfies conditions C3 and C4 of the virtual evidence method and since the revised distribution is also unique then π'_J equals π'_P. □

Reasoning with uncertain inputs in the qualitative possibilistic setting

Jeffrey's rule in the qualitative possibilistic setting

In the qualitative setting, the revision according to Jeffrey's rule is performed as follows (Dubois and Prade 1997):

Definition 3. *Let π be a possibility distribution and λ_1,..,λ_n be a set of exhaustive and mutually exclusive events. The revised possibility degree of any arbitrary event $\phi\subseteq\Omega$ is computed using the following formula:*

$$\forall\phi\subseteq\Omega, \Pi'(\phi)=\max_{\lambda_i}(\min(\Pi(\phi|\lambda_i),\alpha_i)). \quad (12)$$

It is straightforward that for elementary events ω_j, the revised beliefs are computed according the following formula:

$$\forall w_j\in\lambda_i, \pi'(w_j)=\begin{cases}\alpha_i & \text{if } \pi(w_j)\geq\alpha_i \text{ or } \pi(w_j)=\Pi(\lambda_i);\\ \pi(w_j) & \text{otherwise.}\end{cases}$$

Contrary to the probabilistic and quantitative possibilistic settings, there exist situations where the revision according to Equation 12 does not guarantee the existence of a solution satisfying conditions C1 and C2 (Benferhat, Tabia, and Sedki 2011).

Example 8. *Assume that we have beliefs in the form of a possibility distribution $\pi(AB)$ over two binary variables A and B (we have the same beliefs as in Table 3). In Table 7, we have the joint distribution $\pi(AB)$, the marginal distributions $\pi(A)$ and $\pi(B)$) and the conditional one $\pi(B|A)$.*

A	B	$\pi(AB)$
a_1	b_1	1
a_2	b_1	0.4
a_1	b_2	0.1
a_2	b_2	0.4

A	$\pi(A)$
a_1	1
a_2	0.4

B	$\pi(B)$
b_1	1
b_2	0.4

A	B	$\pi(B\|A)$
a_1	b_1	1
a_2	b_1	1
a_1	b_2	0.1
a_2	b_2	1

Table 7: Example of initial possibility distribution π and the underlying marginal and conditional distributions.

Assume now that we want to revise π of Table 7 into π' such $\pi'(a_1)$=.4 and $\pi'(a_2)$=1. The revised distribution using the qualitative counterpart of Jeffrey's rule of Equation 12 is given by π' Table 8.

A	B	$\pi'(AB)$
a_1	b_1	0.4
a_2	b_1	1
a_1	b_2	0.1
a_2	b_2	1

A	$\pi'(A)$
a_1	0.4
a_2	1

B	$\pi'(B)$
b_1	1
b_2	1

A	B	$\pi'(B\|A)$
a_1	b_1	1
a_2	b_1	1
a_1	b_2	0.1
a_2	b_2	1

Table 8: Revised beliefs of the initial distribution given in Table 7 using Jeffrey's rule of Equation 12.

According to the results of Table 7 and 8, it is clear that in this example conditions C1 and C2 are fulfilled.

Pearl's method of virtual evidence in the qualitative possibilistic setting

As in the quantitative setting, the inputs are specified in the same way. Namely, the uncertainty bears on an exhaustive and mutually exclusive set of events λ_1,..,λ_n and the new information is specified as likelihood ratios γ_1:..:γ_n according to condition C3. As shown in the following example, unlike the probabilistic and quantitative possibilistic settings, it is not enough for the parameters γ_1,..,γ_n to satisfy condition C3 to be directly integrated into the conditional possibility table of the new node Z.

Example 9. *Let G be the possibilistic network of Figure 6 and the corresponding joint distribution π_G.*

B	$\pi(B)$
b_1	1
b_2	0.4

B → A

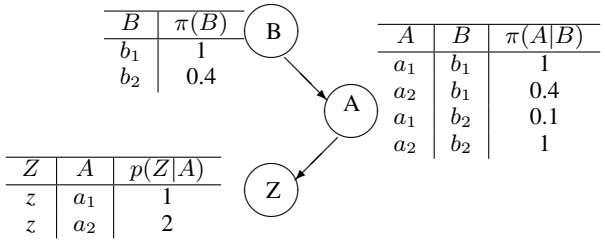
A	B	$\pi(A\|B)$
a_1	b_1	1
a_2	b_1	0.4
a_1	b_2	0.1
a_2	b_2	1

A	B	$\pi_G(AB)$
a_1	b_1	1
a_2	b_1	0.4
a_1	b_2	0.1
a_2	b_2	0.4

Figure 6: Example of a possibilistic network G and the joint distribution $\pi_G(AB)$ encoded by G in the qualitative setting.

Let us assume now that we want to revise the distribution π_G encoded by the network G of Figure 6 into a new distribution π'_G such that γ_{a_1}=1 and γ_{a_2}=2 meaning that the initial belief degree of a_1 is not changed while the degree of a_2 is to be doubled. The augmented network G' encoding the new inputs is shown in Figure 7.

The revised beliefs are given in Table 9.



B	$\pi(B)$
b_1	1
b_2	0.4

A	B	$\pi(A\|B)$
a_1	b_1	1
a_2	b_1	0.4
a_1	b_2	0.1
a_2	b_2	1

Z	A	$p(Z\|A)$
z	a_1	1
z	a_2	2

Figure 7: The possibilistic network G' obtained by augmenting G of Figure 6 with the node Z.

A	B	$\pi_{G'}(AB\|z)$
a_1	b_1	1
a_2	b_1	.4
a_1	b_2	.1
a_2	b_2	.4

Table 9: The conditional distribution $\pi_{G'}(.|z)$ representing the revised distribution of the initial beliefs of Figure 6.

One can notice from the results of Table 9 that condition C3 is not satisfied since $\frac{\Pi_{G'}(a_2|z)}{\Pi_G(a_2)}=\frac{.4}{.4}\neq\gamma_{a_2}=2$.

Hence, instead of using only the inputs $\gamma_1,..,\gamma_n$, the conditional possibility table of the new node Z must be set for each uncertain event λ_i directly to $\Pi'(\lambda_i)=\gamma_i*\Pi(\lambda_i)$ as in Jeffrey's rule. This is imposed by the min-based operator and the definition of conditioning in the qualitative possibilistic setting (see Equations 3 and 4). Clearly, if a parameter $\pi_{G'}(z|a_i)>1$ then it is not taken into account (note that in the probabilistic and quantitative possibilistic settings, the values of γ_i are not necessarily in the interval [0, 1] and they are always taken into account thanks to the definition of conditioning in these settings). Now, even when replacing γ_i by $\gamma_i*\Pi(\lambda_i)$ for i=1..n, it is impossible with Pearl's method of virtual evidence to increase the plausibility of an event as stated in the following proposition.

Proposition 4. *Let G be a* min*-based network and let π_G be the possibility distribution encoded by G. Let also $\gamma_1,..,\gamma_n$ be the likelihood ratios corresponding to the new inputs regarding the exhaustive and mutually exclusive set of events $a_1,..,a_n$. Let G' be the augmented possibilistic network with the virtual node Z to encode $\gamma_1,..,\gamma_n$ such that for i=1..n, $\pi_{G'}(Z=z|a_i)=\Pi_G(a_i)*\gamma_i$. Then we have two cases:*

- *If $\forall i$=1..n, $\gamma_i\leq 1$, then conditions C3 and C4 are satisfied.*
- *Otherwise if $\exists\gamma_i>1$ then the revised possibility degree $\Pi_{G'}(a_i|z)=\Pi_G(a_i)$ implying that C3 is not satisfied while C4 is always satisfied.*

Proposition 4 states that associating with an uncertain event a_i a possibility degree of $\Pi_G(a_i)*\gamma_i$ in the augmented network G', the posterior possibility degree $\Pi_{G'}(a_i|z)$ equals $\Pi_G(a_i)$ (unless $\Pi_{G'}(a_i|z)$ is the greatest one in the context of z in which case $\Pi(a_i|z)$=1 because of normalization). As a consequence of Proposition 4, it is impossible to augment the possibility degree of an event a_i unless $\Pi(a_i)\geq\Pi(a_j)$ for any $j\neq i$ meaning that condition C3 is not satisfied. Indeed, because of the idempotency of the min-based operator used in the min-based chain rule of Equation 4 and the definition of the min-based conditioning of Equation 3, applying directly Pearl's method of virtual evidence does not guarantee that condition C3 will be satisfied. However, condition C4 is always satisfied as it is implied by a graphical property.

Proof sketch. The proof follows from the min-based chain rule and the augmented network G'. Indeed,
$\Pi'_G(a_i|z)=\Pi(a_i,z)=\max_{A_1..a_i..A_n}(\Pi(A_1..a_i..A_n,z))$
$=\max_{A_1..a_i..A_n}(\min(\pi(A_1|U_1),..,\pi(a_i|U_i),..,\pi(A_n|U_n),\pi(z|a_i))$
$\leq\pi(z|a_i)=\gamma_i*\Pi_G(a_i)$ □
However, one can show that due to the encoding of the inputs by means of augmenting the network, for every event ϕ, $\forall a_i\in D_i$, $\Pi(z|a_i,\phi)=\Pi(z|a_i)$ (due to d-separation) meaning that condition C4 is always satisfied since it is a graphical property of the augmented network G'. □

Virtual evidence method in the qualitative possibilistic setting

The min-based counterpart of the quantitative possibilistic virtual evidence method of Definition 2 is defined as follows:

Definition 4. *Let the initial beliefs be encoded by π and the new inputs be $\gamma_1,..,\gamma_n$ specified as likelihood ratios $\gamma_1:..:\gamma_n$ such that $\gamma_i=\Pi(\eta|\lambda_i)=\frac{\Pi(\lambda_i|\eta)}{\Pi(\lambda_i)}$. The revised possibility degree $\Pi'(\phi)$ of any event $\phi\subseteq\omega$ is computed as follows:*

$$\forall\phi\subseteq\Omega,\Pi(\phi|\eta)=\max_{i=1}^{n}(\min(\Pi(\phi|\lambda_i),\gamma_i*\Pi(\lambda_i)) \quad (13)$$

For single interpretations $\omega_k\in\Omega$, the revised degrees are computed as follows:

$$\forall\omega_k\in\lambda_i,\pi'(\omega_k)=\min(\Pi(\omega_k|\lambda_i),\gamma_i*\Pi(\lambda_i)) \quad (14)$$

Example 10. *Let us reuse the beliefs given Table 7 as initial beliefs. Assume now that we want to revise π of Table 7 into π' such $\gamma_{a_1}=\frac{\pi'(a_1)}{\pi(a_1)}$=.75 and $\gamma_{a_2}=\frac{\pi'(a_2)}{\pi(a_2)}$=2.5. The revised distribution using the qualitative counterpart of the virtual evidence method of Equation 13 is given by π' Table 10.*

A	B	$\pi'(AB)$
a_1	b_1	0.75
a_2	b_1	1
a_1	b_2	0.1
a_2	b_2	1

A	$\pi'(A)$
a_1	0.75
a_2	1

γ_{a_1}	.75
γ_{a_2}	2.5

Table 10: Revised beliefs of the initial distribution given in Table 7 using the virtual evidence method of Equation 13.

From the results of Table 7 and 10, one can easily check that the conditions C3 and C4 are satisfied.

The distribution π' computed using Equation 13 always satisfy conditions C3 and C4 as stated in Proposition 5.

Proposition 5. *Let π be the possibility distribution encoding the initial beliefs. Let also $\gamma_1,..,\gamma_n$ be the likelihood ratios corresponding to the new inputs regarding the exhaustive and mutually exclusive set of events $\lambda_1,..,\lambda_n$. Then the revised possibility distribution π' computed using the formula of Equation 13 always satisfy conditions C3 and C4.*

Proof sketch. Let π be the possibility distribution encoding the initial beliefs and let γ_1,..,γ_n be the likelihood ratios corresponding to the new inputs regarding the set of exhaustive and mutually exclusive set of events λ_1,..,λ_n such that for i=1..n, γ_i=$\frac{\Pi'(\lambda_i)}{\Pi(\lambda_i)}$. Let us first show that the revised possibility distribution π' computed using the formula of Equation 13 satisfies condition **(C3)**.

$\Pi(\eta|\lambda_1)$:..: $\Pi(\eta|\lambda_n) = \Pi(\lambda_1, \eta)$:..: $\Pi(\lambda_n|\eta)$
$=\min(\Pi(\lambda_1|\eta), \Pi(\eta))$:..: $\min(\Pi(\lambda_n|\eta), \Pi(\eta))$
$=\min(\max_i(\Pi(\lambda_1|\lambda_i), \gamma_i * \Pi(\lambda_i)))$:..:$\min(\max_i(\Pi(\lambda_n|\lambda_i), \gamma_i * \Pi(\lambda_i)))$
$=\min(\Pi(\lambda_1|\lambda_1), \gamma_1 * \Pi(\lambda_1))$:..:$\min(\Pi(\lambda_n|\lambda_n), \gamma_n * \Pi(\lambda_n))$
$=\gamma_1 * \Pi(\lambda_1)$:..:$\gamma_n * \Pi(\lambda_n)$. □

The proof that π' satisfies condition C4 is similar to the proof of Proposition 1. Let us now provide the proof that if a distribution π' satisfies the conditions **(C3)** and **(C4)** then π' is computed using Equation 13.

$\forall\phi\subseteq\Omega$, $\Pi'(\phi)$= $\Pi(\phi|\eta)$=$\max_{\lambda_i}(\Pi(\phi|\eta, \lambda_i))$
$= \max_{\lambda_i}(\min(\Pi(\phi|\lambda_i), \Pi(\lambda_i|\eta))$
$= \max_{\lambda_i}(\min(\Pi(\phi|\lambda_i), \gamma_i * \Pi(\lambda_i))$ □

Relating the virtual evidence method with Jeffrey's rule in the qualitative possibilistic setting

As in the quantitative setting, it is straightforward to move from Jeffrey's rule to the virtual evidence method and vice versa (because of space limitation, we provide only basic results, the propositions and their proofs are similar to the corresponding transformations in the quantitative setting).

1. *From the virtual evidence method to Jeffrey's rule*: Just set the inputs α_i=γ_i*$\Pi(\lambda_i)$ for i=1..n and use Jeffrey's rule of Equation 12 will give exactly the same revised distribution π' as the distribution $\pi(.|\eta)$ obtained by the min-based possibilistic counterpart of the virtual evidence method of Equation 13.
2. *From Jeffrey's rule to the virtual evidence method* : Here, it is enough to set the inputs γ_i=α_i/$\Pi(\lambda_i)$ for i=1..n then use the virtual evidence method of Equation 13 to obtain $\pi(.|\eta)$ which is equivalent to π' obtained using Jeffrey's rule of Equation 12.

Discussions and concluding remarks

In order to revise the beliefs encoded by means of a possibility distribution one can either use Jeffrey's rule or the virtual evidence method which are shown equivalent in both the quantitative and qualitative settings. However, revising a whole distribution is very costly while Pearl's method of virtual evidence allows to integrate the inputs and compute any possibility degree of interest directly from the network without revising the whole distribution. Moreover, the existing inference algorithms in graphical models (e.g. Junction tree) can be used directly to compute the revised beliefs.

This paper addressed reasoning with uncertain inputs in possibilistic networks. We provided possibilistic counterparts for Pearl's methods and compared them with the well-known Jeffrey's rule of conditioning. In Jeffrey's rule, the inputs (α_i, λ_i) are seen as constraints that should be satisfied leading to fully accepting the new beliefs. The way Jeffrey's method revises the old distribution π in order to fully accept the inputs (α_i, λ_i) complies with the probability kinematics principle (see condition C2) aiming to minimize belief change. In spite of the fact that Pearl's methods specify the inputs differently, the way the new inputs are graphically taken into account (see condition C4) complies with the probability kinematics principle hence minimizing also belief change. Regarding accepting the inputs, it is clear that even specified differently, the inputs to both methods are fully accepted (see conditions C1 and C3).

Regarding iterative revisions, it is well-known that Jeffrey's rule is no commutative (since the new inputs are fully accepted, then revising first with (λ_i, α_i) then with (λ_i, α'_i) will be different from first revising with (λ_i, α'_i) then with (λ_i, α_i)). However in the virtual evidence method, due to the commutativity of multiplication and the definition of the quantitative counterpart of this revision rule, it is easy to show that revising with a set of likelihood ratios γ_1,..,γ_n then revising the resulted beliefs with other new inputs γ'_1,..,γ'_n will give exactly the same results as revising first with γ'_1,..,γ'_n followed by revision with γ_1,..,γ_n. Revision using the qualitative virtual evidence method is not commutative because the inputs are no more likelihood ratios γ_1,..,γ_n but the new beliefs which are fully accepted as in Jeffrey's rule.

To sum up, the contributions of the paper are:

1. Providing counterparts to Pearl's method of virtual evidence and its generalization in the quantitative and qualitative settings. We showed that contrary to the probabilistic and quantitative possibilistic settings, the inputs for the qualitative counterparts of Pearl's methods should be possibility degrees satisfying condition C3. This is due to the fact that in possibilistic networks, we deal with local tables combined with the min-based operator which is idempotent and because of the definition of the qualitative conditioning. We showed also that it is impossible to enhance the possibility degree of an event using Pearl's method of virtual evidence in qualitative networks.
2. Analyzing the existence and uniqueness of the solutions using the proposed possibilistic counterparts of Pearl's methods. In the quantitative setting, the paper showed that the solution always exists and it is unique. In the min-based setting however, depending on the inputs, the solution is not guaranteed to satisfy conditions C3 using Pearl's method of virtual evidence while using the virtual evidence method the solution always exists and it is unique and satisfies conditions C3 and C4.
3. Providing transformations from Jeffrey's rule to the virtual evidence method and comparisons of these methods in both the quantitative and qualitative settings. We provided precise conditions where the methods are equivalent. Finally, we tried to relate the criteria underlying Jeffrey's rule and Pearl's methods and highlighted many related issues like iterated revisions in these formalisms.

References

Benferhat, S.; Dubois, D.; Prade, H.; and Williams, M.-A. 2010. A framework for iterated belief revision using possibilistic counterparts to jeffrey's rule. *Fundam. Inform.* 99(2):147–168.

Benferhat, S.; Da Costa Pereira, C.; and Tettamanzi, A. G. B. 2013. Syntactic computation of hybrid possibilistic conditioning under uncertain inputs. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, 739–745. AAAI Press.

Benferhat, S.; Tabia, K.; and Sedki, K. 2011. Jeffrey's rule of conditioning in a possibilistic framework. *Annals of Mathematics and Artificial Intelligence* 61(3):185–202.

Bilmes, J. 2004. On virtual evidence and soft evidence in bayesian networks. Technical Report UWEETR-2004-0016, Dept of EE, University of Washington, Seattle WA, 98195-2500.

Chan, H., and Darwiche, A. 2005. On the revision of probabilistic beliefs using uncertain evidence. *Artificial Intelligence* 163(1):67–90.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Dubois, D., and Prade, H. 1988a. *Possibility theory*. Plenum Press, New-York.

Dubois, D., and Prade, H. 1988b. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenium Press.

Dubois, D., and Prade, H. 1993. Belief revision and updates in numerical formalisms: an overview, with new results for the possibilistic framework. In *Proceedings of the 13th international joint conference on Artifical intelligence - Volume 1*, IJCAI'93, 620–625. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Dubois, D., and Prade, H. 1994. A survey of belief revision and updating rules in various uncertainty models. *International Journal of Intelligent Systems* 9(1):61–100.

Dubois, D., and Prade, H. 1997. A synthetic view of belief revision with uncertain inputs in the framework of possibility theory. *Int. J. of Approximate Reasoning* 17(2–3):295–324.

Fonck, P. 1997. A comparative study of possibilistic conditional independence and lack of interaction. *International Journal of Approximate Reasoning* 16(2):149–171.

Hisdal, E. 1978. Conditional possibilities independence and non interaction. *Fuzzy Sets and Systems* 283–297.

Jeffrey, R. C. 1965. *The Logic of Decision*. NY: McGraw Hill.

L.M. De Campos, J. H., and Moral, S. 1995. Possibilistic independence. *Proceedings of EUFIT 95* 1:69–73.

Ma, J., and Liu, W. 2011. A framework for managing uncertain inputs: An axiomization of rewarding. *Int. J. Approx. Reasoning* 52(7):917–934.

Pan, R.; Peng, Y.; and Ding, Z. 2006. Belief update in bayesian networks using uncertain evidence. In *ICTAI*, 441–444. IEEE Computer Society.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Shafer, G. 1976. *A Mathematical Theory of Evidence*. Princeton: Princeton University Press.

Spohn, W. 1988. Ordinal conditional functions: A dynamic theory of epistemic states. In *Causation in decision, belief change, and statistics*, volume II. Kluwer Academic Publishers. 105–134.

Vomlel, J. 2004. Probabilistic reasoning with uncertain evidence. *Neural Network World, International Journal on Neural and Mass-Parallel Computing and Information Systems* 14(5):453–456.

Yager, R. 1983. An introduction to applications of possibility theory. *Human Systems Management* 3:246–269.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Relational Logistic Regression*

Seyed Mehran Kazemi, David Buchman, Kristian Kersting, Sriraam Natarajan, and David Poole
cs.ubc.ca/~smkazemi/ cs.ubc.ca/~davidbuc/
www-ai.cs.uni-dortmund.de/PERSONAL/kersting.html
homes.soic.indiana.edu/natarasr/ cs.ubc.ca/~poole/

Abstract

Logistic regression is a commonly used representation for aggregators in Bayesian belief networks when a child has multiple parents. In this paper we consider extending logistic regression to relational models, where we want to model varying populations and interactions among parents. In this paper, we first examine the representational problems caused by population variation. We show how these problems arise even in simple cases with a single parametrized parent, and propose a linear relational logistic regression which we show can represent arbitrary linear (in population size) decision thresholds, whereas the traditional logistic regression cannot. Then we examine representing interactions among the parents of a child node, and representing non-linear dependency on population size. We propose a multi-parent relational logistic regression which can represent interactions among parents and arbitrary polynomial decision thresholds. Finally, we show how other well-known aggregators can be represented using this relational logistic regression.

Introduction

Relational probabilistic models are models where there are probabilities about relations among individuals that can be specified independently of the actual individuals, and where the individuals are exchangeable; before we know anything about the individuals, they are treated identically. One of the features of relational probabilistic models is that the predictions of the model may depend on the number of individuals (the population size) (Poole et al. 2012). Sometimes, this dependence is desirable; in other cases, model weights may need to change (Jian, Bernhard, and Beetz 2007; Jian, Barthels, and Beetz 2009). In either case, it is important to understand how the predictions change with population size.

Varying population sizes are quite common. They can appear in a number of ways including:

- The actual population may be arbitrary. For example, in considering the probability of someone committing a crime (which depends on how many other people could have committed the crime) (Poole 2003) we could consider the population to be the population of the neighbourhood, the population of the city, the population of the country, or the population of the whole world. It would be good to have a model that does not depend on this arbitrary decision. We would like to be able to compare models which involve different choices.
- The population can change. For example, the number of people in a neighbourhood or in a school class may change. We would like a model to make reasonable predictions as the population changes. We would also like to be able to apply a model learned at one or a number of population sizes to different population sizes. For example, models from drug studies are acquired from very limited populations but are applied much more generally.
- The relevant populations can be different for each individual. For example, the happiness of a person may depend on how many of her friends are kind (and how many are not kind). The set of friends is different for each individual. We would like a model that makes reasonable predictions for diverse numbers of friends.

In this paper, we consider applying standard logistic regression to relational domains and tasks and investigate how varying populations can cause a problem for logistic regression. Then we propose *single-parent linear relational logistic regression* which solves this problem with standard logistic regression by taking the population growth into account. This representation is, however, only able to model linear function dependencies of the child on its parents' population sizes. Also when used for multiple parents, it cannot model the interactions among the parents. We examine these two limitations and propose a general relational logistic regression which we prove can represent arbitrary Boolean formulae among the parents as well as every polynomial dependency of the child node on its parents' population sizes. We also show how other well-known aggregators can be represented using our polynomial relational logistic regression.

Our model assumes all the parent variables are categorical and the child variable is Boolean. Extending the model to multi-valued child variables and continuous parent variables (as done by Mitchell (2010) for non-relational models) is left as a future work.

*This work was supported in part by the Institute for Computing Information and Cognitive Systems (ICICS) at UBC, NSERC, MITACS and the German Science Foundation (DFG), KE 1686/2-1.
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Background

Bayesian Belief Networks

Suppose we have a set of random variables $\{X_1,\ldots,X_n\}$. A **Bayesian network (BN)** or **belief network** (Pearl 1988) is an acyclic directed graph where the random variables are the nodes, and the arcs represent interdependence between the random variables. Each variable is independent of its non-descendants given values for its parents. Thus, if X_i is not an ancestor of X_j, then $P(X_i \mid parents(X_i), X_j) = P(X_i \mid parents(X_i))$. The joint probability of the random variables can be factorized as:

$$P(X_1,X_2,...,X_n) = \prod_{i=1}^{n} P(X_i \mid parents(X_i))$$

One way to represent a conditional probability distribution $P(X_i \mid parents(X_i))$ is in terms of a table. Such a tabular representation for a random variable increases exponentially in size with the number of parents. For instance, a Boolean child having 10 Boolean parents requires $2^{10} = 1024$ numbers to specify the conditional probability. A compact alternative to a table is an **aggregation** operator, or aggregator, that specifies a function of how the distribution of a variable depends on the values of its parents. Examples for common aggregators include OR, AND, as well as "noisy-OR" and "noisy-AND". These can be specified much more compactly than as a table.

Logistic Regression

Suppose a Boolean random variable Q is a child of the numerical random variables $\{X_1,X_2,\ldots,X_n\}$. Logistic regression is an aggregation operator defined as:

$$P(q \mid X_1,\ldots,X_n) = \text{sigmoid}(w_0 + \sum_i w_i X_i) \qquad (1)$$

where $q \equiv$ "$Q = True$" and $\text{sigmoid}(x) = 1/(1+e^{-x})$. It follows that $P(q \mid X_1,\ldots,X_n) > 0.5$ iff $w_0 + \sum_i w_i X_i > 0$.

The space of assignments to the w's so that $w_0 + \sum_i w_i X_i = 0$ is called the **decision threshold**, as it is the boundary of where $P(q \mid X_1,\ldots,X_n)$ changes between being closer to 0 and being closer to 1. Logistic regression provides a soft threshold, in that it changes from close to 0 to close to 1 in a continuous manner. How fast it changes can be adjusted by multiplying all weights by a positive constant.

The Factorization Perspective

A simple and general formulation of logistic regression can be defined using a multiplicative factorization of the conditional probability. (1) then becomes a special case, which is equivalent to the general case when variables are binary and probabilities are positive (non-zero).

We define a **general logistic regression** for Q with parents $X_1,\ldots,X_n$ (all variables here may be discrete or continuous) to be when $P(Q \mid X_1,\ldots,X_n)$ can be factored into a product of non-negative pairwise factors and a non-negative factor for Q:

$$P(Q \mid X_1,\ldots,X_n) \propto f_0(Q) \prod_{i=1}^{n} f_i(Q,X_i)$$

where $\propto$ *(proportional-to)* means it is normalized separately for each assignment to the parents. This differs from the normalization for joint distributions (as used in undirected models), where there is a single normalizing constant. Here the constraint that causes the normalization is $\forall X_1,\ldots,X_n : \sum_Q P(Q \mid X_1,\ldots,X_n) = 1$, whereas for joint distributions, the normalization is to satisfy the constraint $\sum_{Q,X_1,\ldots,X_n} P(Q,X_1,\ldots,X_n) = 1$.

If Q is binary, then:

$$P(q \mid X_1,\ldots,X_n) = \frac{f_0(q)\prod_{i=1}^{n} f_i(q,X_i)}{f_0(q)\prod_{i=1}^{n} f_i(q,X_i) + f_0(\neg q)\prod_{i=1}^{n} f_i(\neg q,X_i)}$$

If all factors are positive, we can divide and then use the identity $y = e^{\ln y}$:

$$\begin{aligned} P(q \mid X_1,\ldots,X_n) &= \frac{1}{1 + \frac{f_0(\neg q)}{f_0(\ \ q)} \prod_{i=1}^{n} \frac{f_i(\neg q,X_i)}{f_i(\ \ q,X_i)}} \\ &= \frac{1}{1 + exp\left(\ln \frac{f_0(\neg q)}{f_0(\ \ q)} + \sum_{i=1}^{n} \ln \frac{f_i(\neg q,X_i)}{f_i(\ \ q,X_i)}\right)} \\ &= \text{sigmoid}\left(\ln \frac{f_0(\ \ q)}{f_0(\neg q)} + \sum_{i=1}^{n} \ln \frac{f_i(\ \ q,X_i)}{f_i(\neg q,X_i)}\right). \end{aligned}$$

When the $\ln \frac{f_i(\ \ q,X_i)}{f_i(\neg q,X_i)}$ are linear functions w.r.t. X_i, it is possible to find values for all w's such that this can be represented by Eq. (1). This is always possible when the parents are binary.

The idea of *relational* logistic regression is to extend logistic regression to relational models, by allowing weighted logical formulae to represent the factors in the factorization of a conditional probability.

Relational Models

Relational probabilistic models (Getoor and Taskar 2007) or template based models (Koller and Friedman 2009) extend Bayesian or Markov networks by adding the concepts of individuals (objects, entities, things), relations among individuals (including properties, which are relations of a single individual) and by allowing for probabilistic dependencies among these relations. In these models, individuals about which we have the same information are exchangeable, meaning that, given no evidence to distinguish them, they should be treated identically. We provide some basic definitions and terminologies in these models which are used in the rest of the paper.

A **population** is a set of **individuals**. A population corresponds to a domain in logic. The **population size** is the cardinality of the population which can be any non-negative integer.

A **logical variable** is written in lower case. Each logical variable is typed with a population; we use $|x|$ for the size of the population associated with a logical variable x. Constants, denoting individuals, start with an upper case letter.

A **parametrized random variable (PRV)** is of the form $F(t_1,\ldots,t_k)$ where F is a k-ary functor (a function symbol or a predicate) and each t_i is a logical variable or a constant. Each functor has a range, which is $\{True, False\}$ for predicate symbols. A PRV represents a set of random variables,

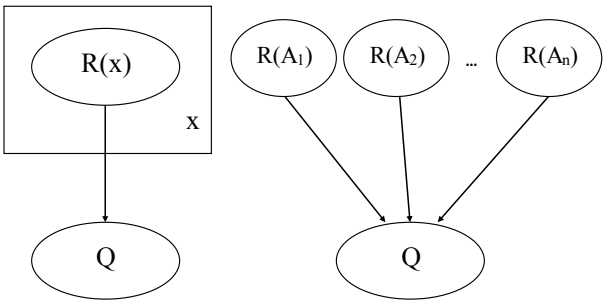


Figure 1: logistic regression (with i.i.d. priors for the $R(x)$). The left side is the relational model in plate notation and on the right is the groundings for the population $\{A_1, A_2, \ldots, A_n\}$.

one for each assignment of individuals to its logical variables. The range of the functor becomes the range of each random variable.

A **relational belief network** is an acyclic directed graph where the nodes are PRVs. A **grounding** of a relational belief network with respect to a population for each logical variable is a belief network created by replacing each PRV with the set of random variables it represents, while preserving the structure.

A **formula** is made up of assignments of values to PRVs with logical connectives. For a Boolean PRV $R(x)$, we represent $R(x) = \mathit{True}$ by $R(x)$ and $R(x) = \mathit{False}$ by $\neg R(x)$.

When using a single population, we write the population as $A_1 \ldots A_n$, where n is the population size, and use $R_1 \ldots R_n$ as short for $R(A_1) \ldots R(A_n)$. We also use n_{val} for the number of individuals x for which $R(x) = val$. When $R(x)$ is binary, we use the shortened $n_T = n_{\mathit{True}}$ and $n_F = n_{\mathit{False}}$.

Relational Logistic Regression

While aggregation is optional in non-relational models, it is necessary in directed relational models whenever the parent of a PRV contains extra logical variables. For example, suppose Boolean PRV Q is a child of the Boolean PRV $R(x)$, which contains an extra logical variable, x, as in Figure 1. In the grounding, Q is connected to n instances of $R(x)$, where n is the population size of x. For the model to be defined before n is known, it needs to be applicable for all values of n.

Common ways to aggregate the parents in relational domains, e.g. (Horsch and Poole 1990; Friedman et al. 1999; Neville et al. 2005; Perlish and Provost 2006; Kisynski and Poole 2009; Natarajan et al. 2010), include logical operators such as *OR*, *AND*, *noisy-OR*, *noisy-AND*, as well as ways to combine probabilities.

Logistic regression, as described above, may also be used for relational models. Since the individuals in a relational model are exchangeable, w_i must be identical for all parents R_i (this is known as parameter-sharing or weight-tying), so Eq. (1) becomes:

$$P(q \mid R_1, \ldots, R_n) = \text{sigmoid}(w_0 + w_1 \sum_i R_i). \qquad (2)$$

Consider what happens with a relational model when n is not fixed.

Example 1. Suppose we want to represent "Q is True if and only if R is True for 5 or more individuals", i.e., $q \equiv \left(|\{i : R_i = \mathit{True}\}| \geq 5\right)$ or $q \equiv (n_T \geq 5)$, using a logistic regression model $(P(q) \geq 0.5) \equiv (w_0 + w_1 \sum_i R_i \geq 0)$, which we fit for a population of 10. Consider what this model represents when the population size is 20.

If $R = \mathit{False}$ is represented by 0 and $R = \mathit{True}$ by 1, this model will have $Q = \mathit{True}$ when R is true for 5 or more individuals out of the 20. It is easy to see this, as $\sum_i R_i$ only depends on the number of individuals for which R is *True*.

However, if $R = \mathit{False}$ is represented by -1 and $R = \mathit{True}$ by 1, this model will have $Q = \mathit{True}$ when R is *True* for 10 or more individuals out of the 20. The sum $\sum_i R_i$ depends on how many more individuals have *R True* than have *R False*.

If $R = \mathit{True}$ is represented by 0 and $R = \mathit{False}$ by any other value, this model will have $Q = \mathit{True}$ when R is *True* for 15 or more individuals out of the 20. The sum $\sum_i R_i$ depends on how many individuals have *R False*.

While the choice of representation for *True* and *False* was arbitrary in the non-relational case, in the relational case different parametrizations can result in different decision thresholds as a function of the population. The following table gives five numerical representations for *False* and *True*, with corresponding parameter settings (w_0 and w_1), such that all regressions represent the same conditional distribution for $n = 10$. However, for $n = 20$, the predictions are different:

False	True	w_0	w_1	Prediction for n = 20
0	1	-4.5	1	$Q \equiv (n_T \geq 5)$
-1	1	0.5	0.5	$Q \equiv (n_T \geq 10)$
-1	0	5.5	1	$Q \equiv (n_T \geq 15)$
-1	2	$-\frac{7}{6}$	$\frac{1}{3}$	$Q \equiv (n_T \geq 8)$
1	2	-14.5	1	$Q \equiv (n_T \geq 0)$

The decision thresholds in all of these are linear functions of population size. It is straightforward to prove the following proposition:

Proposition 1. *Let $R = \mathit{False}$ be represented by the number α and $R = \mathit{True}$ by $\beta \neq \alpha$. Then, for fixed w_0 and w_1 (e.g., learned for one specific population size), the decision threshold for a population of size n is*

$$\frac{w_0}{w_1(\alpha - \beta)} + \frac{\alpha}{\alpha - \beta} n.$$

What is important about this proposition is that the way the decision threshold changes with the population size n, i.e., the coefficient $\frac{\alpha}{\alpha - \beta}$, does not depend on data (which affects the weights w_0 and w_1), but only on the arbitrary choice of the numerical representation of R.

Thus, Eq. (2) with a specific numeric representation of *True* and *False* is only able to model one of the dependencies

of how predictions depend on population size, and so cannot properly fit data that does not adhere to that dependence.

We need an additional degree of freedom to get a relational model that can model any linear dependency on n, regardless of the numerical representation chosen.

Definition 1. Let Q be a Boolean PRV with a single parent $R(x)$, where x is the set of logical variables in R that are not in Q (so we need to aggregate over x). A (single-parent, linear) **relational logistic regression (RLR)** for Q with parents $R(x)$ is of the form:

$$P(q \mid R(A_1),\ldots,R(A_n)) = \text{sigmoid}\left(w_0 + w_1 \sum_i R_i + w_2 \sum_i (1-R_i)\right) \quad (3)$$

where R_i is short for $R(A_i)$, and is treated as 1 when it is *True* and 0 when it is *False*. Note that $\sum_i R_i$ is the number of individuals for which R is *True* ($= n_T$) and $\sum_i (1-R_i)$ is the number of individuals for which R is *False* ($= n_F$).

An alternative but equivalent parametrization is:

$$P(q \mid R(A_1),\ldots,R(A_n)) = \text{sigmoid}(w_0 + w_2 \sum_i 1 + w_3 \sum_i R_i) \quad (4)$$

where 1 is a function that has value 1 for every individual, so $\sum_i 1 = n$. The mapping between these parametrizations is $w_3 = w_1 - w_2$; w_0 and w_2 are the same.

Proposition 2. *Let $R = False$ be represented by α and $R = True$ by $\beta \neq \alpha$. Then, for fixed w_0, w_2 and w_3 in Eq. (4), the decision threshold for a population of size n is*

$$\frac{w_0}{w_3(\alpha-\beta)} + \frac{\alpha + w_2/w_3}{\alpha-\beta} n.$$

Proposition 2 implies that the way the decision threshold in a single-parent linear RLR grows with the population size n, i.e. the coefficient $\frac{\alpha+w_2/w_3}{\alpha-\beta}$, depends on the weights. Moreover, for fixed α and β, any linear function of population can be modeled by varying the weights. This was not true for the traditional logistic regression.

For the rest of this paper, when we embed logical formulae in arithmetic expressions, we take *True* formulae to represent 1, and *False* formulae to represent 0. Thus $\sum_L F$ is the number of assignments to the variables L for which formula F is *True*.

Interactions Among Parents

The RLR proposed in Definition 1 can be extended to multiple (parametrized) parents by having a different pair of weights ((w_1,w_2) or (w_2,w_3)) for each parent PRV. This is similar to the non-relational logistic regression, where each parent has a (single) different weight. However, there are cases where we want to model the interactions among the parents.

Example 2. Suppose we want to model whether someone being happy depends on the number of their friends that are kind. We assume the variable $Happy(x)$ has as parents $Friend(y,x)$ and $Kind(y)$. Note that the number of friends for each person can be different.

Consider the following hypotheses:

(a) A person is happy as long as they have 5 or more friends who are kind.

$$happy(x) \equiv |\{y : Friend(y,x) \wedge Kind(y)\}| \geq 5$$

(b) A person is happy if half or more of their friends are kind.

$$happy(x) \equiv |\{y : Friend(y,x) \wedge Kind(y)\}| \geq |\{y : Friend(y,x) \wedge \neg Kind(y)\}|$$

(c) A person is happy as long as fewer than 5 of their friends are not kind.

$$happy(x) \equiv |\{y : Friend(y,x) \wedge \neg Kind(y)\}| < 5$$

These three hypotheses coincide for people with 10 friends, but make different predictions for people with 20 friends.

All three hypotheses are based on the interaction between the two parents. Linear RLR considers each parent separately from the others, and so cannot model these interactions without introducing new relations. In order to model such aggregators, we need to be able to count the number of instances of a formula that are *True* in an assignment to the parents. We can use the following extended RLR to model these cases:

$$P(happy(x) \mid \Pi) = \text{sigmoid}\Big(w_0 + w_1 \sum_y Friend(y,x) \wedge \quad Kind(y) + w_2 \sum_y Friend(y,x) \wedge \neg Kind(y) \Big) \quad (5)$$

where Π is a complete assignment of *friend* and *kind* to the individuals, and the right hand side is summing over the propositions in Π for each individual. To model each of the above three cases, we can set w_0, w_1, and w_2 in Eq. (5) as follows:

(a) Let $w_0 = -4.5$, $w_1 = 1$, $w_2 = 0$
(b) Let $w_0 = 0.5$, $w_1 = 1$, $w_2 = -1$
(c) Let $w_0 = 5.5$, $w_1 = 0$, $w_2 = -1$

Going from Eq. (3) to Eq. (4) allowed us to only model the positive cases in linear single-parent RLR. We can use a similar construction for more general cases:

Example 3. Suppose a PRV Q is a child of PRVs $R(x)$ and $S(x)$. We want to represent "Q is *True* if and only if there are more than t individuals for x for which $R(x) \wedge \neg S(x)$." As in Example 2, we need to count the number of instances of a formula that are *True* in an assignment to the parents. It turns out that in this case $R(x) \wedge S(x)$ is the only non-atomic formula required to model the interactions between the two parents, because other conjunctive interactions can be represented using this count as follows:

$$\sum_x R(x) \wedge \neg S(x) = \sum_x R(x) - \sum_x R(x) \wedge S(x)$$
$$\sum_x \neg R(x) \wedge S(x) = \sum_x S(x) - \sum_x R(x) \wedge S(x)$$
$$\sum_x \neg R(x) \wedge \neg S(x) = |x| - \sum_x R(x) - \sum_x S(x) + \sum_x R(x) \wedge S(x)$$

with $|x| = \sum_x True$.

Example 3 shows that the positive conjunction of the interacting parents is the only formula required to compute arbitrary conjunctions of the two parents. In more complicated cases, however, subtle changes to the representation may be required.

Example 4. Suppose a PRV Q is a child of PRVs $R(x,y)$ and $S(x,z)$. Suppose we want to represent "Q is *True* if and only if we have $R(x,y) \wedge \neg S(x,z)$ for more than 5 triples $\langle x,y,z \rangle$". If we only count the number of instances of $(R \wedge T)(x,y,z)$ that are *True* given the assignment to the parents and do the same as in Example 3, we would only count the number of pairs $\langle x,y \rangle$ for which $R(x,y)$ is *True*. However, we need the number of triples $\langle x,y,z \rangle$ for which $R(x,y)$ is *True*. We thus need to use $\sum_{x,y,z} R(x,y)$ as the number of assignments to x, y and z, for which $R(x,y)$ is *True*, as follows:

$$\sum_{x,y,z} R(x,y) \wedge \neg S(x,z) = \sum_{x,y,z} R(x,y) - \sum_{x,y,z} R(x,y) \wedge S(x,y,z)$$

So as part of the representation, we need to incude the set of logical variables, and not just a weighted formula.

Non-Linear Decision Thresholds

Examples 3 and 4 suggest how to model interactions among the parents. Now consider the case where the child PRV is a non-linear function of its parents' population sizes. For instance, if the individuals are the nodes in a dense graph, some properties of arcs grow with the square of the population of nodes.

Markov logic networks (MLNs) (Richardson and Domingos 2006; Domingos et al. 2008) define probability distributions over worlds (complete assignments to the ground model) in terms of weighted formulae. The probability of a world is proportional to the exponential of the sum of the weights of the instances of the formulae that are *True* in the world. The probability of any formula is obtained by summing over the worlds in which the formula is *True*. MLNs can also be adapted to define conditional distributions. The following example shows a case where a non-linear conditional distribution is modeled by MLNs.

Example 5. Consider the MLN for PRVs Q and $R(x)$, consisting of a single formula $Q \wedge R(x) \wedge R(y)$ with weight w, where y represents the same population as x. The probability of q given observations of $R(A_i)$ for all A_i has a quadratic decision threshold:

$$P(q \mid R(A_1), \ldots, R(A_n)) = \text{sigmoid}(w\, {n_T}^2).$$

A similar method can be used by RLR to model non-linear decision thresholds. Consider the following example:

Example 6. Suppose a PRV Q is a child of the PRV $R(x)$, and we want to represent "Q is *True* if and only if ${n_T}^2 > n_F$". This dependency can be represented using the single-parent linear RLR by introducing a new logical variable x' with the same population as x and treating $R(x')$ as if it were a separate parent of Q. Then we can use the interaction between $R(x)$ and $R(x')$ to represent the model in this example as:

$$\sum_{x,x'} R(x) \wedge R(x') - \sum_{x} \mathit{True} + \sum_{x} R(x).$$

General Relational Logistic Regression (RLR)

The previous examples show the potential for using RLR as an aggregator for relational models. We need a language for representing aggregation in relational models in which we can address the problems mentioned. We propose a generalized form of RLR which works for multi-parent cases and can model polynomial decision thresholds.

Definition 2. A **weighted parent formula (WPF)** for a PRV $Q(x)$, where x is a set of logical variables, is a triple $\langle L, F, w \rangle$ where L is a set of logical variables for which $L \cap x = \{\}$, F is a Boolean formula of parent PRVs of Q such that each logical variable in F either appears in Q or is in L, and w is a weight. Only those logical variables that do not appear in Q are allowed to be substituted in F.

Definition 3. Let $Q(x)$ be a Boolean PRV with parents $R_i(x_i)$, where x_i is the set of logical variables in R_i. A (multi-parent, polynomial) **relational logistic regression (RLR)** for Q with parents $R_i(x_i)$ is defined using a set of WPFs as:

$$P(q(X) \mid \Pi) = \text{sigmoid}\Big(\sum_{\langle L,F,w \rangle} w \sum_{L} F_{\Pi, x \to X} \Big)$$

where Π represents the assigned values to parents of Q, X represents an assignment of an individual to each logical variable in x, and $F_{\Pi, x \to X}$ is formula F with each logical variable x in it being replaced according to X, and evaluated in Π. (The first summation is over the set of WPFs; the second summation is over the tuples of L. Note that $\sum_{\{\}}$ sums over a single instance.)

Since the logical variables that appear in $Q(x)$ are fixed in the formulae and not allowed to be substituted according to Definition 2, in the rest of the paper we only focus on those logical variables of the parents that do not appear in $Q(x)$.

The single-parent linear RLR (Definition 1) is a subset of Definition 3, because the terms of Eq. (4) can be modeled by WPFs:

- w_0 can be represented by $\langle \{\}, \mathit{True}, w_0 \rangle$
- $w_2 \sum_i 1$ can be represented by $\langle \{x\}, \mathit{True}, w_2 \rangle$
- $w_3 \sum_i R_i$ can be represented by $\langle \{x\}, R(x), w_3 \rangle$

RLR then sums these WPFs, resulting in:

$$\begin{aligned} P(q \mid \Pi) &= \text{sigmoid}\Big(w_0 \sum_{\{\}} \mathit{True} + w_1 \sum_{\{x\}} \mathit{True} + w_2 \sum_{\{x\}} R(x)\Big) \\ &= \text{sigmoid}\Big(w_0 + w_2 n + w_3 \sum_i R_i\Big). \end{aligned}$$

Example 7. Consider the problem introduced in Example 4. Using general RLR (Definition 3), we can model the conditional probability of Q using the following WPFs:

$$\langle \{\}, \mathit{True}, w_0 \rangle$$
$$\langle \{x,y,z\}, R(x,y) \wedge \neg S(y,z), w_1 \rangle$$

Or alternatively:

$$\langle \{\}, \mathit{True}, w_0 \rangle$$
$$\langle \{x,y,z\}, R(x,y), w_1 \rangle$$
$$\langle \{x,y,z\}, R(x,y) \wedge S(y,z), -w_1 \rangle$$

Canonical Forms for RLR

While in Definition 2 we allow for any Boolean formula of parents, we can prove that a positive conjunctive form is sufficient to model all the Boolean interactions among parents. A Boolean interaction is one that can be expressed in terms of logical connectives of values to the parents.

Proposition 3. *Let Q be a Boolean PRV with parents $R_i(x_i)$, where x_i is a set of logical variables in R_i which are not in Q. Using only positive conjunctive form formulae in the WPFs for Q, all Boolean interactions between the parents can be modeled by RLR.*

Proof. We show how to model every conjunctive form interaction between the parents. Other interactions (such as a disjunctive interaction) can be modeled by a set of conjunctives.

For a subset M of parents of Q, we prove by induction on the number of negations $j \leq |M|$, that every conjunctive form interaction between the parents having j negations can be modeled by a set of WPFs.

For $j = 0$, the formula F is in a positive conjunction form and the proposition holds (even if $M = \{\}$). Assume the proposition holds for $j < |M|$. For $j+1$, let $R_i(x_i)$ be one of the negated parents. Removing $\neg R_i(x_i)$ from the formula F gives a new formula F_1 with j negated parents. According to our assumption for j negations, there exists a set S_1 of WPFs that models F_1. Replacing $\neg R_i(x_i)$ in F by $R_i(x_i)$ gives a new formula F_2 with j negated parents, which can be modeled by some set S_2 of WPFs. Let S_1' represent the WPFs in S_1 where each set of logical variables L in each WPFs is replaced by $L \cup x_i$. F can then be modeled by combining S_1' and S_2 and negating the weights associated with S_2. The reason why this is correct can be seen in Example 4. □

Proposition 3 suggests using only positive conjunctive formuale in WPFs. Proposition 4 proves that positive disjunctive RLR has the same representational power as positive conjunctive RLR. Therefore, all propositions proved for positive conjunctive RLR in the rest of the paper also hold for positive disjunctive RLR.

Proposition 4. *A conditional distribution $P(Q \mid R_i(x_i))$ can be expressed by a positive disjunctive RLR if and only if it can be expressed by a positive conjunctive RLR.*

Proof. First, suppose $P(Q \mid R_i(x_i))$ can be expressed by a positive disjunctive RLR. We can write a disjunctive formula as a negated conjunctive formula. So we change all the disjunctive formulae in the WPFs for $P(Q \mid R_i(x_i))$ to negated conjunctive formulae. A negated conjunctive formula in WPF $\langle L, \neg F, w \rangle$ can be modeled by two conjunctive WPFs $\langle L, T, w \rangle$ and $\langle L, F, -w \rangle$. The latter WPF consists of negated parents but we know from Proposition 3 that we can model it by a set of positive conjunctive WPFs. Consequently, $P(Q \mid R_i(x_i))$ can be also expressed by a positive conjunctive RLR.

Now, suppose the conditional distribution can be expressed by a positive conjunctive RLR definition of $P(Q \mid R_i(x_i))$. While Proposition 3 is written for positive conjunctive RLR, it is straight forward to see that it also holds for negative conjunctive RLR. This means that we can express Q by WPFs having negative conjunctive formulae. We can represent each of these formulae in a negated positive disjunctive form. We also mentioned that a negated WPF $\langle L, \neg F, w \rangle$ can be expressed by two WPFs $\langle L, T, w \rangle$ and $\langle L, F, -w \rangle$. The former does not contain any parent and the latter is in positive disjunctive form. Consequently, $P(Q \mid R_i(x_i))$ can be also expressed by a positive disjunctive RLR. □

Buchman et al. (2012) looked at canonical representations for probability distributions with binary variables in the non-relational case. Our positive conjunctive canonical form corresponds to their "canonical parametrization" with a "reference state" *True* (i.e., in which all variables are assigned *True*), and our positive disjunctive canonical form has a connection to using a "reference state" *False*. Their "spectral representation" would correspond to a third positive canonical form for RLR, in terms of **XORs** (i.e., parity functions).

Polynomial Decision Thresholds

We can also model polynomial decision thresholds using RLR. The following example is a case where the child PRV depends on $|x|^2$.

Example 8. Suppose Q is a Boolean PRV with a parent $R(x)$, where x is a set of logical variables in R which are not in Q. By having a WPF $\langle \{x, x'\}, R(x) \wedge R(x'), w \rangle$ for Q where x' is typed with the same population as x, the conditional probability of Q depends on the square of the number of assignments to x for which $R(x)$ is *True*.

Example 8 represents a case where the conditional probability of a child PRV is a non-linear function of its parent's population size. We can prove that by using only positive conjunctive formulae in the WPFs of a child PRV, we can model any polynomial decision threshold. First we prove this for the single-parent case and then for the general case of multi-parents. We assume in the following propositions that Q is a Boolean PRV and $R_i(x_i)$ are its parents where x_i is the set of logical variables in R_i which are not in Q. We also use x_i' to refer to a new logical variable typed with the same population as x_i.

Proposition 5. *A positive conjunctive RLR definition of $P(Q \mid R(x))$ (single-parent case) can represent any decision threshold that is a polynomial of terms each indicating a number of (tuples of) individuals for which R(x) is True or False.*

Proof. Based on Proposition 3 we know that a WPF having any Boolean formula can be written as a set of WPFs each having a positive conjunctive formula. Therefore, in this proof we disregard using only positive conjunctives. The final set of WPFs can be then represented by a set of positive conjunctive WPFs using Proposition 3.

Each term of the polynomial in the single-parent case is of the form $w(\prod_i |y_i|^{d_i}) {n_T}^{\alpha} {n_F}^{\beta}$, where n_T and n_F denote the number of individuals for which $R(x)$ is *True* or *False* respectively, $y_i \in x$ represents the logical variables in x, α, β and d_is are non-negative integers, and w is the weight of the

term. First we prove by induction that for any j, there is a WPF that can build the term ${n_T}^{\alpha}{n_F}^{\beta}$ where $\alpha+\beta=j$, $\alpha \geq 0$ and $\beta \geq 0$.

For $j=0$, ${n_T}^{\alpha}{n_F}^{\beta}=1$. We can trivially build this by WPF $\langle\{\},True,w\rangle$. Assuming it is correct for j, we prove it for $j+1$. For $j+1$, either $\alpha>0$ or $\beta>0$. If $\alpha>0$, using our assumption for j, we can have a WPF $\langle L,F,w\rangle$ which builds the term ${n_T}^{\alpha-1}{n_F}^{\beta}$. So the WPF $\langle L\cup x',F\wedge R(x'),w\rangle$ builds the term ${n_T}^{\alpha}{n_F}^{\beta}$ because the first WPF was *True* ${n_T}^{\alpha-1}{n_F}^{\beta}$ times and now we count it n_T more times because $R(x')$ is *True* n_T times.

If $\alpha=0$ and $\beta>0$, we can have a WPF $\langle L,F,w\rangle$ which builds the term ${n_T}^{\alpha}{n_F}^{\beta-1}$. By the same reasoning as in previous case, we can see that the WPF $\langle L\cup\{x'\},F\wedge\neg R(x'),w\rangle$ produces the term ${n_T}^{\alpha}{n_F}^{\beta}$.

In order to include the population size of logical variables y_i, where $y_i \in x$, and generate the term $(\prod_i |y_i|^{d_i}){n_T}^{\alpha}{n_F}^{\beta}$, we only add d_i extra logical variables y_i' to the set of logical variables of the WPF that generates ${n_T}^{\alpha}{n_F}^{\beta}$. Then we set the weight of this WPF to w to generate the desired term. □

Conclusion. We can conclude from this proposition that a term $w(\prod_i |y_i|^{d_i}){n_T}^{\alpha}{n_F}^{\beta}$ can be generated by having a WPF with its formula consisting of n_T instances of $R(x')$ and n_F instances of $\neg R(x')$, adding d_i of each logical variable y_i to the set of logical variables, and setting the weight of WPF to w. We will use this conclusion for proving the proposition in multi-parent case.

Example 9. Suppose we want to model the case where Q is *True* if ${n_T}^2 \geq 2n_F+5$. In this case, we need to model the sigmoid of the polynomial ${n_T}^2-2n_F-4.5$. The reason for using 4.5 instead of 5 is to make the polynomial positive when ${n_T}^2=2n_F+5$. The following WPFs are used by RLR to model this polynomial. The first one generates the term ${n_T}^2$, the second one generate $-2n_F$ and the third one generates -4.5.

$$\langle\{x,x'\},R(x)\wedge R(x'),1\rangle$$
$$\langle\{x\},\neg R(x),-2\rangle$$
$$\langle\{\},True,-4.5\rangle$$

Note that the second WPF above can be written in positive form by using the following two WPFs:

$$\langle\{x\},True,-2\rangle$$
$$\langle\{x\},R(x),2\rangle$$

Using the conclusion following Proposition 5, we now extend Proposition 5 to the multi-parent case:

Proposition 6. *A positive conjunctive RLR definition of $P(Q \mid R_i(x_i))$ (multi-parent case) can represent any decision threshold that is a polynomial of terms each indicating the number of (tuples of) individuals for which a Boolean function of parents hold.*

Proof. Let $G_1,G_2,\ldots,G_t$ represent the desired Boolean functions of parents for our model. Also let $n_{(i)}$ denote the number of individuals for which G_i is *True*. Each term of the polynomial in the multi-parent case is then of the form: *w * (a polynomial of population sizes) ** $n_{(1)}^{\alpha_1}n_{(2)}^{\alpha_2}\ldots n_{(t)}^{\alpha_t}$. We demonstrate how we can generate any term $n_{(1)}^{\alpha_1}n_{(2)}^{\alpha_2}\ldots n_{(t)}^{\alpha_t}$. The inclusion of population size of logical variables and the weight w for each term is the same as in Proposition 5.

The conclusion of Proposition 5 can be easily generalized to work for any Boolean formula G_i instead of R. We only need to include a conjunction of α_i instances of G_i with different logical variables representing the same population in each instance. We use this generalization in our proof.

For each $n_{(i)}^{\alpha_i}$, we can use the generalization of conclusion of the Proposition 5 to come up with a WPF $\langle L_i,F_i,1\rangle$ which generates this term. Similar to the reasoning for single-parent case, we can see that the WPF $\left\langle\{\cup_{i=1}^{t}L_i,F_1\wedge F_2\wedge\cdots\wedge F_t,w\right\rangle$ generates the term $n_{(1)}^{\alpha_1}n_{(2)}^{\alpha_2}\ldots n_{(t)}^{\alpha_t}$. We can then use Proposition 3 to write this WPF using only positive conjunctive WPFs. □

Example 10. Suppose we want to model the case where Q is *True* if the square of number of individuals for which $R_1(x_1)=True$ multiplied by the number of individuals for which $R_2(x_2)=True$ is less than five times the whole number of *False* individuals in $R_1(x_1)$ and $R_2(x_2)$. In this case, we define $G_1=\neg R_1(x_1)$, $G_2=\neg R_2(x_2)$ and $G_3=R_1(x_1)\wedge R_1(x_1')\wedge R_2(x_2)$ we need to model the sigmoid of the polynomial $5n_{(1)}+5n_{(2)}-n_{(3)}-0.5$. The reason why we use -0.5 in the polynomial is that we want the polynomial to be negative when $5n_{(1)}+5n_{(2)}=n_{(3)}$. The following WPFs are used by RLR to model this polynomial where the first formula generates the term $5n_{(1)}$, the second generates $5n_{(2)}$, the third generates $-n_{(3)}$ and the fourth generates -0.5.

$$\langle\{x_1\},\neg R_1(x_1),5\rangle$$
$$\langle\{x_2\},\neg R_2(x_2),5\rangle$$
$$\langle\{x_1,x_1',x_2\},R_1(x_1)\wedge R_1(x_1')\wedge R_2(x_2),-1\rangle$$
$$\langle\{\},True,-0.5\rangle$$

Note that the first and second WPF above can be written in positive form in the same way as in Example 9.

Proposition 7 proves the converse of Proposition 6:

Proposition 7. *Any decision threshold that can be represented by a positive conjunctive RLR definition of $P(Q \mid R_i(x_i))$ is a polynomial of terms each indicating the number of (tuples of) individuals for which a Boolean function of parents hold.*

Proof. We prove that every WPF for Q can only generate a term of the polynomial. Since RLR sums over these terms, it will always represent the polynomial of these terms.

Similar to Proposition 6, let $G_1,G_2,\ldots,G_t$ represent the desired Boolean functions of parents and let $k_{(i)}$ denote the number of individuals for which G_i is *True*. A positive conjunctive formula in a WPF can consist of α_1 instances of G_1, α_2 instances of G_2, …, α_t instances of G_t. Based on Proposition 6, we know that this formula is *True* $k_{(1)}^{\alpha_1}k_{(2)}^{\alpha_2}\ldots k_{(t)}^{\alpha_t}$ times. The WPF can contain more logical variables in its set of logical variables than the ones in its formula. This, however, will only cause the above term to be multiplied by the

population size of the logical variable generating a term of the polynomial described in Proposition 6. Therefore, each of the WPFs can only generate a term of the polynomial which means that positive conjunctive RLR can only represent the sigmoid of this polynomial. □

Approximating Other Aggregators Using RLR

We can model other well-known aggregators using positive conjunctive RLR. In most cases, however, this is only an approximation because the sigmoid function reaches 0 or 1 only asymptotically and we cannot choose infinitely large numbers. We can, however, get arbitrarily close to 0 or 1 by choosing arbitrarily large weights. In the rest of this section, we use M to refer to a number which can be set sufficiently large to receive the desired level of approximation. n_{val} is the number of individuals x for which $R(x) = val$, when R is not Boolean.

OR. To model OR in RLR, we use the WPFs:

$$\langle\{\}, True, -M\rangle$$
$$\langle\{x\}, S(x), 2M\rangle$$

for which $P(q \mid S(x)) = \text{sigmoid}(-M + 2Mk)$. We can see that if none of the individuals are *True* (i.e. $n_T = 0$), the value inside the sigmoid is $-M$ which is a negative number thus the probability is close to 0. If even one individual is *True* (i.e. $n_T \geq 1$), the value inside the sigmoid becomes positive and the probability becomes closer to 1. In both cases, the value inside the sigmoid is a linear function of M. Increasing M pushes the probability closer to 0 or to 1 and the approximation becomes more accurate.

AND. AND can be modeled similarly to OR, but with the WPFs

$$\langle\{\}, True, M\rangle$$
$$\langle\{x\}, True, -2M\rangle$$
$$\langle\{x\}, S(x), 2M\rangle$$

for which $P(q \mid S(x)) = \text{sigmoid}(M - 2Mn_F)$. When $n_F = 0$, the value inside the sigmoid is $M > 0$, so the probability is closer to 1. When $n_F \geq 1$, the value inside the sigmoid becomes negative and the probability becomes closer to 0. Like OR, accuracy increases with M. Note that the number of WPFs for representing *OR* and *AND* is fixed.

Noisy-OR and noisy-AND. Figure 2 (a) represents how noisy-OR and noisy-AND can be modeled for the network in Figure 1. In this figure, $R(x)$ represents the values of the individuals being combined and $N(x)$ represents the noise probability. For noisy-OR, $S(x) \equiv R(x) \wedge N(x)$, and Q is the OR aggregator of $S(x)$. For noisy-AND, $S(x) \equiv R(x) \vee N(x)$, and Q is the AND aggregator of $S(x)$.

Mean $>$ t. We can model "Q is *True* if $\text{mean}(R(x)) > t$" using the following WPFs (*val* and t are numeric constants):

$$\langle\{\}, True, -M\rangle$$
$$\langle\{x\}, R(x) = val, M^2(val - t)\rangle \quad \text{for each } val \in \text{range}(R)$$

for which

$$P(q \mid R(x)) = \text{sigmoid}(-M + M^2 \sum_{val \in \text{range}(R)} n_{val}(val - t))$$
$$= \text{sigmoid}(-M + M^2(sum - nt))$$

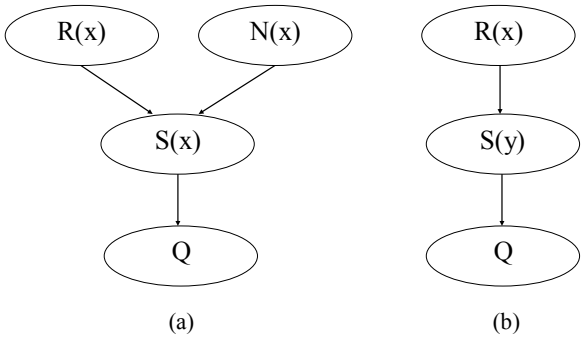


Figure 2: (a) The model for the noisy-OR and noisy-AND aggregators (b) The model for the mode aggregator

where $n = |x|$ and *sum* represents the sum of the values of the individuals. When $mean = \frac{sum}{n} > t$, the value inside the sigmoid is positive and the probability is close to 1. Otherwise, the value inside the sigmoid is negative and the probability is close to 0. Note that M should be greater than $|\frac{1}{sum - nt}|$ to generate a number greater than 1 when multiplied by $(sum - nt)$. Otherwise, it may occur that $sum - nt > 0$ but $M^2(sum - nt) \leq M$ which makes the sigmoid produce a number close to 0. Also note that the number of required WPFs grows with the number of values that the parent can take.

More than t *True*s. "Q is *True* if R is *True* for more than t individuals" can be modeled using the WPFs:

$$\langle\{\}, True, -2Mt - M\rangle$$
$$\langle\{x\}, R(x), 2M\rangle$$

giving $P(q \mid R(x)) = \text{sigmoid}(-2Mt - M + 2Mn_T)$ and the value inside the sigmoid is positive if $n_T > t$. The number of WPFs required is fixed.

More than t% *True*s. "Q is *True* if R is *True* for more than t percent of the individuals" is a special case of the aggregator "mean $> \frac{t}{100}$" when we treat *False* values as 0 and *True* values as 1. This directly provides the WPFs:

$$\langle\{\}, True, -M\rangle$$
$$\left\langle\{x\}, \neg R(x), M^2(0 - \tfrac{t}{100})\right\rangle$$
$$\left\langle\{x\}, \quad R(x), M^2(1 - \tfrac{t}{100})\right\rangle$$

while requiring $M > |\frac{1}{n_T - nt/100}|$, where n is the populations size of x. Note that we can use Proposition 3 to replace the second WPF with two WPFs having positive conjunctive formulae. Unlike the aggregator "mean $> \ldots$", here the number of WPFs is fixed.

Max $>$ t. We can model "Q is *True* if $\max(R(x)) > t$" using the following WPFs:

$$\langle\{\}, True, -M\rangle$$
$$\langle\{x\}, R(x) = val, 2M\rangle \quad \text{for each } val > t,\ val \in \text{range}(R)$$

thus $P(q \mid R(x)) = \text{sigmoid}(-M + 2M \sum_{val > t \in \text{range}(R)} n_{val})$. The value inside the sigmoid is positive if there is an individual having a value greater then t (i.e. $\exists val > t \in \text{range}(R)$:

$n_{val} > 0$). Note that the number of WPFs required grows with the number of values greater than t that the parents can take.

Max. For binary parents, the "max" aggregator is identical to "OR". Otherwise, $\text{range}(Q) = \text{range}(R(x))$. The "max" aggregator can be modeled using a 2-level structure. First, for every $t \in \text{range}(R(x))$, create a separate "max $\geq t$" aggregator, with $R(x)$ as its parents. Then, define the child Q, with all the "max $\geq t$" aggregators as its parents. Q can compute $\max R(x)$ given its parents. Note that while $|parents(Q)| = |\text{range}(R(x))|$ may be arbitrarily large, $|parents(Q)|$ does not change with population size, hence it is possible to use non-relational constructs (e.g., a table) for its implementation.

Mode = t. To model "Q is *True* if $\text{mode}(R(x)) = t$", we first add another PRV $S(y)$ to the network as in Figure 2 (b) where y represents the range of the values for $R(x)$. Then for each individual $S(C)$ of $S(y)$, we use the following WPFs for which $P(s(C) \mid R(x)) = \text{sigmoid}(M - 2M(n_C - n_t))$ and the value inside the sigmoid is positive if $n_t \geq n_C$. Note that the number of WPFs required grows with the number of values that the parent can take.

$$\langle \{\}, True, M \rangle$$
$$\langle \{x\}, R(x) = C, -2M \rangle$$
$$\langle \{x\}, R(x) = t, 2M \rangle$$

Then Q must be *True* if all the individuals in S are *True*. This is because a *False* value for an individual of S means that this individual has occurred more than t and t is not the mode. Therefore, we can use WPFs similar to the ones we used for AND:

$$\langle \{\}, True, M \rangle$$
$$\langle \{y\}, True, -2M \rangle$$
$$\langle \{y\}, S(y), 2M \rangle$$

Mode & Majority. For binary parents, the "mode" aggregator is also called "majority", and can be modeled with the "more than t% Trues" aggregator, with $t = 50$. Otherwise, $\text{range}(Q) = \text{range}(R(x))$, and we can use the same approach as for "max", by having $|\text{range}(R(x))|$ separate "mode $= t$" aggregators, with Q as their child.

Beyond Polynomial Decision Thresholds

Proposition 7 showed that any conditional probability that can be expressed using a positive conjunctive RLR definition of $P(Q \mid R_i(x_i))$ is the sigmoid of a polynomial of the number of *True* and *False* individuals in each parent $R_i(x_i)$. However, given that the decision thresholds are only defined for integral counts, some of the apparently non-polynomial decision thresholds are equivalent to a polynomial and so can be modeled using RLR.

Example 11. Suppose we want to model $Q \equiv \left(\left\lceil \sqrt{n_T} \right\rceil < n_F \right)$. This is a non-polynomial decision threshold, but since n_T and n_F are integers, it is equivalent to the polynomial decision threshold $n_T - (n_F - 1)^2 \leq 0$ which can be formulated using RLR.

Example 12. Suppose we want to model $Q \equiv (2^{n_T} > 3^{n_F})$. This is, however, equivalent to the polynomial form $Q \equiv (n_T \log 2 - n_F \log 3 > 0)$ and can be formulated in positive conjunctive RLR using the WPFs:

$$\langle \{x\}, True, -\log 3 \rangle$$
$$\langle \{x\}, R(x), \quad \log 3 + \log 2 \rangle$$

There are, however, non-polynomial decision thresholds that cannot be converted into a polynomial one and RLR is not able to formulate them.

Example 13. Suppose we want to model $Q \equiv (2^{n_T} > n_F)$. This cannot be converted to a polynomial form and RLR cannot formulate it.

Finding a parametrization that allows to model any non-polynomial decision threshold remains an open problem.

Conclusion

Today's data and models are complex, composed of objects and relations, and noisy. Hence it is not surprising that relational probabilistic knowledge representation currently receives a lot of attention. However, relational probabilistic modeling is not an easy task and raises several novel issues when it comes to knowledge representation:

- What assumptions are we making? Why should we choose one representation over another?
- We may learn a model for some population size(s), and want to apply it to other population sizes. We want to make assumptions explicit and know the consequences of these assumptions.
- If one model fits some data, it is important to understand why it fits the data better.

In this paper, we provided answers to these questions for the case of the logistic regression model. The introduction of the relational logistic regression (RLR) family from first principle is already a major contribution. Based on it, we have investigated the dependence on population size for different variants and have demonstrated that already for simple and well-understood (at the non-relational level) models, there are complex interactions of the parameters with population size. Future work includes inference and learning for these models and understanding the relationship to other models such as undirected models like MLNs. Exploring these directions is important since determining which models to use is more than fitting the models to data; we need to understand what we are representing.

References

Buchman, D.; Schmidt, M.; Mohamed, S.; Poole, D.; and Freitas, N. D. 2012. On sparse, spectral and other parameterizations of binary probabilistic models. In *AISTATS 2012-15th International Conference on Artificial Intelligence and Statistics*.

Domingos, P.; Kok, S.; Lowd, D.; Poon, H.; Richardson, M.; and Singla, P. 2008. Markov logic. In Raedt, L. D.; Frasconi, P.; Kersting, K.; and Muggleton, S., eds., *Probabilistic Inductive Logic Programming*. New York: Springer. 92–117.

Friedman, N.; Getoor, L.; Koller, D.; and Pfeffer, A. 1999. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, volume 99, 1300–1309. Stockholm, Sweden: Morgan Kaufman.

Getoor, L., and Taskar, B. 2007. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.

Horsch, M., and Poole, D. 1990. A dynamic approach to probability inference using bayesian networks. In *Proc. sixth Conference on Uncertainty in AI*, 155–161.

Jian, D.; Barthels, A.; and Beetz, M. 2009. Adaptive Markov logic networks: Learning statistical relational models with dynamic parameters. In *9th European Conference on Artificial Intelligence (ECAI)*, 937–942.

Jian, D.; Bernhard, K.; and Beetz, M. 2007. Extending Markov logic to model probability distributions in relational domains. In *KI*, 129–143.

Kisynski, J., and Poole, D. 2009. Lifted aggregation in directed first-order probabilistic models. In *Twenty-first International Joint Conference on Artificial Intelligence*, 1922–1929.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA.

Mitchell, T. 2010. Generative and discriminative classifiers: naive Bayes and logistic regression. http://www.cs.cmu.edu/ tom/mlbook/NBayesLogReg.pdf.

Natarajan, S.; Khot, T.; Lowd, D.; Tadepalli, P.; and Kersting, K. 2010. Exploiting causal independence in Markov logic networks: Combining undirected and directed models. In *European Conference on Machine Learning (ECML)*.

Neville, J.; Simsek, O.; Jensen, D.; Komoroske, J.; Palmer, K.; and Goldberg, H. 2005. Using relational knowledge discovery to prevent securities fraud. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. MIT Press.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufman.

Perlish, C., and Provost, F. 2006. Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning* 62:65–105.

Poole, D.; Buchman, D.; Natarajan, S.; and Kersting, K. 2012. Aggregation and population growth: The relational logistic regression and Markov logic cases. In *Proc. UAI-2012 Workshop on Statistical Relational AI*.

Poole, D. 2003. First-order probabilistic inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, 985–991.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Probabilistic Sentential Decision Diagrams

Doga Kisa and **Guy Van den Broeck** and **Arthur Choi** and **Adnan Darwiche**
Computer Science Department
University of California, Los Angeles
{*doga,guyvdb,aychoi,darwiche*}*@cs.ucla.edu*

Abstract

We propose the Probabilistic Sentential Decision Diagram (PSDD): A complete and canonical representation of probability distributions defined over the models of a given propositional theory. Each parameter of a PSDD can be viewed as the (conditional) probability of making a decision in a corresponding Sentential Decision Diagram (SDD). The SDD itself is a recently proposed complete and canonical representation of propositional theories. We explore a number of interesting properties of PSDDs, including the independencies that underlie them. We show that the PSDD is a tractable representation. We further show how the parameters of a PSDD can be efficiently estimated, in closed form, from complete data. We empirically evaluate the quality of PSDDs learned from data, when we have knowledge, a priori, of the domain logical constraints.

Introduction

The interplay between logic and probability has been of great interest throughout the history of AI. One of the earliest proposals in this direction is Nilsson's (1986) probabilistic logic, which aimed at augmenting first-order logic with probabilities. This has prompted similar approaches, including, for example, Halpern (1990). The focus of these approaches, however, was mainly semantical, yielding no effective schemes for realizing them computationally. More recently, the area of lifted probabilistic inference has tackled this interplay, while employing a different compromise (Poole 2003). In these efforts, the focus has been mostly on restricted forms of first-order logic (e.g., function-free and finite domain), but with the added advantage of efficient inference (e.g., algorithms whose complexity is polynomial in the domain size).

On the propositional side, the thrust of the interplay has been largely computational. An influential development in this direction has been the realization that enforcing certain properties on propositional representations, such as decomposability and determinism, provides one with the power to answer probabilistic queries efficiently. This development was actually based on two technical observations. First, that decomposable and deterministic representations allow one to perform weighted model counting efficiently. Second, that probabilistic reasoning can be reduced to weighted model counting. This development, which has its first roots in Darwiche (2002), has been underlying an increasing number of probabilistic reasoning systems in the last decade. This is especially true for representations that employ both logical and probabilistic elements (e.g., Chavira, Darwiche, and Jaeger (2006) and Fierens et al. (2011)). Moreover, the technique has been extended recently to certain first-order representations as well (Van den Broeck et al. 2011).

This paper is concerned with an orthogonal contribution to this interplay between propositional logic and probability theory. The problem we tackle here is that of developing a representation of probability distributions in the presence of massive, logical constraints. That is, given a propositional logic theory which represents domain constraints, our goal is to develop a representation that induces a unique probability distribution over the models of the given theory. Moreover, the proposed representation should satisfy requirements that are sometimes viewed as necessary for the practical employment of such representations. These include a clear semantics of the representation parameters; an ability to reason with the representation efficiently; and an ability to learn its parameters from data, also efficiently.

Our proposal is called a *Probabilistic Sentential Decision Diagram (PSDD).* It is based on the recently proposed *Sentential Decision Diagram (SDD)* for representing propositional theories (Darwiche 2011; Xue, Choi, and Darwiche 2012; Choi and Darwiche 2013). While the SDD is comprised of *logical* decision nodes, the PSDD is comprised of *probabilistic* decision nodes, which are induced by supplying a distribution over the branches of a logical decision node. Similar to SDDs, the PSDD is a canonical representation, but under somewhat more interesting conditions. Moreover, computing the probability of a term can be done in time linear in the PSDD size. In fact, the probability of each and every literal can be computed in only two passes over the PSDD. It is particularly notable that the local parameters of a PSDD have clear semantics with respect to the global distribution induced by the PSDD. We will also show that these parameters can be learned efficiently from complete data.

This paper is structured as follows. We start by a concrete discussion on some of the applications that have driven the development of PSDDs and follow by an intuitive expo-

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

L	K	P	A	Students
0	0	1	0	6
0	0	1	1	54
0	1	1	1	10
1	0	0	0	5
1	0	1	0	1
1	0	1	1	0
1	1	0	0	13
1	1	1	0	8
1	1	1	1	3

Table 1: Student enrollment data.

sure of PSDDs and their salient features. We next provide a formal treatment of the syntax, semantics and properties of PSDDs. This allows us to present the main inference algorithm for PSDDs and the one for learning PSDD parameters from complete data. The paper concludes with some experimental results showing the promise of PSDDs in learning probability distributions under logical constraints. Proofs of theorems are delegated to the full version of the paper due to space limitations.

Motivation

PSDDs were inspired by the need to learn probability distributions that are subject to domain constraints. Take for example a computer science department that organizes four courses: Logic (L), Knowledge Representation (K), Probability (P), and Artificial Intelligence (A). Students are asked to enroll for these courses under the following restrictions:

– A student must take at least one of Probability or Logic.

– Probability is a prerequisite for AI.

– The prerequisite for KR is either AI or Logic.

The department may have data on student enrollments, as in Table 1, and may wish to learn a probabilistic model for reasoning about student preferences. For example, the department may need to know whether students are more likely to satisfy the prerequisite of KR using AI or using Logic.

A mainstream approach for addressing this problem is to learn a probabilistic graphical model, such as a Bayesian network. In this case, a network structure is constructed manually or learned from data. The structure is then turned into a Bayesian network by learning its parameters from the data. Other graphical models can also be used. This includes, for example, Markov networks or their variations.

What is common among all these approaches is that they lack a principled and effective method for accommodating the domain constraints into the learning process—that is, ensuring, for example, that a student with a profile $A \wedge K \wedge L \wedge \neg P$, or a profile $\neg A \wedge K \wedge \neg L \wedge P$, has zero probability in the learned model. In principle, the zero parameters of a graphical model can capture logical constraints, although a fixed model structure will not in general accommodate arbitrary logical constraints. We could introduce additional structure into the model to capture such constraints, using, e.g., the method of virtual evidence (Pearl 1988; Mateescu and Dechter 2008). However, incorporating constraints in this manner will in general lead to a highly-connected network, making inference intractable. Even if inference remained tractable, such an approach is not ideal as we now have to learn a distribution that is conditioned on the constraints. This would require new learning algorithms (e.g., gradient methods) for performing parameter estimation as traditional methods may no longer be applicable. For example, in Bayesian networks, the closed-form parameter estimation algorithm under complete data will no longer be valid in this case.

The domain constraints of our example can be expressed using the following CNF.

$$\begin{aligned} P \vee L \\ A \Rightarrow P \\ K \Rightarrow A \vee L \end{aligned} \tag{1}$$

Even though there are 16 combinations of courses, the CNF says that only 9 of them are valid choices. An approach that observes this information must learn a probability distribution that assigns a zero probability to every combination that is not allowed by these constraints.

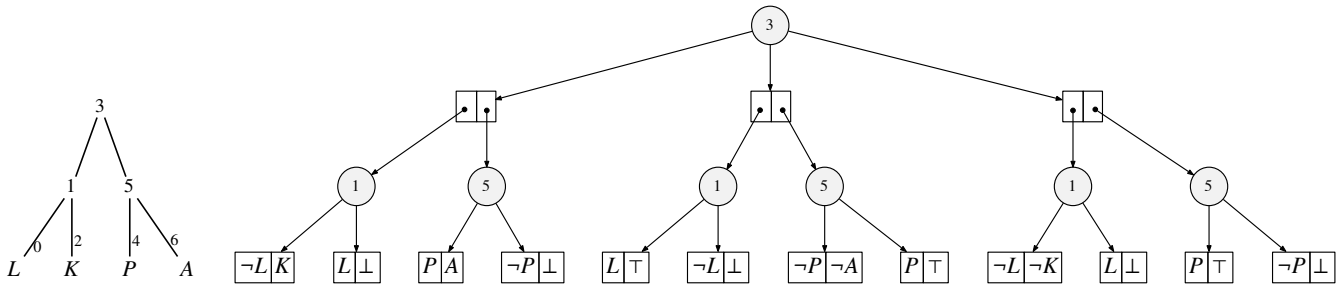
None of the standard learning approaches we are familiar with has been posed to address this problem. The complication here is not strictly with the learning approaches, but with the probabilistic models that are amenable to being learned under these circumstances. In particular, these models are not meant to induce probability distributions that respect a given set of logical constraints.

The simple problem we posed in this section is exemplary of many real-world applications. We mention in particular *configuration* problems that arise when purchasing products, such as cars and computers. These applications give users the option to configure products, but subject to certain constraints. Data is abundant for these applications and there is a clear economic interest in learning probabilistic models under the given constraints. We also mention *reasoning about physical systems,* which includes verification and diagnosis applications. Here, propositional logic is typically used to encode some system functionality, while leaving out some system behaviors which may have a non-deterministic nature (e.g., component failures and probabilistic transitions). There is also an interest here to learn probabilistic models of these systems, subject to the given constraints.

Our goal in this paper is to introduce the PSDD representation for addressing this particular need. We will start by an intuitive (and somewhat informal) introduction to PSDDs, followed by a more formal treatment of their syntax, semantics and the associated reasoning and learning algorithms.

PSDDs

We will refer to domain constraints as the *base* of a probability distribution. Our proposed approach starts by representing this base as a *Sentential Decision Diagram (SDD)* as in Figure 1 (Darwiche 2011; Xue, Choi, and Darwiche 2012; Choi and Darwiche 2013). An SDD is determined by a *vtree,* which is a full binary tree with leaves corresponding to the domain variables (Pipatsrisawat and Darwiche 2008). The choice of a particular SDD can then be thought of as a choice of a particular vtree. We will later discuss the impact of this



(a) Vtree (b) SDD

Figure 1: A vtree and SDD for the student enrollment problem. Numbers in circles correspond to vtree nodes.

choice on the represented distribution. For now, however, we will develop some further understanding of SDDs as they are the backbones of our probability distributions.

SDDs. An SDD is either a *decision node* or a *terminal node.* A terminal node is a literal, the constant $\top$ (true) or the constant $\bot$ (false). A decision node is a disjunction of the form $(p_1 \wedge s_1) \vee \ldots \vee (p_n \wedge s_n)$, where each pair (p_i, s_i) is called an *element.* A decision node is depicted by a circle and its elements are depicted by paired boxes. Here, $p_1, \ldots, p_n$ are called *primes* and $s_1, \ldots, s_n$ are called *subs.* Primes and subs are themselves SDDs. Moreover, the primes of a decision node are always consistent, mutually exclusive and exhaustive. The SDD in Figure 1 has seven decision nodes. The decision node to the far left has two elements $(\neg L, K)$ and $(L, \bot)$. It represents $(\neg L \wedge K) \vee (L \wedge \bot)$, which is equivalent to $\neg L \wedge K$. There are two primes for this node $\neg L$ and L. The two corresponding subs are K and $\bot$.

Structure. An SDD can be viewed as a *structure* that induces infinitely many probability distributions (all having the same base). By *parameterizing* an SDD, one obtains a PSDD that induces a particular probability distribution.

Parameters. Figure 2 depicts a PSDD which is obtained by parameterizing the SDD in Figure 1. Both decision and terminal SDD nodes are parameterized, but we focus here on decision nodes. Let n be a decision node having elements $(p_1, s_1), \ldots, (p_n, s_n)$. To parameterize node n is to provide a distribution $\theta_1, \ldots, \theta_n$. Intuitively, θ_i is the probability of prime p_i given that the decision of node n has been implied. We will formalize and prove this semantics later. We will also provide an efficient procedure for learning the parameters of a PSDD from complete data. The PSDD parameters in Figure 2 were learned using this procedure from the data in Table 2. The table also depicts the probability distribution induced by the learned PSDD.

Independence. The PSDD structure is analogous to a Bayesian network structure in the following sense. The latter can be parameterized in infinitely many ways, with each parameterization inducing a particular probability distribution. Moreover, all the induced distributions satisfy certain independences that can be inferred from the underlying Bayesian network structure. The same is true for PSDDs. Each parameterization of a PSDD structure yields a unique probability distribution. Moreover, all the induced distributions satisfy independences that can be inferred from the PSDD structure.

L	K	P	A	Students	Learned PSDD Distribution	
0	0	1	0	6	$0.6 \cdot 0.1$	6.0%
0	0	1	1	54	$0.6 \cdot 0.9$	54.0%
0	1	1	1	10	0.1	10.0%
1	0	0	0	5	$0.3 \cdot 0.2 \cdot 0.6$	3.6%
1	0	1	0	1	$0.3 \cdot 0.2 \cdot 0.4 \cdot 0.75$	1.8%
1	0	1	1	0	$0.3 \cdot 0.2 \cdot 0.4 \cdot 0.25$	0.6%
1	1	0	0	13	$0.3 \cdot 0.8 \cdot 0.6$	14.4%
1	1	1	0	8	$0.3 \cdot 0.8 \cdot 0.4 \cdot 0.75$	7.2%
1	1	1	1	3	$0.3 \cdot 0.8 \cdot 0.4 \cdot 0.25$	2.4%

Table 2: Student enrollment data and learned distribution.

We will show, however, that PSDD independence is more refined than Bayesian network independence as it allows one to express more qualified independence statements.

The Syntax and Semantics of PSDDs

PSDDs are based on *normalized* SDDs in which every node n is associated with (normalized for) a vtree node v according to the following rules (Darwiche 2011).

- If n is a terminal node, then v is a leaf node which contains the variable of n (if any).
- If n is a decision node, then its primes (subs) are normalized for the left (right) child of v.
- If n is the root SDD node, then v is the root vtree node.

The SDD in Figure 1 is normalized. Each decision node in this SDD is labeled with the vtree node it is normalized for. We are now ready to define the syntax of a PSDD.

Definition 1 (PSDD Syntax) *A PSDD is a normalized SDD with the following parameters.*

- *For each decision node $(p_1, s_1), \ldots, (p_k, s_k)$ and prime p_i, a positive parameter θ_i is supplied such that $\theta_1 + \ldots + \theta_k = 1$ and $\theta_i = 0$ iff $s_i = \bot$.*
- *For each terminal node $\top$, a positive parameter θ is supplied such that $0 < \theta < 1$.*

A terminal node $\top$ with parameter θ will be denoted by $X : \theta$, where X is the variable of leaf vtree node that $\top$ is normalized for. Other terminal nodes (i.e., $\bot$, X and $\neg X$) have fixed, implicit parameters (discussed later) and will be denoted as is. A decision PSDD node will be denoted by

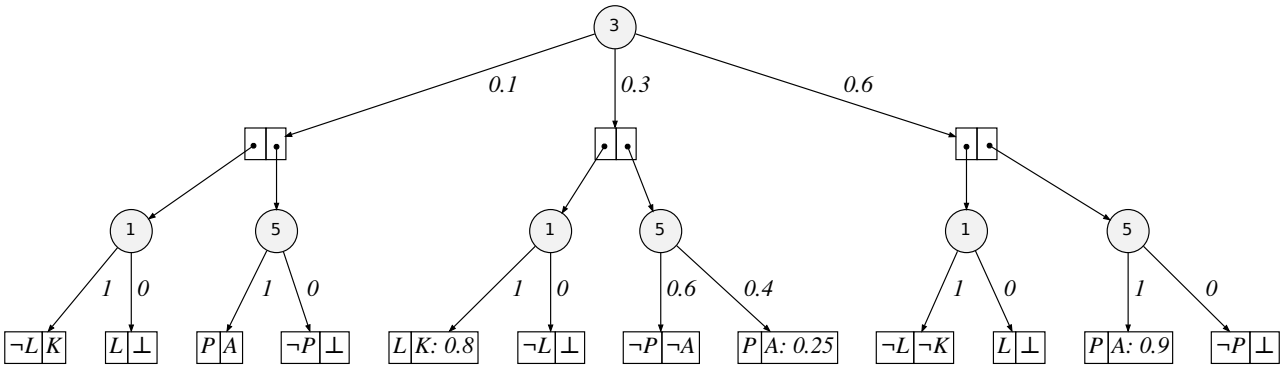


Figure 2: A PSDD for the student enrollment problem, which results from parameterizing the SDD in Figure 1. The parameters were learned from the dataset in Table 1 (also shown in Table 2).

$(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$. Graphically, we will just annotate the edge into element (p_i, s_i) with the parameter θ_i. Figure 2 provides examples of this notation.

We next define the distribution of a PSDD, inductively. That is, we first define the distribution induced by a terminal node. We then define the distribution of a decision node in terms of the distributions induced by its primes and subs.

Definition 2 (PSDD Distribution) *Let n be a PSDD node that is normalized for vtree node v. Node n defines a distribution Pr_n over the variables of vtree v as follows.*

– *If n is a terminal node, and v has variable X, then*

n	$\mathsf{Pr}_n(X)$	$\mathsf{Pr}_n(\neg X)$
$X : \theta$	θ	$1-\theta$
$\bot$	0	0
X	1	0
$\neg X$	0	1

– *If n is a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$ and v has left variables $\mathbf{X}$ and right variables $\mathbf{Y}$, then*

$$\mathsf{Pr}_n(\mathbf{xy}) \stackrel{def}{=} \mathsf{Pr}_{p_i}(\mathbf{x}) \cdot \mathsf{Pr}_{s_i}(\mathbf{y}) \cdot \theta_i \quad \textit{for } i \textit{ where } \mathbf{x} \models p_i.$$

Applying this definition to the PSDD of Figure 2 leads to the distribution in Table 2 for its root node. The following table depicts the distribution induced by a non-root node in this PSDD, which appears in the middle of Figure 2.

$\mathbf{x}$	$\mathbf{y}$	$\mathsf{Pr}_{p_i}(\mathbf{x})$	$\mathsf{Pr}_{s_i}(\mathbf{y})$	θ_i	$\mathsf{Pr}_n(\mathbf{xy})$
P	A	1	0.25	0.4	0.1
P	$\neg A$	1	0.75	0.4	0.3
$\neg P$	A	1	0	0.6	0.0
$\neg P$	$\neg A$	1	1	0.6	0.6

Table 3: Distribution of node $n = (\neg P, \neg A)(P, \top)$.

The SDD node associated with a PSDD node n is called the *base* of n and is denoted by $[n]$. When there is no ambiguity, we will often not distinguish between a PSDD node n and its base $[n]$.

A PSDD assigns a strictly positive probability to a variable instantiation iff the instantiation satisfies its base. This can be seen, for example, in Table 3. This is also the first key property of PSDDs.

Theorem 1 (Base) *Consider a PSDD node n that is normalized for vtree node v. If $\mathbf{Z}$ are the variables of vtree v, then $\mathsf{Pr}_n(\mathbf{z}) > 0$ iff $\mathbf{z} \models [n]$.*

We will now discuss the second key property of PSDDs, which reveals the *local* semantics of PSDD parameters.

Theorem 2 (Parameter Semantics) *Let n be a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$. We have $\theta_i = \mathsf{Pr}_n([p_i])$.*

Consider the PSDD in Figure 2 and its decision node n in Table 3. Prime $\neg P$ of this node has parameter 0.6. According to Theorem 2, we must then have $\mathsf{Pr}_n(\neg P) = 0.6$, which can be verified in Table 3. Similarly, $\mathsf{Pr}_n(P) = 0.4$.

The third key property of PSDDs is the relationship between the local distributions induced by its various nodes (node distributions) and the global distribution induced by its root node (PSDD distribution)—for example, the relationship between the distribution of node n in Table 3 and the PSDD distribution given in Table 2.

Node distributions are linked to the PSDD distribution by the notion of *context.*

Definition 3 (Context) *Let $(p_1, s_1), \ldots, (p_k, s_k)$ be the elements appearing on some path from the SDD root to node n.*[1] *Then $p_1 \wedge \ldots \wedge p_k$ is called a sub-context for node n and is feasible iff $s_i \neq \bot$. The context is a disjunction of all sub-contexts and is feasible iff some sub-context is feasible.*

Consider Figure 1. The three decision nodes normalized for vtree node $v = 5$ have the contexts $\neg L \wedge K$, L and $\neg L \wedge \neg K$. Moreover, the terminal nodes normalized for vtree $v = 6$ have the contexts:

– A: $\neg L \wedge K \wedge P$

– $\neg A$: $L \wedge \neg P$

– $\bot$: $(\neg L \wedge K \wedge \neg P) \vee (\neg L \wedge \neg K \wedge \neg P) = (\neg L \wedge \neg P)$

– $\top$: $(L \wedge P) \vee (\neg L \wedge \neg K \wedge P) = (L \vee \neg K) \wedge P$.

Contexts satisfy interesting properties.

[1] That is, $n = p_k$ or $n = s_k$.

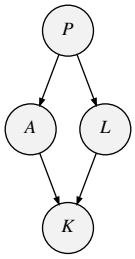


Figure 3: A Bayesian network structure.

Theorem 3 (Context) *A node is implied by its context and the underlying SDD. Nodes normalized for the same vtree node have mutually exclusive and exhaustive contexts. The sub-contexts of a node are mutually exclusive. A context/sub-context is feasible iff it has a strictly positive probability.*

Contexts give a global interpretation to node distributions.

Theorem 4 (Node Distribution) *Consider a PSDD r and let n be one of its nodes. If γ_n is a feasible sub-context or feasible context of node n, then $\mathsf{Pr}_n(.) = \mathsf{Pr}_r(. \mid \gamma_n)$.*

Contexts also give a *global* interpretation to parameters.

Corollary 1 (Parameter Semantics) *Consider a PSDD r and node n with feasible sub-context or feasible context γ_n.*

- *If n is a terminal node $X:\theta$, then $\theta = \mathsf{Pr}_r(X \mid \gamma_n)$.*
- *If n is a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$, then $\theta_i = \mathsf{Pr}_r([p_i] \mid \gamma_n)$ for $i = 1, \ldots, k$.*

This corollary says that the parameters of a node are conditional probabilities of the PSDD distribution.

We show in the Appendix that PSDDs are *complete* as they are capable of representing any probability distribution. We also show that PSDDs are *canonical* under a condition known as compression. More precisely, we show that there is a unique compressed PSDD for each distribution and vtree. This is particularly important for learning PSDDs (structure and parameters) as it reduces the problem of searching for a PSDD into the problem of searching for a vtree.

PSDD Independence

Consider the Bayesian network structure in Figure 3, which corresponds to our earlier example. This structure encodes a number of probabilistic independences that hold in any distribution it induces (i.e., regardless of its parameters). These independences are

- A and L are independent given P.
- K and P are independent given AL.

These independences are conditioned on variables. That is, "given AL" reads "given any state of variables A and L." The second independence can therefore be expanded into $2 \times 2 \times 4$ statements of the form "α is independent of β given γ," where α, β and γ are propositional sentences (e.g., $\neg K$ is independent of P given $A \wedge \neg L$).

The structure of a PSDD also encodes independences that hold in every induced distribution. These independences fall into two major categories, the first coming from Theorem 4.

Corollary 2 (Independence I) *Consider a PSDD r and a node n with context γ_n and feasible sub-context β_n.*

- *If n is a terminal node $X:\theta$, then*

$$\mathsf{Pr}_r(X \mid \gamma_n, \beta_n) = \mathsf{Pr}_r(X \mid \gamma_n) = \mathsf{Pr}_r(X \mid \beta_n) = \theta.$$

- *If n is a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$, then*

$$\mathsf{Pr}_r([p_i] \mid \gamma_n, \beta_n) = \mathsf{Pr}_r([p_i] \mid \gamma_n) = \mathsf{Pr}_r([p_i] \mid \beta_n) = \theta_i$$

for $i = 1, \ldots, k$.

That is, the probability of a prime is independent of a sub-context once the context is known. This is also true for the probability of a terminal sub. Moreover, which specific sub-context we know is irrelevant. All are equivalent as far as defining the semantics of parameters is concerned.

The second category of independences is as follows.

Theorem 5 (Independence II) *Let γ_v be a feasible context for a PSDD node normalized for vtree node v. Variables inside v are independent of those outside v given context γ_v.*

To read the independences characterized by this theorem, one iterates over each vtree node v, identifying its corresponding, feasible contexts γ_v. Consider the PSDD in Figure 2. There are three decision nodes which are normalized for vtree node $v = 5$, with contexts $\neg L \wedge K$, L and $\neg L \wedge \neg K$. Using the second context, we get

variables PA and LK are independent given context L.

This reads as "If we know that someone took Logic, then whether they took KR has no bearing on whether they took Probability or AI."

PSDD independences are conditioned on propositional sentences (contexts) instead of variables. This kind of independence is known to be more expressive and is usually called *context-specific independence* (Boutilier et al. 1996). This kind of independence is beyond the scope of probabilistic graphical models, which can only condition independence statements on variables. Consider the example statement we discussed above. If we were to condition on the variable L instead of the propositional sentence L, we would also get "If we know that someone *did not take* Logic, then whether they took KR has no bearing on whether they took Probability or AI." This is actually contradicted by the logical constraints for this problem ($K \Rightarrow A \vee L$). If someone did not take Logic, but took KR, they must have taken AI.

Reasoning with PSDDs

We now present the main algorithms for reasoning with PSDDs. In particular, given a PSDD r and an instantiation $\mathbf{e}$ of some variables (*evidence*), we provide an algorithm for computing the probability of this evidence $\mathsf{Pr}_r(\mathbf{e})$. We also present an algorithm for computing the conditional probability $\mathsf{Pr}_r(X \mid \mathbf{e})$ for every variable X. Both algorithms run in time which is linear in the PSDD size.

We start with the first algorithm. For variable instantiation $\mathbf{e}$ and vtree node v, we will use $\mathbf{e}_v$ to denote the subset of instantiation $\mathbf{e}$ that pertains to the variables of vtree v, and $\mathbf{e}_{\bar{v}}$ to denote the subset of $\mathbf{e}$ that pertains to variables outside v. The first algorithm is based on the following result.

Algorithm 1: Probability of Evidence

Input: PSDD r and evidence $\mathbf{e}$

$evd(n) \leftarrow 0$ for every node n
// visit children before parents
foreach *node n in the PSDD* **do**
  if *n is a terminal node* **then**
    $v \leftarrow$ leaf vtree node that n is normalized for
    $evd(n) \leftarrow \mathsf{Pr}_n(\mathbf{e}_v)$
  else
    foreach *element (p_i, s_i, θ_i) of node n* **do**
      $evd(n) \leftarrow evd(n) + evd(p_i) \cdot evd(s_i) \cdot \theta_i$

Theorem 6 *Consider a decision node $n = (p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$ that is normalized for vtree node v, with left child l and right child r. For evidence $\mathbf{e}$, we have*

$$\mathsf{Pr}_n(\mathbf{e}_v) = \sum_{i=1}^{k} \mathsf{Pr}_{p_i}(\mathbf{e}_l) \cdot \mathsf{Pr}_{s_i}(\mathbf{e}_r) \cdot \theta_i$$

When n is a terminal node, v is a leaf vtree node and $\mathbf{e}_v$ is either a literal or the empty instantiation. In this case, we can just look up the value of $\mathsf{Pr}_n(\mathbf{e}_v)$ from the distribution induced by the terminal node n (from Definition 2).

Theorem 6 leads to Algorithm 1, which traverses the PSDD bottom-up, computing $\mathsf{Pr}_n(\mathbf{e}_v)$ for each node n and storing the result in $evd(n)$. The probability of evidence is then $evd(r)$, where r is the PSDD root.

We now turn to computing the probability $\mathsf{Pr}_r(X, \mathbf{e})$ for each variable X. One can use Algorithm 1 to perform this computation, but the algorithm would need to be called once for each variable X. However, with the following theorem, we can compute all of these node marginals using a single, second pass on the PSDD, assuming that Algorithm 1 did the first pass.

Theorem 7 *Consider a PSDD r, variable X, and its leaf vtree node v. Let $n_1, \ldots, n_k$ be all the terminal nodes normalized for v and let $\gamma_{n_1}, \ldots, \gamma_{n_k}$ be their corresponding contexts. For evidence $\mathbf{e}$, we have*

$$\mathsf{Pr}_r(X, \mathbf{e}_{\bar{v}}) = \sum_{i=1}^{k} \mathsf{Pr}_{n_i}(X) \cdot \mathsf{Pr}_r(\gamma_{n_i}, \mathbf{e}_{\bar{v}}).$$

If $\mathbf{e} \models \neg X$, then $\mathsf{Pr}_r(X, \mathbf{e}) = 0$. Otherwise, $X, \mathbf{e}_{\bar{v}} = X, \mathbf{e}$ and $\mathsf{Pr}_r(X, \mathbf{e}) = \mathsf{Pr}_r(X, \mathbf{e}_{\bar{v}})$.

The term $\mathsf{Pr}_{n_i}(X)$ in Theorem 7 is immediately available. Algorithm 2 computes $\mathsf{Pr}_r(\gamma_n, \mathbf{e}_{\bar{v}})$ for every PSDD node n that has context γ_n and is normalized for vtree node v. The algorithm traverses the PSDD top-down, computing this probability for each visited node n, storing it in $ctx(n)$. If n is a terminal node, the algorithm also computes $\mathsf{Pr}_r(X, \mathbf{e}_{\bar{v}})$ and $\mathsf{Pr}_r(\neg X, \mathbf{e}_{\bar{v}})$, storing them in $mrg(X)$ and $mrg(\neg X)$.

The simplicity of Algorithm 2 is due to the following. The probability of a sub-context can be computed by multiplying the parameters appearing on its corresponding path. Since sub-contexts are mutually exclusive, their probabilities can be added to obtain the context probability. Algorithm 2 does precisely this except that it accounts for evidence as well using quantities computed by Algorithm 1.

Algorithm 2: Probability of Contexts

Input: PSDD r

$ctx(n) \leftarrow 0$ for nodes $n \neq r$ and $ctx(r) \leftarrow 1$
$mrg(X) \leftarrow 0$ and $mrg(\neg X) \leftarrow 0$ for every variable X
// visit parents before children
foreach *node n in the PSDD* **do**
  if *n is a terminal node* **then**
    $X \leftarrow$ variable of node n
    $mrg(X) \leftarrow mrg(X) + ctx(n) \cdot \mathsf{Pr}_n(X)$
    $mrg(\neg X) \leftarrow mrg(\neg X) + ctx(n) \cdot \mathsf{Pr}_n(\neg X)$
  else
    foreach *element (p_i, s_i, θ_i) of node n* **do**
      $ctx(p_i) \leftarrow ctx(p_i) + \theta_i \cdot evd(s_i) \cdot ctx(n)$
      $ctx(s_i) \leftarrow ctx(s_i) + \theta_i \cdot evd(p_i) \cdot ctx(n)$

Learning with PSDDs

We now present an algorithm for learning the parameters of a PSDD from a *complete* dataset. We start first with some basic definitions. An instantiation of all variables is called an *example*. There are 2^n distinct examples over n propositional variables. A complete dataset is a multi-set of examples.[2] That is, an example may appear multiple times in a dataset. Given a PSDD structure (a normalized SDD), and a complete dataset, our goal is to learn the value of each PSDD parameter. More precisely, we wish to learn *maximum likelihood* parameters: ones that maximize the probability of examples in the dataset.

We will use Pr_θ to denote the distribution induced by the PSDD structure and parameters θ. The *likelihood* of these parameters given dataset $\mathbf{D}$ is defined as

$$L(\theta|\mathbf{D}) = \prod_{i=1}^{N} \mathsf{Pr}_\theta(\mathbf{d}_i),$$

where $\mathbf{d}_i$ ranges over all N examples in dataset $\mathbf{D}$. Our goal is then to find the maximum likelihood parameters

$$\theta^{ml} = \operatorname*{argmax}_\theta L(\theta|\mathbf{D}).$$

We will use $\mathbf{D}\#(\alpha)$ to denote the number of examples in dataset $\mathbf{D}$ that satisfy propositional sentence α. For a decision node $n = (p_1, s_1, \theta_1), ..., (p_k, s_k, \theta_k)$ with context γ_n, we propose the following estimate for parameter θ_i:

$$\theta_i^{ml} = \frac{\mathbf{D}\#(p_i, \gamma_n)}{\mathbf{D}\#(\gamma_n)}. \quad (2)$$

For terminal node $n = X{:}\theta$ with context γ_n, we propose the following estimate for parameter θ:

$$\theta^{ml} = \frac{\mathbf{D}\#(X, \gamma_n)}{\mathbf{D}\#(\gamma_n)}. \quad (3)$$

[2] In an *incomplete* dataset, an example corresponds to an instantiation of some variables (not necessarily all).

We can now show the following.

Theorem 8 *The parameter estimates of Equations 2 and 3 are the only estimates that maximize the likelihood function.*

Our parameter estimates admit a *closed-form*, in terms of the counts $\mathbf{D}\#(\alpha)$ in the data. One can compute these estimates using a single pass through the examples of a dataset. Moreover, each distinct example can be processed in time linear in the PSDD size.[3] These are very desirable properties for a parameter learning algorithm. These properties are shared with Bayesian network representations, but are missing from many others, including Markov networks.

When learning probabilistic graphical models, one makes a key distinction between learning structures versus learning parameters (the former being harder in general). While learning PSDD structures is beyond the scope of this paper, the experimental results we present next do use a basic method for learning structures. In particular, since we compile logical constraints into an SDD (i.e., a PSDD structure), the compilation technique we use is effectively "learning" a structure. We used the publicly available SDD package for this purpose (*http://reasoning.cs.ucla.edu/sdd/*). The SDD package tries to dynamically minimize the size of the compiled SDD and, as a result, tries to minimize the number of PSDD parameters.

Preliminary Experimental Results

In this Section, we empirically evaluate our parameter estimation algorithm for PSDDs. First, we illustrate how our algorithm can effectively recover the parameters of a PSDD from data, assuming that the dataset was indeed generated by the distribution of a PSDD. Second, we highlight how knowing the logical constraints underlying a given dataset can impact the accuracy of learned models.

In our first set of experiments, using synthetic data, we show how we can recover the true parameters that generated a dataset, given that the logical constraints are known. As we will be simulating datasets from a known PSDD, we shall use the KL-divergence to compare the original PSDD distribution $\Pr(\mathbf{Z})$, and the PSDD distribution $\Pr'(\mathbf{Z})$ that we learned from the data:

$$KL(\Pr, \Pr') = \sum_{\mathbf{z}} \Pr(\mathbf{z}) \log \frac{\Pr(\mathbf{z})}{\Pr'(\mathbf{z})}.$$

Note that the KL-divergence is non-negative, and zero iff the two distributions are equivalent.[4]

[3] A dataset may not include every example that is consistent with the domain constraints. If this is the case, a parameter θ for prime p may be estimated to zero, even though its sub s may not be $\bot$; see Theorem 1. To address this, one can assume a pseudo-count for each distinct example, which can be thought of as providing a prior distribution on parameters. In our experiments, we assumed a pseudo-count of $1/\mathsf{mc}$ for each distinct example, where mc is the model count of the SDD. This corresponds to a very weak prior since, in aggregate, these pseudo-counts contribute a total count that is equivalent to one real example in the dataset.

[4] The KL–divergence between two PSDDs can be computed efficiently if they share the same structure.

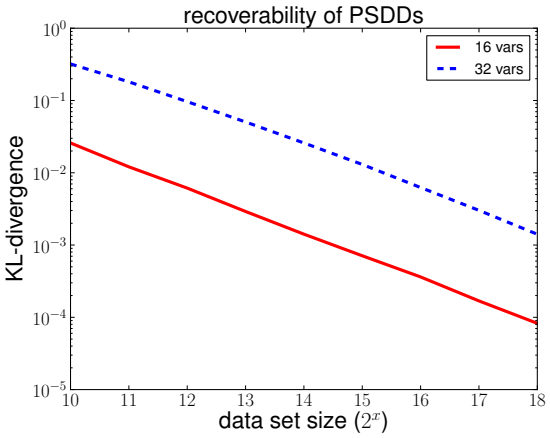


Figure 4: We observe how our parameter estimation algorithm can recover the original PSDD as we increase the size of the training dataset (x-axis), measured by the KL-divergence (y-axis, where lower is better).

Consider Figure 4, where each point in the plot represents an average of 100 randomly generated PSDDs. We generated 10 random PSDD structures by compiling random 3-CNFs into SDDs (the used clause to variable ratio is 3). For each SDD, we randomly generated 10 different parameterizations, giving us $10 \times 10 = 100$ PSDDs total. The SDDs were compiled using dynamic minimization, which reduces the size of the SDD, and hence, the number of parameters in the corresponding PSDD. We evaluated PSDDs over 16 variables and 32 variables (one curve each).

As we go right on the x-axis, we increase the size of the dataset that we are learning our PSDD from. We clearly see that as more data becomes available, the parameters that we learn more closely match the true parameters of the original PSDD (measured using the KL-divergence, on the y-axis). Indeed, both curves are steadily tending to zero. The PSDDs over 16 variables converge faster than those over 32 variables. The PSDDs over 16 variables have fewer parameters, roughly, a tenth as many.

Our final experiment tries to put PSDD learning in the context of learning probabilistic graphical models. It is hard to make a direct comparison here since existing approaches do not factor domain constraints into the learning process. Hence, there are no available systems that tackle the learning problem we posed in this paper, nor are there reported experimental results that are directly relevant. Still, we conceived an experiment that is revealing in this regard, which we explain next.

A standard experiment for evaluating a learning approach goes as follows. The dataset is divided into a *training* portion and *testing* one. The learning algorithm is then given the training dataset and asked to find a probabilistic model. The quality of the model is then measured by computing the likelihood of learned parameters, but using the testing dataset. The literature contains many experiments of this form (e.g., Table 4). The question we posed is this: What if the dataset embedded the domain constraints, in that any example that

Table 4: Log Likelihoods of Testing Datasets. A question mark indicates the unavailability of likelihood for the corresponding approach/dataset.

benchmark	#var	ACMN	ACBN	LTM	CLT	PSDD
NLTCS	16	-6.01	-6.02	-6.49	-6.76	-6.31
MSNBC	17	-6.04	-6.04	-6.52	-6.54	-6.20
KDDCup	64	-2.15	-2.16	-2.18	-2.29	-2.11
Plants	69	-12.89	-12.85	-16.39	-16.52	-10.99
Audio	100	-40.32	-41.13	-41.90	-44.37	-19.34
Jester	100	-53.35	-54.43	-55.17	-58.23	-18.49
Netflix	100	-57.26	-57.75	-58.53	-60.25	-19.41
Accidents	111	?	?	?	-33.19	-19.09
Retail	135	?	?	?	-10.94	-10.68
Pumsb Star	163	?	?	?	-30.79	-18.25
DNA	180	?	?	?	-87.68	-14.75
MSWeb	294	-9.77	-9.81	-10.21	-10.19	-9.25
Book	500	-35.62	-36.02	-34.22	-37.83	-17.38
EachMovie	500	?	?	?	-64.83	-14.89
WebKB	839	-161.30	-159.85	-156.84	-163.43	-15.89
Kosarak	883	?	?	?	-19.87	-9.03
Reuters-52	889	-89.54	-89.27	-91.23	-94.41	-17.58
20 NG	910	-159.56	-159.65	-156.77	-164.13	-18.91
BBC	1058	?	?	?	-261.80	-14.22
Ad	1556	?	?	?	-16.41	-10.31

does not appear in the dataset (training or testing) is an impossible example? Under this assumption, the domain constraints correspond to a sentence which is obtained by disjoining all examples in the dataset. This is indeed what we did in our last experiment. We compiled the mentioned disjunction (which is a DNF) into an SDD, learned its parameters using the training dataset, and measured the quality of learned PSDD using the testing dataset.

We compared the obtained results on the corpus in Table 4; see, e.g., (Davis and Domingos 2010) for more statistics on these datasets. These real-world datasets are commonly used to evaluate algorithms for learning probabilistic graphical models as we just discussed.[5] Here, ACMN are Arithmetic Circuits for Bayesian networks, ACBN are Arithmetic Circuits for Markov networks, LTM are Latent Tree Models, and CLT are Chow-Liu trees. Each algorithm learns its corresponding model using the training dataset, and evaluates it using the testing dataset. The likelihoods of learned models are given in Table 4; these likelihoods can also be found in (Lowd and Rooshenas 2013). For the two datasets over a small number of variables, our PSDDs obtain competitive likelihoods (smaller magnitude is better). In datasets over a larger number of variables, the PSDDs obtain better, and often significantly better, likelihoods than other learned models.

While this experiment is contrary to what is usual, it is both revealing and suggestive. First, it highlights the potential impact of accommodating logical constraints into the learning process. Second, it begs the question: What if the data was indeed indicative of the domain constraints? For example, when the dataset is large enough, and the domain is known to be constrained enough, it is not unreasonable to assume that the dataset is indicative of domain constraints. Moreover, theoretically, this assumption does hold in the limit. As far as we know, there is no existing approach that can exploit this assumption as done by this work. In fact, this observation and the need to address it is what prompted us to develop PSDDs in the first place.

[5]The models that were learned in Table 4 correspond to restricted Bayesian or Markov networks (they have a corresponding arithmetic circuit (AC) or have a tree structure). These restrictions are meant to allow one to compute likelihoods efficiently.

Related Work

The PSDD can be viewed from two angles: As a knowledge representation and reasoning formalism, and as a learning formalism. We next discuss related work across both angles.

As a KR formalism, the PSDD is related to work on extending Binary Decision Diagrams (BDDs) to represent real-valued functions. The relationship between SDDs (which underly PSDDs) and BDDs is known (Darwiche 2011; Xue, Choi, and Darwiche 2012; Choi and Darwiche 2013). In sum though, BDDs branch on literals, instead of sentences, which leads to limitations that are inherited by their extensions (Xue, Choi, and Darwiche 2012). Algebraic Decision Diagrams (ADDs) (R.I. Bahar et al. 1993) in particular are often used to represent joint probability distributions. However, they are not factorized representations and, as such, they need to explicitly represent every distinct probability in a distribution. Affine ADDs (Sanner and McAllester 2005) and Edge-valued Binary Decision Diagrams (Lai and Sastry 1992) do offer a factorized representation, but their parameters are not interpretable as in PSDDs (i.e., as probabilities of the global distribution).

There is also a body of work on facilitating the representation of logical constraints in the context of probabilistic representations. This includes Richardson and Domingos (2006), who facilitate the representation of logical constraints in Markov networks. Additionally, Mateescu and Dechter (2008) separate an unconstrained probabilistic model (e.g., Bayesian network) from the logical constraints and define the target distribution as the former conditioned on the latter. Hence, individual parameters in their framework have no local semantics in the target distribution. Several more first-order probabilistic languages can directly express logical dependencies (De Raedt et al. 2008; Getoor and Taskar 2007). Within these representations, there has been considerable interest in probabilistic reasoning in the presence of logical constraints. See for example Poon and Domingos (2006) and Gogate and Dechter (2007).

Perhaps the most strongly related formalism is the probabilistic decision graph (Bozga and Maler 1999; Jaeger 2004). These circuits are governed by a variable forest instead of a vtree. Their parameters represent contextual conditional probabilities. However, for a fixed variable forest, probabilistic decision graphs are not a complete representation. There are distributions that cannot be represented because the variable forest already encodes certain conditional independencies. Hence, these representations are canonical only in a weaker sense although they have been learned from data (Jaeger, Nielsen, and Silander 2006). As a logical representation, they are situated in between BDDs and SDDs:

They provide stronger decompositions than BDDs but still branch over single variables instead of sentences.

Viewing PSDDs as a learning formalism brings up two potential connections to existing work. First, as a formalism that aims at learning *generative* probabilistic models with well defined bases (i.e., logical constraints), the PSDD is somewhat unique as this learning problem has not been posed this explicitly before. The only possible exception is (Chang et al. 2008), which learns a *discriminative* probabilistic model given logical constraints (they propose a generalization of linear models called Constrained Conditional Models).[6] Second, as a formalism that aims at learning *tractable* representations of distributions, the PSDD falls into the recent body of work on *deep learning*. The closest connection here is to the sum-product network (Poon and Domingos 2011), which is a new (but also tractable) class of deep architectures. These networks represent probability distributions as a (deep) hierarchy of mixtures (essentially, an arithmetic circuit with latent variables). However, they do not take logical constraints as an input. Moreover, even though they provide a less constrained representation of distributions, compared to the PSDD, the implication is that learning becomes harder (e.g., there is no closed form for parameter learning under complete data).

Conclusion

We presented the PSDD as a representation of probability distributions that respect a given propositional theory. The PSDD is a complete and canonical representation, with parameters that are interpretable as conditional probabilities. The PSDD encodes context-specific independences, which can be derived from its structure. The PSDD is a tractable representation, allowing one to compute the probability of any term in time linear in its size. The PSDD has unique maximum likelihood parameters under complete data, which can be learned efficiently using a closed form. Preliminary experimental results suggest that the PSDD can be quite effective in learning distributions under domain constraints.

Acknowledgments

This work has been partially supported by ONR grant #N00014-12-1-0423, NSF grant #IIS-1118122, and the Research Foundation-Flanders (FWO-Vlaanderen).

Appendix

Completeness and Canonicity of PSDDs

We will now address two connected questions with regards to the representational power of PSDDs. The first question asks whether every distribution can be represented by a PSDD. The second question asks whether the PSDD representation is canonical (given a vtree). The answers to both questions rely on the notion of *probabilistic* $\mathbf{XY}$*-partitions,* which are based on the $\mathbf{XY}$-partitions underlying SDDs.

We review the latter notion first. Consider a propositional sentence α over disjoint variables $\mathbf{X}$ and $\mathbf{Y}$. One can always express this sentence as

$$\alpha = (p_1 \wedge s_1) \vee \ldots \vee (p_k \wedge s_k) \tag{4}$$

where $p_1, \ldots, p_k$ mention only variables $\mathbf{X}$ and $s_1, \ldots, s_k$ mention only variables $\mathbf{Y}$. One can also always ensure that each p_i is consistent, that $p_i \wedge p_j$ is inconsistent for $i \neq j$, and that $p_1 \vee \ldots \vee p_k$ is valid. Under these conditions, the form in (4) is called an $\mathbf{XY}$-partition of α (Darwiche 2011). One obtains an SDD for some sentence α by recursively decomposing it into $\mathbf{XY}$-partitions (Darwiche 2011). A parallel notion exists for distributions.

Definition 4 *Let* $\mathsf{Pr}(\mathbf{XY})$ *be a distribution with base* α*. A probabilistic* $\mathbf{XY}$*-partition of this distribution has the form*

$$(\mathsf{Pr}_{p_1}(\mathbf{X}), \mathsf{Pr}_{s_1}(\mathbf{Y}), \theta_1), \ldots, (\mathsf{Pr}_{p_k}(\mathbf{X}), \mathsf{Pr}_{s_k}(\mathbf{Y}), \theta_k)$$

such that

- $(p_1 \wedge s_1) \vee \ldots \vee (p_k \wedge s_k)$ *is an* $\mathbf{XY}$*-partition of base* α*.*
- *Each* Pr_{p_i} *is a (prime) distribution with base* p_i*.*
- *Each* Pr_{s_i} *is a (sub) distribution with base* s_i*.*
- $\theta_i \geq 0$*,* $\theta_i = 0$ *iff* $s_i = \bot$*, and* $\theta_1 + \cdots + \theta_k = 1$*.*
- $\mathsf{Pr}(\mathbf{xy}) = \mathsf{Pr}_{p_i}(\mathbf{x}) \cdot \mathsf{Pr}_{s_i}(\mathbf{y}) \cdot \theta_i$ *where* p_i *satisfies* $\mathbf{x} \models p_i$*.*

The above definition shows how one can decompose a distribution $\mathsf{Pr}(\mathbf{XY})$ into a number of smaller distributions $\mathsf{Pr}(\mathbf{X})$ and $\mathsf{Pr}(\mathbf{Y})$. More importantly though, the bases of these smaller distributions correspond to a decomposition of the base for distribution $\mathsf{Pr}(\mathbf{XY})$.

One can always decompose a distribution this way.

Theorem 9 *Let* Pr *be a probability distribution and suppose that its variables are partitioned into* $\mathbf{X}$ *and* $\mathbf{Y}$*. There must exist a probabilistic* $\mathbf{XY}$*-partition of distribution* Pr*.*

Corollary 3 *Every probability distribution can be represented by a PSDD.*

Let v be a vtree over the distribution variables. The root of this vtree partitions the variables into $\mathbf{X}$ (variables in left subtree of v) and $\mathbf{Y}$ (variables in right subtree of v). By Theorem 9, we can construct a probabilistic $\mathbf{XY}$-partition of the distribution, which defines the root PSDD node. By repeating the process recursively, we can construct PSDDs for the prime and sub distributions of each constructed $\mathbf{XY}$-partition, until we reach distributions over single variables.

A distribution may have multiple (or even many) probabilistic $\mathbf{XY}$-partitions for the same sets of variables $\mathbf{X}$ and $\mathbf{Y}$. However, exactly one of these is *compressed.*

Theorem 10 *A probabilistic* $\mathbf{XY}$*-partition is compressed iff its sub distributions are distinct. A distribution has a unique compressed* $\mathbf{XY}$*-partition for each* $\mathbf{X}$ *and* $\mathbf{Y}$*.*[7]

[6] Generative learning is concerned with learning a distribution that is optimized for generating the data, yet is not connected to any particular reasoning task. Discriminative learning is concerned with learning a distribution that is optimized for a classification task (i.e., a probabilistic classifier).

[7] Consider a prime distribution Pr_{p_i} where sub $s_i = \bot$. Since $\theta_i = 0$ in this case, the non-zero probabilities of the prime distribution Pr_{p_i} are irrelevant to the distribution represented by the probabilistic $\mathbf{XY}$-partition. The uniqueness claimed in this theorem is modulo such prime distributions.

A decision PSDD node is compressed iff its subs (not their bases) are distinct. A PSDD is compressed iff all its decision nodes are compressed. We now have our canonicity result.

Corollary 4 *Every distribution is induced by a unique, compressed PSDD (given a vtree).*

We close this section by stressing the following point. The notion of compression in PSDDs generalizes a corresponding notion of compression for SDDs (Darwiche 2011). In particular, an SDD is said to be compressed if every decision node has distinct subs. The main observation here is that the SDD of a compressed PSDD may itself be uncompressed. That is, a compressed PSDD may have two distinct sub distributions with equal bases.[8] In fact, the PSDD representation is complete only if one allows the underlying SDDs to be uncompressed.

References

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context–specific independence in Bayesian networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, 115–123.

Bozga, M., and Maler, O. 1999. On the representation of probabilities over structured domains. In *Computer Aided Verification*, 261–273. Springer.

Chang, M.-W.; Ratinov, L.-A.; Rizzolo, N.; and Roth, D. 2008. Learning and inference with constraints. In Fox, D., and Gomes, C. P., eds., *AAAI*, 1513–1518. AAAI Press.

Chavira, M.; Darwiche, A.; and Jaeger, M. 2006. Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning* 42(1-2):4–20.

Choi, A., and Darwiche, A. 2013. Dynamic minimization of sentential decision diagrams. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, 187–194.

Darwiche, A. 2002. A logical approach to factoring belief networks. In *Proceedings of KR*, 409–420.

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Darwiche, A. 2011. SDD: A new canonical representation of propositional knowledge bases. In *IJCAI*, 819–826.

Davis, J., and Domingos, P. 2010. Bottom-up learning of Markov network structure. In *ICML*, 271–278.

De Raedt, L.; Frasconi, P.; Kersting, K.; and Muggleton, S., eds. 2008. *Probabilistic inductive logic programming: theory and applications*. Berlin, Heidelberg: Springer-Verlag.

Fierens, D.; Van den Broeck, G.; Thon, I.; Gutmann, B.; and De Raedt, L. 2011. Inference in probabilistic logic programs using weighted CNF's. In *Proceedings of UAI*, 211–220.

Getoor, L., and Taskar, B., eds. 2007. *An Introduction to Statistical Relational Learning*. MIT Press.

Gogate, V., and Dechter, R. 2007. Samplesearch: A scheme that searches for consistent samples. In *International Conference on Artificial Intelligence and Statistics*, 147–154.

[8]This is never true in our experiments since we start off with a compressed SDD (the SDD package outputs compressed SDDs).

Halpern, J. 1990. An analysis of first-order logics of probability. *Artificial Intelligence* 46(3):311–350.

Jaeger, M.; Nielsen, J. D.; and Silander, T. 2006. Learning probabilistic decision graphs. *International Journal of Approximate Reasoning* 42(1):84–100.

Jaeger, M. 2004. Probabilistic decision graphscombining verification and ai techniques for probabilistic inference. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 12(supp01):19–42.

Lai, Y.-T., and Sastry, S. 1992. Edge-valued binary decision diagrams for multi-level hierarchical verification. In *Proceedings of the 29th ACM/IEEE Design Automation Conference*, 608–613. IEEE Computer Society Press.

Lowd, D., and Rooshenas, A. 2013. Learning Markov networks with arithmetic circuits. In *AISTATS*, 406–414.

Mateescu, R., and Dechter, R. 2008. Mixed deterministic and probabilistic networks. *Annals of Mathematics and Artificial Intelligence* 54(1-3):3–51.

Nilsson, N. 1986. Probabilistic logic. *Artificial intelligence* 28(1):71–87.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, California.

Pipatsrisawat, K., and Darwiche, A. 2008. New compilation languages based on structured decomposability. In *AAAI*, 517–522.

Poole, D. 2003. First-order probabilistic inference. In *Proceedings of IJCAI*, 985–991.

Poon, H., and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, volume 6, 458–463.

Poon, H., and Domingos, P. 2011. Sum-product networks: A new deep architecture. *Proc. 12th Conf. on Uncertainty in Artificial Intelligence* 337–346.

R.I. Bahar; E.A. Frohm; C.M. Gaona; G.D. Hachtel; E. Macii; A. Pardo; and F. Somenzi. 1993. Algebraic Decision Diagrams and Their Applications. In *IEEE /ACM International Conference on CAD*, 188–191. Santa Clara, California: IEEE Computer Society Press.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine learning* 62(1):107–136.

Sanner, S., and McAllester, D. A. 2005. Affine algebraic decision diagrams (aadds) and their application to structured probabilistic inference. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 1384–1390.

Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of IJCAI*, 2178–2185.

Xue, Y.; Choi, A.; and Darwiche, A. 2012. Basing decisions on sentences in decision diagrams. In *AAAI*, 842–849.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Linear Programs for Measuring Inconsistency in Probabilistic Logics

Nico Potyka
Department of Computer Science
FernUniversität in Hagen,
Germany

Abstract

Inconsistency measures help analyzing contradictory knowledge bases and resolving inconsistencies. In recent years several measures with desirable properties have been proposed, but often these measures correspond to combinatorial or non-convex optimization problems that are hard to solve in practice. In this paper, I study a new family of inconsistency measures for probabilistic knowledge bases. All members satisfy many desirable properties and can be computed by means of convex optimization techniques. For two members, I present linear programs whose computation is barely harder than a probabilistic satisfiability test.

1 Introduction

A central problem in the design of expert systems is to assure consistency of the knowledge base. The represented knowledge may originate from several experts or different sources of statistical data, which might be incompatible with each other. Even for a single expert designing a consistent knowledge base can be difficult.

This work focuses on probabilistic knowledge bases. Some heuristic approaches have been developed to resolve inconsistencies by adapting probabilities or deleting parts of the knowledge base (Rödder and Xu 1999; Finthammer, Kern-Isberner, and Ritterskamp 2007), but such methods do not help understanding possible causes for contradictions and lack a well-founded methodology. Inconsistency measures evaluate the degree of inconsistency (Knight 2002; Grant and Hunter 2008; Thimm 2013) and in this way can complement and guide heuristic approaches; however, many existing measures rely on combinatorial or non-convex optimization problems that can be hard to solve in practice.

In this paper, I study a new family of inconsistency measures that correspond to convex optimization problems. As opposed to non-convex problems, they lack (non-global) local minima and therefore are significantly easier to compute. Two members can be computed by particularly efficient linear programming techniques. The measures are based on the fact that a probabilistic knowledge base is satisfiable if and only if a corresponding system of linear equations can be solved. If the equation system is unsolvable, we can search for a vector that violates the system in a minimal way. The overall violation can be measured with respect to some norm like the euclidean norm. If the violation value is zero, the knowledge base is consistent. Otherwise, the value increases continuously with the degree of violation. This is the basic idea of the family of *minimal violation measures*.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Inconsistency Measures for Probabilistic Logics

In recent years various approaches have been developed to measure the degree to which knowledge bases are inconsistent, see, e.g., (Knight 2002; Grant and Hunter 2008; Doder et al. 2010). A short overview of measures for classical logics can be found in (Hunter and Konieczny 2006). Extensions and new measures for probabilistic logics are summarized in (Thimm 2013). I quickly sketch some examples that will be referred to in the following.

The easiest measure one can possibly think of is the *drastic measure* (Hunter and Konieczny 2006; Thimm 2013) that yields 0 if the knowledge base is consistent and 1 otherwise.

A class of measures that is interesting for both classical and probabilistic knowledge bases relies on *minimal inconsistent sets* (Hunter and Konieczny 2008; Thimm 2013). A subset of a knowledge base is called a minimal inconsistent set, if it is inconsistent and removing a single rule renders the set consistent. Hence, the contained rules are in conflict with each other and the conflict can be resolved by removing a single rule. Interesting measures can be defined by counting the minimal inconsistent sets, or by combining the numbers of minimal inconsistent sets that contain certain rules (Hunter and Konieczny 2008; Thimm 2013).

A particularly interesting class of measures derives from the idea of *minimally changing* the knowledge base to a consistent one. In (Thimm 2009), the probabilities of the rules are changed and an inconsistency measure is defined by the minimal sum of all absolute changes. In (Picado-Muiño 2011) the probabilities are relaxed to probability intervals. The corresponding inconsistency measure is defined by arranging the size of these intervals in a vector and measuring the vector's size with respect to a norm. Both measures are particularly interesting because they yield an *optimal repair* for a knowledge base, that is, a consistent knowledge base that is obtained from an inconsistent one by minimal changes.

Computational Aspects: The drastic measure corresponds to a satisfiability test and can be regarded as the measure that is most easily computable, because each reasonable measure should distinguish consistent from inconsistent sets. Deciding whether a probabilistic knowledge base is inconsistent is NP-hard in general (Georgakopoulos, Kavvadias, and Papadimitriou 1988). Still, for knowledge bases containing more than one hundred propositional variables and several hundred rules satisfiability can be tested by linear programming techniques in reasonable time (Jaumard, Hansen, and Poggi 1991; Hansen and Perron 2008).

Measures based on minimal inconsistent sets correspond to combinatorial optimization problems. There has been some progress in implementing such measures efficiently for classical logics (Konieczny and Roussel 2013), but the key idea is that a classical knowledge base can be regarded as a single formula in conjunctive normal form. This, however, does not apply to probabilistic knowledge bases. As the number of subsets of a knowledge base is exponential in the size of the knowledge base, the vital problems are efficiently enumerating candidates for minimal inconsistent sets and testing their satisfiability.

Minimal change measures like in (Thimm 2009) and (Picado-Muiño 2011) correspond to non-convex optimization problems that suffer from (non-global) local optima. Therefore, computing these measures can be hard in practice.

Overview of the Paper

The logical framework and some notations are explained in Section 2. In Section 3, I introduce the family of minimal violation measures and demonstrate in some examples how they differ from other measures and how they can be used to analyze and to resolve inconsistencies. In Section 4, I show that minimal violation measures satisfy several desirable properties that have been considered in the literature (Knight 2003; Hunter and Konieczny 2006; Thimm 2013). Section 5 deals with computational issues. I present linear programs for two minimal violation measures and compare their performance to other measures using prototypical implementations. Subsequently, I summarize some empirical estimates for the performance of typical optimization algorithms (Boyd and Vandenberghe 2004; Matousek and Gärtner 2007) and use them to provide some rough estimates for the performance of different measures when applying general purpose algorithms.

2 Preliminaries

Consider a propositional language $\mathcal{L}_\Sigma$ built up over a finite set of propositional variables Σ using logical connectives like conjunction and disjunction in the usual way. For formulas $\mathcal{F}, \mathcal{G} \in \mathcal{L}_\Sigma$, negation $\neg\mathcal{F}$ is abbreviated by an overbar $\overline{\mathcal{F}}$ and conjunction $\mathcal{F} \wedge \mathcal{G}$ by juxtaposition $\mathcal{F}\mathcal{G}$. A *possible world* is a classical logical interpretation $\omega : \Sigma \to \{0,1\}$ assigning a truth value to each propositional variable. Let Ω denote the set of all possible worlds. An atom $\mathfrak{a}$ is satisfied by ω iff $\omega(\mathfrak{a}) = 1$. The definition is extended to complex formulas in the usual way. The *classical models* $\mathrm{Mod}(\mathcal{G})$ of a formula $\mathcal{G}$ are the possible worlds $\omega \in \Omega$ that satisfy $\mathcal{G}$. $\mathcal{F}, \mathcal{G} \in \mathcal{L}_\Sigma$ are called equivalent, $\mathcal{F} \equiv \mathcal{G}$, iff $\mathrm{Mod}(\mathcal{F}) = \mathrm{Mod}(\mathcal{G})$.

Consider the probabilistic conditional language $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma) := \{(\mathcal{G}|\mathcal{F})[\rho] \,|\, \mathcal{G}, \mathcal{F} \in \mathcal{L}_\Sigma, \rho \in [0,1]\}$ over Σ (Lukasiewicz 1999). Intuitively, a conditional $(\mathcal{G}|\mathcal{F})[\rho]$ expresses that our belief in $\mathcal{G}$ given that $\mathcal{F}$ holds is ρ. Unconditioned probabilistic formulas are obtained for the special case $(\mathcal{G} \,|\, \top)[\rho]$, where $\top$ denotes a tautological formula. $(\mathcal{G} \,|\, \top)[\rho]$ is abbreviated by $(\mathcal{G})[\rho]$.

Probabilistic interpretations $\mathcal{P} : \Omega \to [0,1]$ are probability functions over possible worlds. For a formula $\mathcal{G} \in \mathcal{L}_\Sigma$ let $\mathcal{P}(\mathcal{G}) = \sum_{\omega \in \mathrm{Mod}(\mathcal{G})} \mathcal{P}(\omega)$. A probabilistic interpretation $\mathcal{P}$ satisfies a conditional $(\mathcal{G}|\mathcal{F})[\rho]$ iff $\mathcal{P}(\mathcal{G}\mathcal{F}) = \rho \cdot \mathcal{P}(\mathcal{F})$. If $\mathcal{P}(\mathcal{F}) > 0$ this definition corresponds to the definition of conditional probability $\mathcal{P}(\mathcal{G} \,|\, \mathcal{F}) = \frac{\mathcal{P}(\mathcal{G}\mathcal{F})}{\mathcal{P}(\mathcal{F})} = \rho$. As often done (Paris and Vencovská 1989; Lukasiewicz 1999; Thimm 2013), I do not demand $\mathcal{P}(\mathcal{F}) > 0$.

A *probabilistic knowledge base* $\mathcal{R} \subset (\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$ is a finite set of conditionals. A probabilistic interpretation $\mathcal{P}$ satisfies $\mathcal{R}$ iff it satisfies each $c \in \mathcal{R}$. Let $\mathrm{Mod}(\mathcal{R})$ denote the set of probabilistic models of $\mathcal{R}$, i.e., the set of probabilistic interpretations over Ω that satisfy $\mathcal{R}$.

Two conditionals $c_1, c_2 \in (\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$ are called *equivalent*, $c_1 \equiv c_2$, iff $\mathrm{Mod}(c_1) = \mathrm{Mod}(c_2)$. Following (Thimm 2013), two knowledge bases $\mathcal{R}_1, \mathcal{R}_2$ over $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$ are called *(semi-extensionally) equivalent*, $\mathcal{R}_1 \equiv \mathcal{R}_2$, iff there is a bijection $B : \mathcal{R}_1 \to \mathcal{R}_2$ such that $c_1 \equiv B(c_1)$ for each $c_1 \in \mathcal{R}_1$. That is, both knowledge bases contain the same number of conditionals and these conditionals differ only in their syntactical representation. If both $\mathcal{G}\mathcal{F}$ and $\overline{\mathcal{G}}\mathcal{F}$ are satisfiable, $(\mathcal{G}|\mathcal{F})[\rho]$ is called a *normal conditional*. The following lemma states an intermediate result that was obtained in (Thimm 2013) in the proof of Theorem 2, (4).

Lemma 2.1. *(Thimm 2013) Let $c_1, c_2 \in \mathcal{R}$ be normal, $c_i = (\mathcal{G}_i \,|\, \mathcal{F}_i)[\rho_i]$. If $c_1 \equiv c_2$, then $\mathcal{F}_1 \equiv \mathcal{F}_2$ and either*

- *$\mathcal{G}_1\mathcal{F}_1 \equiv \mathcal{G}_2\mathcal{F}_2$ and $\rho_1 = \rho_2$, or*
- *$\mathcal{G}_1\mathcal{F}_1 \equiv \overline{\mathcal{G}_2}\mathcal{F}_2$ and $\rho_1 = 1 - \rho_2$.*

As done in (Thimm 2013), I will assume that knowledge bases contain only normal conditionals.

If $\mathrm{Mod}(\mathcal{R}) \neq \emptyset$, $\mathcal{R}$ is called consistent. Given a consistent knowledge base, we can draw probabilistic conclusions in different ways. For example, we can consider the whole set $\mathrm{Mod}(\mathcal{R})$ and compute probability intervals for arbitrary formulas, see, e.g., (Nilsson 1986; Jaumard, Hansen, and Poggi 1991). Another possible way is to select a single best model from $\mathrm{Mod}(\mathcal{R})$ with respect to some evaluation function and to use it to compute point probabilities for arbitrary (conditional) formulas, see, e.g., (Paris and Vencovská 1989; Kern-Isberner 2001).

An inconsistency measure assigns a nonnegative inconsistency value to each knowledge base.

Definition 2.2. *An* inconsistency measure Inc *is a function* $\mathrm{Inc} : 2^{(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)} \to \mathbb{R}_{\geq 0}$ *such that* $\mathrm{Inc}(\mathcal{R}) = 0$ *if and only if* $\mathcal{R}$ *is consistent.*

Several interesting properties have been considered in the literature to motivate and to compare different inconsistency measures, see, e.g., (Knight 2002; Hunter and Konieczny 2006; Picado-Muiño 2011). An overview of inconsistency measures for probabilistic logics and a comparison of some of their properties can be found in (Thimm 2013).

Matrix representation of knowledge bases

Sometimes it is more convenient to represent the conditional satisfaction constraints in matrix notation (Nilsson 1986; Jaumard, Hansen, and Poggi 1991). Suppose that the possible worlds in Ω are ordered in an arbitrary but fixed sequence $\omega_1, \ldots, \omega_n$, $n = |\Omega|$, so that each probabilistic interpretation $\mathcal{P}$ can be identified with a column vector $\vec{\pi} = (\mathcal{P}(\omega_1) \ldots \mathcal{P}(\omega_n))^\mathsf{T}$, where the symbol $^\mathsf{T}$ denotes transposition of vectors. The set of all probabilistic interpretations of $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$ corresponds to the set $\Pi_n = \{\vec{\pi} \in \mathbb{R}^n \mid \vec{\pi} \geq \vec{0} \text{ and } \sum_{i=1}^n \vec{\pi}_i = 1\}$, where $\geq$ denotes componentwise inequality.

To compare vectors, we can consider different norms on $\mathbb{R}^n$. A p-norm is a function $\|.\|_p : \mathbb{R}^n \to \mathbb{R}$, which is defined by $\|\vec{\pi}\|_p = \sqrt[p]{\sum_{i=1}^n |\vec{\pi}_i|^p}$. The best known special cases are the 1-*norm* $\|\vec{\pi}\|_1 = \sum_{i=1}^n |\vec{\pi}_i|$, the *euclidean norm* $\|\vec{\pi}\|_2 = \sqrt[2]{\sum_{i=1}^n \vec{\pi}_i^2}$ and the limit for $p \to \infty$, the *maximum norm* $\|\vec{\pi}\|_\infty = \max\{|\vec{\pi}_i| \mid 1 \leq i \leq n\}$.

Recall that a probabilistic model of c satisfies the equation $\mathcal{P}(\mathcal{GF}) = \rho \cdot \mathcal{P}(\mathcal{F})$. Subtracting $\rho \cdot \mathcal{P}(\mathcal{F})$ yields

$$\begin{aligned}
0 &= \mathcal{P}(\mathcal{GF}) - \rho \cdot \mathcal{P}(\mathcal{F}) \\
&= \mathcal{P}(\mathcal{GF}) - \rho \cdot (\mathcal{P}(\mathcal{GF}) + \mathcal{P}(\overline{\mathcal{G}}\mathcal{F})) \\
&= (1-\rho) \cdot \mathcal{P}(\mathcal{GF}) - \rho \cdot \mathcal{P}(\overline{\mathcal{G}}\mathcal{F}) \\
&= \sum_{\omega \in \mathrm{Mod}(\mathcal{GF})} \mathcal{P}(\omega) \cdot (1-\rho) - \sum_{\omega \in \mathrm{Mod}(\overline{\mathcal{G}}\mathcal{F})} \mathcal{P}(\omega) \cdot \rho \\
&= \sum_{\omega \in \Omega} \mathcal{P}(\omega) \cdot (1_{\{\mathcal{GF}\}}(\omega) \cdot (1-\rho) - 1_{\{\overline{\mathcal{G}}\mathcal{F}\}}(\omega) \cdot \rho),
\end{aligned}$$

where for a formula F the indicator function $1_{\{F\}}(\omega)$ maps to 1 iff $\omega \models_{\mathcal{L}_\Sigma} F$ and to 0 otherwise. In vector notation this equation becomes

$$\vec{a}_c^\mathsf{T} \vec{\pi} = 0,$$

where $\vec{a}_c^\mathsf{T}$ is the transpose of the vector

$$(1_{\{\mathcal{GF}\}}(\omega_j) \cdot (1-\rho) - 1_{\{\overline{\mathcal{G}}\mathcal{F}\}}(\omega_j) \cdot \rho)_{1 \leq j \leq n}.$$

That is, the j-th component of $\vec{a}^\mathsf{T}$ is either $1-\rho$, $-\rho$ or 0 dependent on whether the conditional is verified ($1_{\{\mathcal{GF}\}}(\omega_j) = 1$), falsified ($1_{\{\overline{\mathcal{G}}\mathcal{F}\}}(\omega_j) = 1$) or not applicable ($\omega_j \models_{\mathcal{L}_\Sigma} \overline{\mathcal{F}}$) in the j-th world.

Given a knowledge base $\mathcal{R}$, suppose that the conditionals in $\mathcal{R}$ are ordered in an arbitrary but fixed sequence $c_1, \ldots, c_m$, $m = |\mathcal{R}|$, so that we can identify the i-th conditional with a row vector $\vec{a}_i^\mathsf{T}$. Then we can associate $\mathcal{R}$ with the $(m \times n)$-matrix

$$A_\mathcal{R} = \begin{pmatrix} \vec{a}_1^\mathsf{T} \\ \ldots \\ \vec{a}_m^\mathsf{T} \end{pmatrix}.$$

Using the introduced notations, the question whether $\mathcal{R}$ is consistent becomes the question whether the linear equation system $A_\mathcal{R} x = 0$ can be solved by a probability vector $\vec{\pi} \in \Pi_n$ (Nilsson 1986; Jaumard, Hansen, and Poggi 1991).

3 Minimal Violation Measures

Given a knowledge base $\mathcal{R}$ over $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$, the corresponding matrix $A_\mathcal{R}$ and some p-norm $\|.\|_p$ as explained in Section 2, we consider the following optimization problem:

$$\begin{aligned}
\min_{\vec{\pi} \in \mathbb{R}^n} \quad & \|A_\mathcal{R} \vec{\pi}\|_p \qquad (1) \\
\textit{subject to} \quad & \sum_{i=1}^n \vec{\pi}_i = 1 \\
& \vec{\pi} \geq 0.
\end{aligned}$$

The constraints just guarantee that the solution is a probabilistic interpretation in Π_n. The objective $\|A_\mathcal{R} \vec{\pi}\|_p$ measures to which degree $\vec{\pi}$ violates $\mathcal{R}$ with respect to the p-norm. The choice of p depends on the application. For $p = 1$ a lower total violation might be obtained but the violation can be rather extreme. As p grows, higher violations are penalized more heavily. The following example illustrates this behavior for the extreme cases $p = 1$ and $p = \infty$.

Example 3.1. *Consider the vectors $v_1 = (\frac{1}{4}\ \frac{1}{4}\ \frac{1}{4}\ \frac{1}{4})^\mathsf{T}$, $v_2 = (0\ 0\ 0\ \frac{1}{2})^\mathsf{T}$ in $\mathbb{R}^4$. v_1 corresponds to a violation of $\frac{1}{4}$ for four conditionals. In v_2 no conditional but the last one is violated. We have $\|v_1\|_1 = 1$, $\|v_2\|_1 = \frac{1}{2}$. Hence v_2 is preferable to v_1 with respect to $\|.\|_1$, because the overall error is lower. In contrast, $\|v_1\|_\infty = \frac{1}{4}$, $\|v_2\|_\infty = \frac{1}{2}$. Hence v_1 is preferable to v_2 with respect to $\|.\|_\infty$, because the highest error is lower. Which behavior is more desirable, depends on the specific application.*

The *minimal violation inconsistency measures* are defined by means of optimization problem (1). A vector $\vec{\pi} \in \Pi_n$ minimizing optimization problem (1) will be called *minimal violating* in the following. If $\mathcal{R}$ is consistent, the minimal violating $\vec{\pi}$ are the models of $\mathcal{R}$. For inconsistent $\mathcal{R}$, they can be regarded to be as close to a model as possible.

Definition 3.2. *For $p \in \mathbb{N} \cup \{\infty\}$ the* minimal violation p-inconsistency measure $\mathrm{Inc}_\Pi^p : 2^{(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)} \to \mathbb{R}$ *is defined by $\mathrm{Inc}_\Pi^p(\mathcal{R}) = \epsilon$, where ϵ is the solution of optimization problem* (1)*.*

Proposition 3.3. Inc_Π^p *is an inconsistency measure.*

Proof. To begin with, we convince ourselves that Inc_Π^p is well-defined. That is, there always exists a unique minimum for optimization problem (1). The objective is the composition of a p-norm and the linear function induced by the constraint matrix $A_\mathcal{R}$ and therefore continuous and convex. The feasible region of optimization problem (1) is the convex and compact probability simplex. As each continuous function on a compact set takes on its minimum, the minimum exists. It is also unique, because the minimum for each convex function on a convex set is unique.

It remains to show that Inc_Π^p is an inconsistency measure as defined in Definition 2.2. By definition of optimization

problem (1), it holds $\mathrm{Inc}^p_\Pi(\mathcal{R}) \geq 0$. $\mathcal{R}$ is consistent if and only if there is a $\vec{\pi} \in \Pi_n$ such that $A_\mathcal{R}\vec{\pi} = 0$. Hence, by definiteness of norms, it follows $\mathrm{Inc}^p_\Pi(\mathcal{R}) = 0$ if and only if $\mathcal{R}$ is consistent. □

Before investigating general properties of Inc^p_Π, we consider some simple examples and explore differences between Inc^p_Π and the drastic measure (Hunter and Konieczny 2006; Thimm 2013), measures based on minimal inconsistent sets (Hunter and Konieczny 2008; Thimm 2013) as well as minimal change measures (Thimm 2009; Picado-Muiño 2011) that have been explained in Section 1.

Example 3.4. *Consider $\Sigma = \{A\}$ and the knowledge base $\mathcal{R}_\rho = \{(A)[\rho], (A)[1-\rho]\}$ over Σ for $0 \leq \rho \leq 0.5$. For instance, $\mathcal{R}_{0.5} = \{(A)[0.5], (A)[0.5]\} = \{(A)[0.5]\}$ is consistent and $\mathcal{R}_0 = \{(A)[0], (A)[1]\}$ is inconsistent. Intuitively, the degree of inconsistency of $\mathcal{R}_\rho$ should increase as ρ decreases from 0.5 to 0. The following table shows some inconsistency values of $\mathcal{R}_\rho$ with respect to Inc^p_Π.*

	$\mathcal{R}_{0.5}$	$\mathcal{R}_{0.49}$	$\mathcal{R}_{0.4}$	$\mathcal{R}_{0.2}$	$\mathcal{R}_0$
Inc^1_Π	0	0.02	0.2	0.6	1
Inc^∞_Π	0	0.01	0.1	0.3	0.5

Minimal change measures also capture this intuitive increase of inconsistency. In contrast, the drastic measure and inconsistency measures based on minimal inconsistent sets cannot distinguish the knowledge bases $\mathcal{R}_\rho$ for $0 \leq \rho < 0.5$, because all these $\mathcal{R}_\rho$ are inconsistent and in each case the only minimal inconsistent set is $\mathcal{R}_\rho$ itself.

Example 3.5. *Consider $\Sigma = \{A, B\}$ and the knowledge base $\mathcal{R} = \{(A)[0.8], (B)[0.6], (B \mid A)[0.9]\}$. We find*

$$\mathrm{Inc}^1_\Pi(\mathcal{R}) \approx 0.120 \text{ and } \mathrm{Inc}^\infty_\Pi(\mathcal{R}) \approx 0.041.$$

Intuitively, we can conjecture that decreasing the probability of A, increasing the probability of B or decreasing the probability of B given A might resolve the contradiction. We start by decreasing the probability of A and obtain for $\mathcal{R}_1 = \{(A)[0.75], (B)[0.6], (B \mid A)[0.9]\}$

$$\mathrm{Inc}^1_\Pi(\mathcal{R}_1) \approx 0.075 \text{ and } \mathrm{Inc}^\infty_\Pi(\mathcal{R}_1) \approx 0.025.$$

As suspected, the degree of inconsistency decreased. Next, we increase the probability of B and obtain for $\mathcal{R}_2 = \{(A)[0.75], (B)[0.65], (B \mid A)[0.9]\}$

$$\mathrm{Inc}^1_\Pi(\mathcal{R}_2) \approx 0.024 \text{ and } \mathrm{Inc}^\infty_\Pi(\mathcal{R}_2) \approx 0.008.$$

Again, the degree of inconsistency decreased significantly. Now we decrease the probability of B given A and obtain for $\mathcal{R}_3 = \{(A)[0.75], (B)[0.65], (B \mid A)[0.85]\}$

$$\mathrm{Inc}^1_\Pi(\mathcal{R}_3) = 0 \text{ and } \mathrm{Inc}^\infty_\Pi(\mathcal{R}_3) = 0.$$

Hence $\mathcal{R}_3$ yields a repair for $\mathcal{R}$.

If we had no idea, which conditionals might have caused the inconsistencies and in which way, we could try to adjust probabilities arbitrarily. For example, if we increased the probability of A in the first step instead of decreasing it, we would obtain with $\mathcal{R}_0 = \{(A)[0.85], (B)[0.6], (B \mid A)[0.9]\}$

$$\mathrm{Inc}^1_\Pi(\mathcal{R}_0) \approx 0.165 \text{ and } \mathrm{Inc}^\infty_\Pi(\mathcal{R}_0) \approx 0.057.$$

So the inconsistency values increased compared to our initial knowledge base $\mathcal{R}$ and we can see that this is not a reasonable adjustment. In the same way, we can learn that decreasing the probability of B or increasing the probability of B given A aggravates the conflict.

In contrast, the drastic measure provides no interesting information to resolve such inconsistencies. Measures based on minimal inconsistent sets are more helpful. $\mathcal{R}$ is the only minimal inconsistent set, so we could repair $\mathcal{R}$ by deleting a single conditional. Furthermore, we learn that all conditionals are in conflict with each other, so each adjustment might be reasonable. However, $\mathcal{R}$, $\mathcal{R}_0$ and $\mathcal{R}_1$ cannot be distinguished by measures based on minimal inconsistent sets, because in each case the knowledge base itself is the only minimal inconsistent subset. Therefore, they cannot provide information about reasonable adjustments, unless a conflict has been resolved by the adjustment coincidentally.

The best choice in this scenario is admittedly a minimal change measure, because it yields a best repair without further ado. A best repair (there can be several) computed by the measure from (Thimm 2009) is $\mathcal{R}_{opt} = \{(A)[0.752], (B)[0.648], (B \mid A)[0.86]\}$. As we will see later on, minimal violation measures become superior to minimal change measures for larger knowledge bases because of computational advantages.

4 Properties of Minimal Violation Measures

We now turn to some properties that have been considered interesting for inconsistency measures in the literature (Knight 2003; Hunter and Konieczny 2006; Thimm 2013).

To give an intuitive overview, Inc^p_Π is *language* and *representation invariant*. That is, the degree of inconsistency of a knowledge base determined by Inc^p_Π is independent of the number of propositional variables in the underlying language and the syntactical representation of the conditionals. Inc^p_Π is *monotone* and for $p = 1$ *super-additive*. The former means that adding new conditionals can only increase the degree of inconsistency. The latter means that if we combine two knowledge bases, the inconsistency value of the combined knowledge base is at least as large as the sum of both single inconsistency values. Furthermore, Inc^p_Π is *weakly independent*. That is, if we add consistent knowledge about propositional variables that are not contained in our current knowledge base, the inconsistency value does not increase. The term *weak* stems from the fact that stronger independence properties for inconsistent sets have been considered in the literature (Hunter and Konieczny 2006; Thimm 2013). Finally, Inc^p_Π is *continuous*. That is, small adjustments in the probabilities of the knowledge base change the corresponding inconsistency value only slightly. Some other properties have been considered in (Knight 2003; Hunter and Konieczny 2006; Thimm 2013), but in my opinion the stated ones are among the most intuitive for probabilistic knowledge bases and I leave it for future work to

answer whether minimal violation measures satisfy or violate further properties.

We need some additional notation in this section. For a conditional $c \in (\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$ let $\mathrm{At}(c)$ denote the atoms from Σ that appear in c and for a knowledge base $\mathcal{R}$ let $\mathrm{At}(\mathcal{R})$ denote the atoms appearing in $\mathcal{R}$. Note that we can consider the same knowledge base $\mathcal{R}$ with respect to different languages $(\Sigma' \mid \Sigma')$ as long as $\mathrm{At}(\mathcal{R}) \subseteq \Sigma'$. We will see that the size of Σ' does not influence the inconsistency value with respect to Inc^p_Π, but to prove this fact, we have to make a distinction between the same knowledge base in different languages. For a knowledge base $\mathcal{R}$ that is defined with respect to a language $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$, I call $\mathcal{R}$ *restricted to* $(\Sigma' \mid \Sigma')$ if $\Sigma' \subseteq \Sigma$ and I call $\mathcal{R}$ *lifted to* $(\Sigma' \mid \Sigma')$ if $\Sigma \subseteq \Sigma'$.

We first consider language invariance, which means that changing our language does not change the inconsistency value of a knowledge base (Knight 2003).

Lemma 4.1. *Let V be a new propositional variable not contained in Σ and let $\Sigma' = \Sigma \cup \{V\}$. Let $\mathcal{R}$ be a knowledge base over $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$ and let $\mathcal{R}'$ denote $\mathcal{R}$ lifted to $(\Sigma' \mid \Sigma')$. Then $\mathrm{Inc}^p_\Pi(\mathcal{R}) = \mathrm{Inc}^p_\Pi(\mathcal{R}')$.*

Proof. Let $\omega_1, \ldots, \omega_n$ be the ordered possible worlds over Σ. Let $\omega'_1, \ldots, \omega'_n, \omega'_{n+1}, \ldots, \omega'_{2n}$ denote the possible worlds over Σ', such that ω'_i, ω'_{i+n} correspond to ω_i for the variables in Σ, $\omega'_1, \ldots, \omega'_n$ falsify V, and $\omega'_{n+1}, \ldots, \omega'_{2n}$ satisfy V. Note that $A_{\mathcal{R}'} = (A_\mathcal{R}\ A_\mathcal{R})$ by construction, because $\mathcal{R}'$ corresponds to $\mathcal{R}$ and does not contain V .

Let $\mathrm{Inc}^p_\Pi(\mathcal{R}) = \delta$ and let $\vec{\pi}^*$ be minimal violating, i.e., $\|A_\mathcal{R}\vec{\pi}^*\|_p = \delta$. Then $\begin{pmatrix}\vec{\pi}^* \\ \vec{0}\end{pmatrix} \in \mathbb{R}^{2n}$ corresponds to a probability function over the interpretations of Σ' and

$$\begin{aligned}\mathrm{Inc}^p_\Pi(\mathcal{R}') &\leq \| \begin{pmatrix} A_\mathcal{R}\ A_\mathcal{R}\end{pmatrix} \begin{pmatrix}\vec{\pi}^* \\ \vec{0}\end{pmatrix} \|_p = \|A_\mathcal{R}\vec{\pi}^*\|_p \\ &= \mathrm{Inc}^p_\Pi(\mathcal{R}).\end{aligned}$$

Conversely, if $\mathrm{Inc}^p_\Pi(\mathcal{R}') = \delta'$ and $\begin{pmatrix}\vec{\pi}^*_1 \\ \vec{\pi}^*_2\end{pmatrix} \in \mathbb{R}^{2n}$ is minimal violating, then $(\vec{\pi}^*_1 + \vec{\pi}^*_2) \in \mathbb{R}^n$ corresponds to a probability function over the interpretations of Σ and

$$\begin{aligned}\mathrm{Inc}^p_\Pi(\mathcal{R}) &\leq \|A_\mathcal{R}(\vec{\pi}^*_1 + \vec{\pi}^*_2)\|_p = \|A_\mathcal{R}\vec{\pi}^*_1 + A_\mathcal{R}\vec{\pi}^*_2\|_p \\ &= \| \begin{pmatrix} A_\mathcal{R}\ A_\mathcal{R}\end{pmatrix} \begin{pmatrix}\vec{\pi}^*_1 \\ \vec{\pi}^*_2\end{pmatrix} \|_p = \mathrm{Inc}^p_\Pi(\mathcal{R}').\end{aligned}$$

Hence, $\mathrm{Inc}^p_\Pi(\mathcal{R}') \leq \mathrm{Inc}^p_\Pi(\mathcal{R}) \leq \mathrm{Inc}^p_\Pi(\mathcal{R}')$, that is, both inconsistency values have to be equal. □

Proposition 4.2 (Language invariance). *Let $\mathcal{R}$ be a knowledge base over $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$ and let $\mathrm{At}(\mathcal{R}) \subseteq \Sigma'$. Let $\mathcal{R}'$ denote $\mathcal{R}$ over $(\Sigma' \mid \Sigma')$. Then $\mathrm{Inc}^p_\Pi(\mathcal{R}) = \mathrm{Inc}^p_\Pi(\mathcal{R}')$.*

Proof. For $\Sigma \subseteq \Sigma'$, the claim follows from Lemma 4.1 by induction over the variables in $\Sigma' \setminus \Sigma$.

For the general case, consider $\Sigma'' = \Sigma \cap \Sigma'$ and let $\mathcal{R}'' \subset (\Sigma'' \mid \Sigma'')$ denote $\mathcal{R}$ restricted to $(\Sigma'' \mid \Sigma'')$. Then $\Sigma'' \subseteq \Sigma$ and $\Sigma'' \subseteq \Sigma'$ and therefore $\mathrm{Inc}^p_\Pi(\mathcal{R}'') = \mathrm{Inc}^p_\Pi(\mathcal{R}) = \mathrm{Inc}^p_\Pi(\mathcal{R}')$. □

A measure should also be invariant with respect to syntactical changes of the conditionals (Knight 2003). This property has also been called *irrelevance of syntax* (Thimm 2013).

Proposition 4.3 (Representation invariance). *Let $\mathcal{R}_1, \mathcal{R}_2$ be knowledge bases over $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$. If $\mathcal{R}_1 \equiv \mathcal{R}_2$, then $\mathrm{Inc}^p_\Pi(\mathcal{R}_1) = \mathrm{Inc}^p_\Pi(\mathcal{R}_2)$.*

Proof. As $\mathcal{R}_1 \equiv \mathcal{R}_2$, there is a bijection $B : \mathcal{R}_1 \to \mathcal{R}_2$ such that $c \equiv B(c)$ for each $c \in \mathcal{R}_1$. Consider an arbitrary conditional $c_1 \in \mathcal{R}_1$ such that $B(c_1) = c_2$. For $i = 1, 2$ let $c_i = (\mathcal{G}_i \mid \mathcal{F}_i)[\rho_i]$ and let $\vec{a}^\mathsf{T}_i$ denote the corresponding constraint vectors. By Lemma 2.1, there are two possible cases. If $\mathcal{G}_1\mathcal{F}_1 \equiv \mathcal{G}_2\mathcal{F}_2$ and $\rho_1 = \rho_2$, then $\vec{a}^\mathsf{T}_1 = \vec{a}^\mathsf{T}_2$. Otherwise, if $\mathcal{G}_1\mathcal{F}_1 \equiv \overline{\mathcal{G}_2}\mathcal{F}_2$ and $\rho_1 = 1 - \rho_2$, then also $\mathcal{G}_2\mathcal{F}_2 \equiv \overline{\mathcal{G}_1}\mathcal{F}_1$ because $\equiv$ is symmetric. Therefore, for the j-th component of the constraint vectors it holds

$$\begin{aligned}(\vec{a}^\mathsf{T}_1)_j &= 1_{\{\mathcal{G}_1\mathcal{F}_1\}}(\omega_j) \cdot (1 - \rho_1) - 1_{\{\overline{\mathcal{G}_1}\mathcal{F}_1\}}(\omega_j) \cdot \rho_1 \\ &= 1_{\{\overline{\mathcal{G}_2}\mathcal{F}_2\}}(\omega_j) \cdot \rho_2 - 1_{\{\mathcal{G}_2\mathcal{F}_2\}}(\omega_j) \cdot (1 - \rho_2) \\ &= -(1_{\{\mathcal{G}_2\mathcal{F}_2\}}(\omega_j) \cdot (1 - \rho_2) - 1_{\{\overline{\mathcal{G}_2}\mathcal{F}_2\}}(\omega_j) \cdot \rho_2) \\ &= -(\vec{a}^\mathsf{T}_2)_j.\end{aligned}$$

Hence $\vec{a}^\mathsf{T}_1 = -\vec{a}^\mathsf{T}_2$ and $|\vec{a}^\mathsf{T}_1\vec{\pi}| = |-\vec{a}^\mathsf{T}_2\vec{\pi}| = |\vec{a}^\mathsf{T}_2\vec{\pi}|$ for all $\vec{\pi} \in \mathbb{R}^n$. As $c_1 \in \mathcal{R}_1$ was arbitrary, it follows $\mathrm{Inc}^p_\Pi(\mathcal{R}_1) = \mathrm{Inc}^p_\Pi(\mathcal{R}_2)$. □

Monotonicity states that if we add additional conditionals to a knowledge base, the degree of inconsistency cannot decrease (Hunter and Konieczny 2006; Thimm 2013).

Proposition 4.4 (Monotonicity). *Let $\mathcal{R}, \mathcal{R}'$ be knowledge bases over $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$. If $\mathcal{R}' \subseteq \mathcal{R}$, then $\mathrm{Inc}^p_\Pi(\mathcal{R}') \leq \mathrm{Inc}^p_\Pi(\mathcal{R})$.*

Proof. Let $\{1, \ldots, m\}$ denote the indices of the conditionals in $\mathcal{R}'$ and let $\{m+1, \ldots, m+k\}$ denote the indices of the additional conditionals in $\mathcal{R} \setminus \mathcal{R}'$. Let $\vec{a}^\mathsf{T}_i$ denote the corresponding row vectors and let $\vec{\pi}$ be minimal violating for $\mathcal{R}$. Then

$$\begin{aligned}\mathrm{Inc}^p_\Pi(\mathcal{R}') &\leq \sqrt[p]{\sum_{i=1}^{m} |\vec{a}^\mathsf{T}_i\vec{\pi}|^p} \leq \sqrt[p]{\sum_{i=1}^{m+k} |\vec{a}^\mathsf{T}_i\vec{\pi}|^p} \\ &= \mathrm{Inc}^p_\Pi(\mathcal{R}).\end{aligned}$$

□

For $p = 1$, Inc^p_Π also satisfies a stronger property that has been called *super-additivity* in (Thimm 2013).

Proposition 4.5 (Super-Additivity). *Let $\mathcal{R}_1, \mathcal{R}_2$ be knowledge bases over $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$. If $\mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset$, then $\mathrm{Inc}^1_\Pi(\mathcal{R}_1 \cup \mathcal{R}_2) \geq \mathrm{Inc}^1_\Pi(\mathcal{R}_1) + \mathrm{Inc}^1_\Pi(\mathcal{R}_2)$.*

Proof. Let $\{1, \ldots, m_1\}$ denote the indices of the conditionals in $\mathcal{R}_1$, and let $\{m_1+1, \ldots, m_1+m_2\}$ denote the indices

of the conditionals in $\mathcal{R}_2$. Let $\vec{a}_i^\mathsf{T}$ denote the corresponding row vectors. Then

$$\begin{aligned}\mathrm{Inc}^1_\Pi(\mathcal{R}_1 \cup \mathcal{R}_2) &= \min_{\vec{\pi}\in\mathbb{R}^n}\Big(\sum_{i=1}^{m_1} |\vec{a}_i^\mathsf{T}\vec{\pi}| + \sum_{i=m_1+1}^{m_1+m_2} |\vec{a}_i^\mathsf{T}\vec{\pi}|\Big)\\ &\geq \Big(\min_{\vec{\pi}\in\mathbb{R}^n}\sum_{i=1}^{m_1} |\vec{a}_i^\mathsf{T}\vec{\pi}|\Big) + \Big(\min_{\vec{\pi}\in\mathbb{R}^n}\sum_{i=m_1+1}^{m_1+m_2} |\vec{a}_i^\mathsf{T}\vec{\pi}|\Big)\\ &= \mathrm{Inc}^1_\Pi(\mathcal{R}_1) + \mathrm{Inc}^1_\Pi(\mathcal{R}_2).\end{aligned}$$

□

Remark 4.6. *For $p > 1$ super-additivity can be violated.*

If we add consistent knowledge about propositional variables that were not contained in our knowledge base before, we should expect that the knowledge base does not become more inconsistent. This intuition is captured by *weak independence* (Thimm 2013).

Proposition 4.7 (Weak Independence). *Let $\mathcal{R}_1, \mathcal{R}_2$ be knowledge bases over $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$. If $\mathcal{R}_2$ is consistent and $\mathrm{At}(\mathcal{R}_1) \cap \mathrm{At}(\mathcal{R}_2) = \emptyset$, then $\mathrm{Inc}^p_\Pi(\mathcal{R}_1) = \mathrm{Inc}^p_\Pi(\mathcal{R}_1 \cup \mathcal{R}_2)$.*

Proof. Let $\mathcal{P}_1$ be minimal violating for $\mathcal{R}_1$ and let $\mathcal{P}_2$ be a model of $\mathcal{R}_2$. Let $\Sigma_1 = \mathrm{At}(\mathcal{R}_1)$ and $\Sigma_2 = \Sigma \setminus \Sigma_1$. For $i = 1, 2$, let $C_i = \{c \in \mathcal{L}_\Sigma \mid c = \bigwedge_{\alpha\in\Sigma_i} \alpha^{b_\alpha}, b_\alpha \in \{0, 1\}\}$ denote the set of complete conjunctions over Σ_i, where $\alpha^0 = \overline{\alpha}$ and $\alpha^1 = \alpha$ for each atom $\alpha \in \Sigma$. Note that $\Sigma = \Sigma_1 \uplus \Sigma_2$, hence we can identify each interpretation $\omega \in \Omega$ with the (unique) complete conjunction c_1c_2 that is satisfied by ω, where $c_1 \in \Sigma_1, c_2 \in \Sigma_2$.

Now consider the function $\mathcal{P} : \Omega \to \mathbb{R}$ defined by $\mathcal{P}(c_1c_2) = (\sum_{\overline{c_2}\in\Sigma_2} \mathcal{P}_1(c_1\overline{c_2}))(\sum_{\overline{c_1}\in\Sigma_1} \mathcal{P}_2(\overline{c_1}c_2))$. As $\mathcal{P}_1$ and $\mathcal{P}_2$ are nonnegative, so is $\mathcal{P}$. Also, $\mathcal{P}(c_1c_2) \leq 1$ by normalization of $\mathcal{P}_1$ and $\mathcal{P}_2$. Furthermore,

$$\begin{aligned}\mathcal{P}(\Omega) &= \sum_{c_1\in\Sigma_1}\sum_{c_2\in\Sigma_2} \mathcal{P}(c_1c_2)\\ &= \underbrace{\sum_{c_1\in\Sigma_1}\Big(\sum_{\overline{c_2}\in\Sigma_2} \mathcal{P}_1(c_1\overline{c_2})\Big)}_{=1}\underbrace{\sum_{c_2\in\Sigma_2}\Big(\sum_{\overline{c_1}\in\Sigma_1} \mathcal{P}_2(\overline{c_1}c_2)\Big)}_{=1}\\ &= 1,\end{aligned}$$

hence $\mathcal{P} : \Omega \to [0, 1]$ is normalized and therefore a probabilistic interpretation of $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$.

A formula F over Σ_1 is satisfied by c_1c_2 if and only if F is satisfied by c_1. Therefore,

$$\begin{aligned}\mathcal{P}(F) &= \sum_{c_1\in\Sigma_1}\sum_{c_2\in\Sigma_2} \mathcal{P}(c_1c_2)\, 1_{\{F\}}(c_1)\\ &= \underbrace{\sum_{c_2\in\Sigma_2}\Big(\sum_{\overline{c_1}\in\Sigma_1} \mathcal{P}_2(\overline{c_1}c_2)\Big)}_{=1}\underbrace{\sum_{c_1\in\Sigma_1}\Big(\sum_{\overline{c_2}\in\Sigma_2} \mathcal{P}_1(c_1\overline{c_2})\, 1_{\{F\}}(c_1)\Big)}_{=\mathcal{P}_1(F)}\\ &= \mathcal{P}_1(F).\end{aligned}$$

Hence, $\mathcal{P}$ coincides with $\mathcal{P}_1$ for formulas over Σ_1 and analogously it follows that $\mathcal{P}$ coincides with $\mathcal{P}_2$ for formulas over Σ_2. Therefore, if we denote by $\vec{\pi}, \vec{\pi}_1, \vec{\pi}_2$ the probability vectors corresponding to $\mathcal{P}, \mathcal{P}_1, \mathcal{P}_2$, it holds $\vec{a}_c^\mathsf{T}\,\vec{\pi} = \vec{a}_c^\mathsf{T}\,\vec{\pi}_1$ for all $c \in \mathcal{R}_1$ and $\vec{a}_c^\mathsf{T}\,\vec{\pi} = \vec{a}_c^\mathsf{T}\,\vec{\pi}_2 = 0$ for all $c \in \mathcal{R}_2$. Hence, if we number the conditionals in $\mathcal{R}_1$ by $1, \ldots, m$ and the conditionals in $\mathcal{R}_2$ by $m + 1, \ldots, m + k$, we find

$$\begin{aligned}\min_{\vec{\pi}\in\mathbb{R}^n} \|A_{\mathcal{R}_1\cup\mathcal{R}_2}\vec{\pi}\|_p &\leq \sqrt[p]{\sum_{i=1}^{m+k} |\vec{a}_i^\mathsf{T}\vec{\pi}|^p}\\ &= \sqrt[p]{\sum_{i=1}^{m} |\vec{a}_i^\mathsf{T}\vec{\pi}_1|^p + 0} = \mathrm{Inc}^p_\Pi(\mathcal{R}).\end{aligned}$$

Hence, $\mathrm{Inc}^p_\Pi(\mathcal{R}_1 \cup \mathcal{R}_2) \leq \mathrm{Inc}^p_\Pi(\mathcal{R}_1)$. By monotonicity (Proposition 4.4), it also holds $\mathrm{Inc}^p_\Pi(\mathcal{R}_1) \leq \mathrm{Inc}^p_\Pi(\mathcal{R}_1 \cup \mathcal{R}_2)$ and therefore $\mathrm{Inc}^p_\Pi(\mathcal{R}_1) = \mathrm{Inc}^p_\Pi(\mathcal{R}_1 \cup \mathcal{R}_2)$. □

As a desirable property of inconsistency measures for probabilistic knowledge bases, Thimm (2009) considered *continuity*. The intuitive idea is that if we change the probabilities in a knowledge base slightly, the corresponding inconsistency value should change only slightly, too. Let $\mathcal{R}[\vec{\rho}] = \langle(\mathcal{G}_1 \,|\, \mathcal{F}_1)[\vec{\rho}_1], \ldots, (\mathcal{G}_m \,|\, \mathcal{F}_m)[\vec{\rho}_m]\rangle$ denote an ordered knowledge base with conditional probabilities $\vec{\rho} \in \mathbb{R}^m$.

Proposition 4.8 (Continuity). *If $\lim_{k\to\infty} \vec{\rho}_k = \vec{\rho}$, then $\lim_{k\to\infty} \mathrm{Inc}^p_\Pi(\mathcal{R}[\vec{\rho}_k]) = \mathrm{Inc}^p_\Pi(\mathcal{R}[\vec{\rho}])$.*

Proof. Let $\mathcal{R} = \mathcal{R}[\vec{\rho}]$ be a knowledge base and let $(\vec{\rho}_k)$ be a sequence in $\mathbb{R}^m$ such that $\lim_{k\to\infty} \vec{\rho}_k = \vec{\rho}$. Let $A_\mathcal{R}$ be the matrix with row vectors $\vec{a}_i^\mathsf{T}$ corresponding to $\mathcal{R}$. Let A_k denote the matrix with row vectors $\vec{a_{ki}}^\mathsf{T} = (1_{\{\mathcal{G}\mathcal{F}\}}(\omega_j) \cdot (1 - \vec{\rho}_{ki}) - 1_{\{\overline{\mathcal{G}}\mathcal{F}\}}(\omega_j) \cdot \vec{\rho}_{ki})_{1\leq j\leq n}$ corresponding to $\mathcal{R}[\vec{\rho}_k]$. The linearity of the limit implies that $\lim_{k\to\infty} \vec{a_{ki}}^\mathsf{T} = \vec{a_i}^\mathsf{T}$ and that $\lim_{k\to\infty} \vec{a_{ki}}^\mathsf{T}\vec{\pi} = \vec{a_i}^\mathsf{T}\vec{\pi}$ for all $\vec{\pi} \in \mathbb{R}^n$. Hence, $\lim_{k\to\infty} A_k = A_\mathcal{R}$ and $\lim_{k\to\infty} A_k\vec{\pi} = A_\mathcal{R}\vec{\pi}$. Finally, by continuity of the p-norm, it follows that $\lim_{k\to\infty} \|A_k\vec{\pi}\|_p = \|A_\mathcal{R}\vec{\pi}\|_p$. As $\vec{\pi} \in \mathbb{R}^n$ was arbitrary, it follows $\lim_{k\to\infty} \mathrm{Inc}^p_\Pi(\mathcal{R}[\vec{\rho}_k]) = \mathrm{Inc}^p_\Pi(\mathcal{R}[\vec{\rho}])$. □

5 Computing Minimal Violation Measures

In this section, we explore how Inc^p_Π can be computed and how its computational performance relates to other measures. In general, computing Inc^p_Π corresponds to a convex optimization problem. Various approved algorithms exist to solve such problems, see (Boyd and Vandenberghe 2004) for an overview. For $p = 1$ and $p = \infty$, Inc^p_Π can be computed by linear programs, as I show in the following. The key idea of introducing auxiliary variables stems from unconstrained norm optimization problems (Boyd and Vandenberghe 2004).

For $p = 1$, we introduce a new vector $\vec{y} \in \mathbb{R}^m$ containing an auxiliary variable $\vec{y}_i$ for each conditional. Basically, the auxiliary variables $\vec{y}_i$ measure the error for the i-th conditional with respect to the 1-norm.

Proposition 5.1. $\mathrm{Inc}^1_\Pi(\mathcal{R})$ *can be computed by solving the following linear program:*

$$\min_{(\vec{\pi},\vec{y})\in\mathbb{R}^{n+m}} \quad \sum_{i=1}^{m} \vec{y}_i \qquad (2)$$

subject to

$$\sum_{i=1}^{n} \vec{\pi}_i = 1$$
$$-\vec{y} \leq A_\mathcal{R}\vec{\pi} \leq \vec{y}$$
$$\vec{y} \geq 0$$
$$\vec{\pi} \geq 0.$$

Proof. Along the same line of reasoning as for Proposition 3.3, it follows that (2) has a unique solution. It remains to show that optimization problem (2) yields the same solution as optimization problem (1) for $p = 1$ and arbitrary $\mathcal{R}$ over $(\mathcal{L}_\Sigma|\mathcal{L}_\Sigma)$.

To begin with, let ϵ be an optimal solution of (1). Then there is a $\vec{\pi} \in \Pi_n$ such that $\epsilon = \|A_\mathcal{R}\vec{\pi}\|_1 = \sum_{i=1}^m |\vec{a}_i^\mathsf{T}\vec{\pi}|$. Let $\vec{y} \in \mathbb{R}^m$ be defined by $\vec{y}_i = |\vec{a}_i^\mathsf{T}\vec{\pi}|$. Then $\begin{pmatrix}\vec{\pi}\\ \vec{y}\end{pmatrix}$ is feasible for (2), hence its optimum is at most $\sum_{i=1}^m \vec{y}_i = \epsilon$.

Conversely, suppose that $\sum_{i=1}^m |\vec{y}_i|$ is optimal for (2) and that $\begin{pmatrix}\vec{\pi}\\ \vec{y}\end{pmatrix}$ is an optimal point. Then $\vec{\pi}$ is feasible for (1) and its optimum is at most $\|A_\mathcal{R}\vec{\pi}\|_1 = \sum_{i=1}^m |\vec{a}_i^\mathsf{T}\vec{\pi}| \leq \sum_{i=1}^m |\vec{y}_i|$.

Hence both optima have to be equal. □

For $p = \infty$, it suffices to introduce a single auxiliary variable $y \in \mathbb{R}$, because the maximum norm measures the maximum error with respect to all conditionals. In the following proposition, $y\,\vec{1}$ denotes multiplication of the vector $\vec{1} \in \mathbb{R}^m$ (that contains only ones) by the scalar y.

Proposition 5.2. $\mathrm{Inc}^\infty_\Pi(\mathcal{R})$ *can be computed by solving the following linear program:*

$$\min_{(\vec{\pi},y)\in\mathbb{R}^{n+1}} \quad y \qquad (3)$$

subject to

$$\sum_{i=1}^{n} \vec{\pi}_i = 1$$
$$-y\,\vec{1} \leq A_\mathcal{R}\vec{\pi} \leq y\,\vec{1}$$
$$y \geq 0$$
$$\vec{\pi} \geq 0.$$

The proof is analogous to the last one and therefore left out.

Performance Evaluation

To test the computational performance of different measures, I computed inconsistency values for three test sets each of which contained nine knowledge bases and compared the runtime results. In each test set, from one knowledge base to the next, the number of conditionals and the size of the language was increased to capture the sensitivity of different measures with respect to the problem size. From one test set to the next, the number of conflicts between variables was increased to capture the sensitivity with respect to stronger correlations between the variables. The test sets were build up inductively as described in the following.

Test Set	$\lvert\Sigma_i\rvert$	$\lvert\Omega_i\rvert$	$\lvert\mathcal{R}_i\rvert$	
A	$2i$	4^i		$3i$
B	$2i$	4^i	3 *for* $i = 1$,	$4i - 1$ *for* $i > 1$
C	$2i$	4^i	3 *for* $i = 1$,	$5i - 2$ *for* $i > 1$

Table 1: Number of propositional variables $|\Sigma_i|$, worlds $|\Omega_i|$ and conditionals $|\mathcal{R}_i|$ in test sets A, B and C. For instance, $|\mathcal{R}_1^A| = 3$, $|\mathcal{R}_9^C| = 5 \cdot 9 - 2 = 43$, $|\Omega_9^C| = 4^9 = 2^{18}$.

Test Set A: The first (inconsistent) knowledge base $\mathcal{R}_1^A$ contains three conditionals built up over two variables. Σ_i^A and $\mathcal{R}_i^A$ are obtained from Σ_{i-1}^A and $\mathcal{R}_{i-1}^A$ by adding a new copy of the variables in Σ_1^A and the conditionals in $\mathcal{R}_1^A$.

$$\Sigma_1^A = \{A_1, B_1\},$$
$$\mathcal{R}_1^A = \{(A_1)[0.8], (B_1)[0.6], (B_1 \mid A_1)[0.8]\},$$
$$\Sigma_i^A = \Sigma_{i-1}^A \cup \{A_i, B_i\}, i > 1,$$
$$\mathcal{R}_i^A = \mathcal{R}_{i-1}^A \cup \{(A_i)[0.8], (B_i)[0.6], (B_i \mid A_i)[0.8]\}, i > 1.$$

Test Set B: For $i > 1$ the knowledge bases $\mathcal{R}_i^B$ differ from $\mathcal{R}_i^A$ by a new conflicting conditional that connects B_i and A_{i-1}. In this way all variables in test set B are correlated, whereas $\{A_i, B_i\}$ and $\{A_j, B_j\}$ are independent for $i \neq j$ in test set A.

$$\Sigma_i^B = \Sigma_i^A, i \in \mathbb{N},$$
$$\mathcal{R}_1^B = \mathcal{R}_1^A,$$
$$\mathcal{R}_i^B = \mathcal{R}_i^A \cup \{(B_k \mid A_{k-1})[0.85] \mid 2 \leq k \leq i\}, i > 1.$$

Test Set C: Compared to test set B, test set C contains yet another conflicting conditional between consecutive sets.

$$\Sigma_i^C = \Sigma_i^A, i \in \mathbb{N},$$
$$\mathcal{R}_1^C = \mathcal{R}_1^A,$$
$$\mathcal{R}_i^C = \mathcal{R}_i^B \cup \{(A_i \mid B_{i-1})[0.1] \mid 2 \leq k \leq i\}, i > 1.$$

Table 1 summarizes the calculation rules for the size of the knowledge bases and the corresponding languages.

Implementations: I used LPSolve[1], which is based on the Simplex algorithm, to compute Inc^1_Π and Inc^∞_Π. To compare the performance to other measures, I applied three prototypical implementations of the drastic measure (Hunter and Konieczny 2006; Thimm 2013), the MI measure (Hunter and Konieczny 2008; Thimm 2013) that counts the number of minimal inconsistent sets and the minimal change measure from (Thimm 2009).

[1] http://lpsolve.sourceforge.net

The drastic measure was computed by Phase 1 of the Simplex algorithm, which basically searches for a feasible solution. State of the art solvers for the probabilistic satisfiability problem apply more sophisticated techniques to speed up the computation (Hansen and Perron 2008; Cozman and Ianni 2013), but these can be applied to compute Inc_Π^1 and Inc_Π^∞ as well. Therefore, Phase 1 of the Simplex algorithm seems to be a reasonable baseline in this test series.

To find all minimal inconsistent sets, increasing subsets of the knowledge base were generated in lexicographic order. Each inconsistent subset was stored. The lexicographic order guaranteed minimality and that no supersets of minimal inconsistent sets were considered as candidates. To check consistency, again Phase 1 of the Simplex algorithm was applied. There already exist very sophisticated algorithms for enumerating minimal inconsistent sets for classical logics (Konieczny and Roussel 2013), but these rely on the fact that classical knowledge bases can be transformed into a single formula in conjunctive normal form. This fact, however, does not transfer to probabilistic logics.

The minimal change measure from (Thimm 2009) was computed by an implementation in the expert system shell MECore[2] (Finthammer et al. 2009). It is based on JCobyla[3], a Java implementation of Cobyla that approximates the non-convex optimization problem by linear programs (Powell 1994). There already exists an implementation of the measure from (Thimm 2009) in the Tweety library[4], but it relies on external libraries and is difficult to set up. Therefore, it was not used. Other methods might perform better than Cobyla, but to the best of my knowledge there are no algorithms for constrained non-convex problems that can handle more than a few thousand optimization variables and the used knowledge bases induced up to 2^{18} variables.

Figure 1 visualizes the runtime results. Note that the y-axis ranges to 10 seconds for test set A and B, but to 20 seconds for test set C. Execution was canceled if algorithms required more than 5 minutes to compute an inconsistency value.

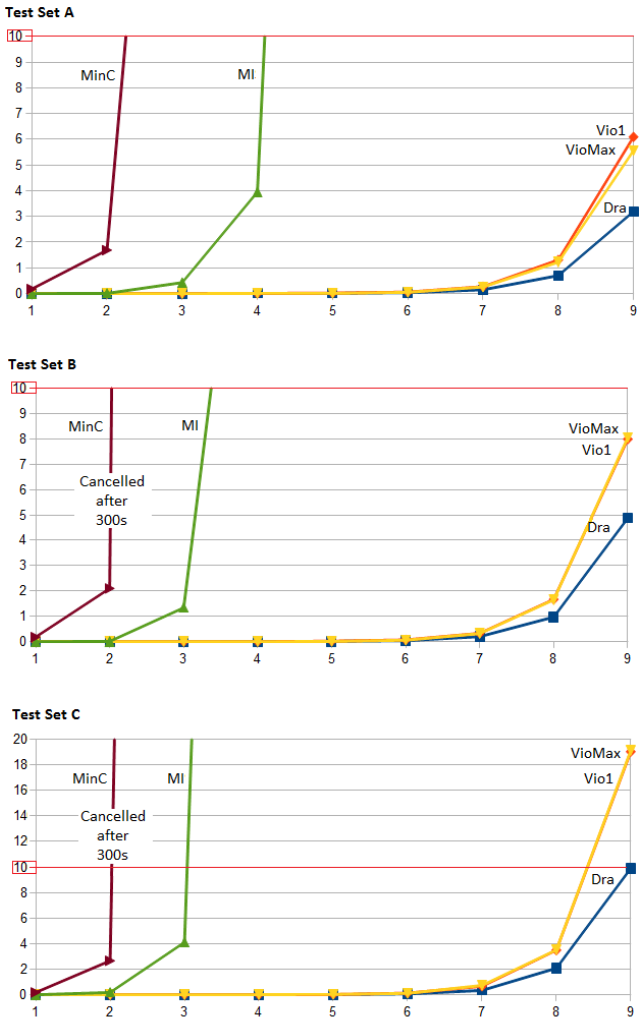


Figure 1: Runtime results in seconds (y-axis) for knowledge bases of increasing size (x-axis): Minimal change (MinC, red), minimal inconsistent sets (MI, green), Inc_Π^1 (Vio1, orange), Inc_Π^∞ (VioMax, yellow), drastic (Dra, blue) measure.

Interpretation of Performance Results: The runtime for each algorithm in each test set in Figure 1 grows exponentially from one knowledge base to the next, because the number of optimization variables corresponds to the number of worlds $|\Omega|$ that is exponential in the size of $|\Sigma|$. Note that $|\Omega|$ quadruples from one knowledge base to the next one. Inc_Π^1, Inc_Π^∞ and the drastic measure scale much better than the minimal change measure and the minimal inconsistent set measure. In the next subsection, I summarize some empirical estimates from (Boyd and Vandenberghe 2004; Matousek and Gärtner 2007) for typical optimization algorithms that support these results.

Also note that the minimal inconsistent set measure is more sensitive with respect to the size of the knowledge base. For example, when comparing test set A and B, the runtime for linear programs increases by at most 50 percent for all knowledge bases, whereas the runtime for the minimal inconsistent set measure triples from $\mathcal{R}_3^A$ to $\mathcal{R}_3^B$ and quadruples from $\mathcal{R}_4^A$ to $\mathcal{R}_4^B$. This observation corresponds to the fact that the number of possible candidates for minimal inconsistent subsets increases exponentially with the size of the knowledge base.

On Computational Complexity

Finding reasonable estimates for the performance of inconsistency measures is difficult, because it depends heavily on the applied algorithm and the structure of the knowledge base at hand. For example, one can construct examples where the Simplex algorithm requires an exponential number of iterations, even though it often performs faster than

[2]The measure was implemented by Sebastian Lewalder in his ongoing bachelor thesis.

[3]https://github.com/cureos/jcobyla

[4]http://tweety.sourceforge.net

algorithms with a polynomial runtime guarantee in practice (Matousek and Gärtner 2007). In this section, I summarize some empirical estimates for typical optimization algorithms to provide some intuition for the practical difficulty of different measures.

Empirical estimates: Practical experiments indicate that linear programs can be solved by the Simplex algorithm in about $2\overline{m}$ to $3\overline{m}$ iterations (Matousek and Gärtner 2007), where $\overline{m}$ is the number of constraints after transforming the problem to a standard form consisting of equations and non-negativity constraints. Each iteration usually causes costs in the order of $|\Omega|\,|\mathcal{R}|$, so that the overall cost can be regarded as in the order of $|\Omega|\,|\mathcal{R}|^2$.

For convex programs, Boyd and Vandenberghe (2004) estimate 10 to 100 iterations with cost $\max\{|\Omega|^3, |\Omega|^2\,|\mathcal{R}|\}$ per iteration when using interior-point methods. For the special case of $p = 2$, Inc^2_Π corresponds to a quadratic programming problem that can be solved in the order of $|\Omega|^2\,|\mathcal{R}|$ (Boyd and Vandenberghe 2004). Algorithms for non-convex programs can also be applied to convex programs, but usually are less efficient, because they rely on mechanisms to avoid (non-global) local minima. Therefore, one can expect that their computational cost is at least $|\Omega|^3$.

Practical Implications: As the number of worlds grows exponentially with the number of propositional variables, $|\Omega|$ usually dominates the size of the optimization problems. Therefore, one can expect to compute the drastic measure, Inc^1_Π and Inc^∞_Π with linear complexity, Inc^2_Π with quadratic complexity and Inc^p_Π for $p > 2$ with cubic complexity with respect to $|\Omega|$. For linear programs, several techniques can be applied to reduce the exponential influence of $|\Omega|$ (Jaumard, Hansen, and Poggi 1991; Hansen and Perron 2008; Cozman and Ianni 2013). The key idea is that only $\overline{m}$ columns of the $(\overline{m} \times \overline{n})$ constraint matrix have to be stored and in each iteration one column has to be switched. Selecting a good column from the $\overline{n} - \overline{m}$ possible candidates is the crucial problem.

Several measures rely on computing all minimal inconsistent sets (Hunter and Konieczny 2008; Thimm 2013). If $\#\mathit{MI}$ denotes the number of minimal inconsistent sets in a knowledge base, then we need to perform at least $\#\mathit{MI}$ satisfiability tests to verify their inconsistency. Based on the estimate $|\mathcal{R}|^2\,|\Omega|$ for probabilistic satisfiability tests, the cost for computing minimal consistent sets is at least $\#\mathit{MI}\,|\mathcal{R}|^2\,|\Omega|$.

As explained above, computing the non-convex problems corresponding to minimal change measures as introduced in (Thimm 2009) and (Picado-Muiño 2011) is at least as expensive as solving convex problems. Therefore, their computational cost can be expected to be greater than $|\Omega|^3$.

Table 2 summarizes the size of the optimization problems and the explained estimates for their practical computational cost. $2|\mathcal{R}|$ optimization variables for Inc^1_Π and Inc^∞_Π correspond to slack variables that have to be generated to transform the problem into a normal form for the Simplex algorithm. For the same reason the number of constraints is increased.

Note, however, that the mere number of optimization variables $\overline{n}$ and constraints $\overline{m}$ can be misleading when comparing two algorithms. For example, as Table 2 shows, the overall number of variables $\overline{n}$ is smaller for Inc^∞_Π than for Inc^1_Π. But whereas the single auxiliary variable for Inc^∞_Π appears in $2|\mathcal{R}|$ constraints, each of the $|\mathcal{R}|$ auxiliary variables for Inc^1_Π appear in only 2 constraints (c.f. Proposition 5.1 and 5.2). Therefore, the variables are looser correlated for Inc^1_Π and, indeed, Inc^∞_Π is slightly outperformed by Inc^1_Π in Figure 1 for test sets B and C.

Measure	$\overline{n}$	$\overline{m}$	Estimate
Drastic	$\lvert\Omega\rvert + \lvert\mathcal{R}\rvert$	$\lvert\mathcal{R}\rvert + 1$	$\sim \overline{n}\,\overline{m}^2$
Inc^∞_Π	$\lvert\Omega\rvert + 2\lvert\mathcal{R}\rvert + 1$	$2\lvert\mathcal{R}\rvert + 1$	$\sim \overline{n}\,\overline{m}^2$
Inc^1_Π	$\lvert\Omega\rvert + 3\lvert\mathcal{R}\rvert$	$2\lvert\mathcal{R}\rvert + 1$	$\sim \overline{n}\,\overline{m}^2$
Inc^2_Π	$\lvert\Omega\rvert$	1	$\sim \overline{n}^2\,\overline{m}$
$\mathrm{Inc}^p_\Pi, p > 2$	$\lvert\Omega\rvert$	1	$\sim \overline{n}^3$
MI	$\lvert\Omega\rvert + \lvert\mathcal{R}\rvert$	$\lvert\mathcal{R}\rvert + 1$	$> \#\mathit{MI}\,\overline{n}\,\overline{m}^2$
MinChange	$\lvert\Omega\rvert + \lvert\mathcal{R}\rvert$	$3\lvert\mathcal{R}\rvert + 1$	$> \overline{n}^3$

Table 2: Number of optimization variables $\overline{n}$, number of constraints $\overline{m}$ (without non-negativity constraints) and rough performance estimates for different measures. $\#\mathit{MI}$ denotes the number of minimal inconsistent sets. Performance estimates are based on empirical values from (Boyd and Vandenberghe 2004; Matousek and Gärtner 2007) for typical optimization algorithms.

6 Conclusion and Future Work

As shown in Section 4, minimal violation measures satisfy many desirable properties that have been considered in the literature (Knight 2003; Hunter and Konieczny 2006; Thimm 2013). In contrast to the naive drastic measure or measures based on minimal inconsistent sets (Hunter and Konieczny 2008; Thimm 2013), minimal violation measures take the numerical structure of probabilistic knowledge bases into account and in this way provide a more fine-grained insight into inconsistencies, as demonstrated in Examples 3.4 and 3.5. When applied as a guide to repair knowledge bases, they can never yield better theoretical guarantees than measures based on minimal changes, because these yield the best repair by definition (Thimm 2009; Picado-Muiño 2011). However, the lack of efficient algorithms for constrained non-convex problems prevents us from computing minimal change measures for larger knowledge bases. As explained in Section 5, minimal violation measures possess better computational properties. In particular, Inc^1_Π and Inc^∞_Π stand out among conventional measures for probabilistic logics, because they provide similar good performance guarantees like the naive drastic measure, while measuring the degree of inconsistency continuously.

One long term goal of this work is resolving inconsistencies in probabilistic knowledge bases with methods that are better guided than purely heuristic approaches and that provide better computational properties than minimal change approaches. A first algorithm based on minimal violation measures might iterate over the conditionals, check whether an adaptation yields an improvement in the inconsistency

value and select the best adaptation as illustrated in Example 3.5. To speed up repeated computations of Inc_{Π}^{1} and $\mathrm{Inc}_{\Pi}^{\infty}$, column generation techniques can be applied (Hansen and Perron 2008). In future work such an algorithm shall be implemented and evaluated in the expert system shell MECore (Finthammer et al. 2009).

It might also be interesting to ask whether the minimal violating interpretations can be used to infer reasonable information from contradictory knowledge bases. The minimal violating interpretations yield a compact and convex set just like the models of consistent knowledge bases. Therefore, computing probabilities from minimal violating interpretations of inconsistent knowledge bases can be performed like computing probabilities from the models of consistent knowledge bases. However, the question remains whether the computed probabilities yield reasonable results and whether they can be justified by some rationales.

For $p = 1$, $p = 2$ and $p = \infty$, Java-implementations of Inc_{Π}^{p} are made available in the Tweety library[5] (they can be found in the package net.sf.tweety.logics.pcl.analysis. Note that LPSolve[6] has to be installed and configured properly to use Inc_{Π}^{1} and $\mathrm{Inc}_{\Pi}^{\infty}$).

References

Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. New York, NY, USA: Cambridge University Press.

Cozman, F., and Ianni, L. 2013. Probabilistic satisfiability and coherence checking through integer programming. In van der Gaag, L. C., ed., *Proc. Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU 2013*, volume 7958 of *Lecture Notes in Computer Science*, 145–156. Springer Berlin Heidelberg.

Doder, D.; Raskovic, M.; Markovic, Z.; and Ognjanovic, Z. 2010. Measures of inconsistency and defaults. *International Journal of Approximate Reasoning* 51(7):832–845.

Finthammer, M.; Beierle, C.; Berger, B.; and Kern-Isberner, G. 2009. Probabilistic reasoning at optimum entropy with the MEcore system. In Lane, H., and Guesgen, H., eds., *Proc. International FLAIRS Conference, FLAIRS 2009*, 535–540. AAAI Press, Menlo Park, California.

Finthammer, M.; Kern-Isberner, G.; and Ritterskamp, M. 2007. Resolving inconsistencies in probabilistic knowledge bases. In Hertzberg, J.; Beetz, M.; and Englert, R., eds., *Proc. German Conference on Artificial Intelligence, KI 2007*, volume 4667 of *Lecture Notes in Computer Science*, 114–128. Springer.

Georgakopoulos, G. F.; Kavvadias, D. J.; and Papadimitriou, C. H. 1988. Probabilistic satisfiability. *Journal of Complexity* 4(1):1–11.

Grant, J., and Hunter, A. 2008. Analysing inconsistent first-order knowledgebases. *Artificial Intelligence* 172(8-9):1064–1093.

Hansen, P., and Perron, S. 2008. Merging the local and global approaches to probabilistic satisfiability. *International Journal of Approximate Reasoning* 47(2):125 – 140.

Hunter, A., and Konieczny, S. 2006. Shapley inconsistency values. In Doherty, P.; Mylopoulos, J.; and Welty, C. A., eds., *Proc. Principles of Knowledge Representation and Reasoning, KR 2006*, 249–259. AAAI Press.

Hunter, A., and Konieczny, S. 2008. Measuring inconsistency through minimal inconsistent sets. In Brewka, G., and Lang, J., eds., *Proc. Principles of Knowledge Representation and Reasoning, KR 2008*, 358–366. AAAI Press.

Jaumard, B.; Hansen, P.; and Poggi, M. 1991. Column generation methods for probabilistic logic. *ORSA - Journal on Computing* 3(2):135–148.

Kern-Isberner, G. 2001. *Conditionals in Nonmonotonic Reasoning and Belief Revision*. Number 2087 in Lecture Notes in Computer Sience. Berlin: Springer-Verlag.

Knight, K. 2002. Measuring inconsistency. *Journal of Philosophical Logic* 31(1):77–98.

Knight, K. M. 2003. Two information measures for inconsistent sets. *Journal of Logic, Language and Information* 12(2):227–248.

Konieczny, S., and Roussel, S. 2013. A reasoning platform based on the MI shapley inconsistency value. In van der Gaag, L. C., ed., *Proc. Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU 2013*, volume 7958 of *Lecture Notes in Computer Science*, 315–327. Springer.

Lukasiewicz, T. 1999. Probabilistic deduction with conditional constraints over basic events. *Journal of Artificial Intelligence Research* 10:380–391.

Matousek, J., and Gärtner, B. 2007. *Understanding and Using Linear Programming*. Universitext (1979). Springer.

Nilsson, N. J. 1986. Probabilistic logic. *Artificial Intelligence* 28:71–88.

Paris, J. B., and Vencovská, A. 1989. On the applicability of maximum entropy to inexact reasoning. *International Journal of Approximate Reasoning* 3(1):1–34.

Picado-Muiño, D. 2011. Measuring and repairing inconsistency in probabilistic knowledge bases. *International Journal of Approximate Reasoning* 52(6):828–840.

Powell, M. 1994. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in Optimization and Numerical Analysis*, volume 275 of *Mathematics and Its Applications*. Springer Netherlands. 51–67.

Rödder, W., and Xu, L. 1999. Entropy-driven inference and inconsistency. In *Proc. Artificial Intelligence and Statistics*, volume 99, 272–277.

Thimm, M. 2009. Measuring inconsistency in probabilistic knowledge bases. In Bilmes, J., and Ng, A. Y., eds., *Proc. Uncertainty in Artificial Intelligence, UAI 2009*, 530–537. AUAI Press.

Thimm, M. 2013. Inconsistency measures for probabilistic logics. *Artificial Intelligence* 197:1–24.

[5] http://tweety.sourceforge.net

[6] http://lpsolve.sourceforge.net



Short Papers

Axioms .2 and .4 as Interaction Axioms

Guillaume Aucher
University of Rennes 1
INRIA
Rennes, France
guillaume.aucher@irisa.fr

Abstract

In epistemic logic, some axioms dealing with the notion of knowledge are rather convoluted and it is difficult to give them an intuitive interpretation, even if some of them, like .2 and .4, are considered by some epistemic logicians to be key axioms. We show that they can be characterized in terms of understandable interaction axioms relating knowledge and belief. In order to show it, we first present a theory dealing with the characterization of axioms in terms of interaction axioms in modal logic. We then apply the main results and methods of this theory to obtain our results related to epistemic logic.

1 Introduction

The development of modern epistemic logic can be viewed as an attempt to elucidate the nature of the interaction between knowledge and belief by means of formal and logical methods. On the basis of a semantics very close to the Kripke semantics of modal logic, Hintikka and subsequent philosophers and logicians tried to formulate explicit principles governing and relating expressions of the form "a knows that φ" (subsequently formalized as $K\varphi$) and "a believes that φ" (subsequently formalized as $B\varphi$), where a is a human agent and φ is a proposition. In other words, they sought to determine 'the' logic of knowledge and belief. This quest was grounded in the observation that our intuitions of these epistemic notions comply to some systematic reasoning properties, and was driven by the attempt to better *understand* and *elucidate* them (Lenzen 1978, p. 15). For example, the interaction axioms $K\varphi \rightarrow B\varphi$ and $B\varphi \rightarrow KB\varphi$ are often considered to be intuitive principles: if agent a knows φ then (s)he also believes φ, or if agent a believes φ, then (s)he knows that (s)he believes φ.

In computer science, the logic of knowledge is usually considered to be S5, which is the logic S4 composed of the axioms $K\varphi \rightarrow \varphi$ and $K\varphi \rightarrow KK\varphi$ to which is added the axiom 5: $\neg K\varphi \rightarrow K\neg K\varphi$. This last axiom is validated in situations where the agent cannot have mistaken beliefs. For this very reason, it has been attacked by various philosophers because it cannot hold in general. Dropping this axiom 5, we obtain a rich variety of weaker logics of knowledge which have been proposed and examined by logicians (Lenzen 1979), such as S4.2, S4.3, S4.3.2 and S4.4. Even if these logics are characterized by axioms which are rather intricate, some of them have been proclaimed by some epistemic logicians as key axioms characterizing the notion of knowledge. For example, Lenzen claimed that "[t]here is strong evidence in favor of the assumption that S4.2 is the logic of knowledge" (Lenzen 1979, p. 33), where the axiom .2 is $\neg K\neg K\varphi \rightarrow K\neg K\neg\varphi$. Likewise, Kutschera argues for S4.4 as the logic of knowledge, where the axiom .4 is $(\varphi \wedge \langle K\rangle K\varphi) \rightarrow K\varphi$ (Kutschera 1976). As one can easily observe, it is difficult to provide these axioms with a natural and easily understandable reading. In fact, Lenzen derived his axiom .2 from a set of interaction axioms relating knowledge and belief.

To better grasp the intuitions underlying these intricate axioms, we show that axioms .2 and .4 can be characterized equivalently in terms of interaction axioms relating knowledge and belief. In order to do so, we first need to explain what we mean by "interaction axiom" and what we mean by "characterizing an axiom in terms of interaction axioms". This will lead us to develop a meta-theory of modal logic dealing with these notions. Then, we will apply the general results of this theory to the specific case of epistemic logic. Note that the problems that we address in this article have never been addressed in the logical literature, neither for multi-modal logics nor for combinations of modal logics (Marx and Venema 1997; Gabbay et al. 1998).

2 Towards a Theory of Interaction Axioms

In this section, we start by recalling the basics of modal logic. Then, we present our primitive meta-theory of modal logic dealing with interaction axioms.

2.1 Modal Logic

Syntax. In the rest of the article, Φ is a set of propositional letters and $\mathbb{A} \subseteq \{1, 2\}$. We define the *modal language* $\mathcal{L}_{\mathbb{A}}$ by the following BNF grammar:

$$\mathcal{L}_{\mathbb{A}} \ : \ \varphi \ ::= \ p \ \mid \ \neg\varphi \ \mid \ \varphi \wedge \varphi \ \mid \ [i]\,\varphi$$

where p ranges over Φ and i ranges over $\mathbb{A}$. The formula $\top$ is an abbreviation for $p \vee \neg p$ (for some fixed $p \in \Phi$), the

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

formula $\bot$ is an abbreviation for $\neg\top$ and $\langle i\rangle\varphi$ an abbreviation for $\neg[i]\neg\varphi$. An occurrence of a proposition letter p is a *positive* occurrence if it is in the scope of an even number of negation signs $\neg$. It is *positive in* p if all occurrences of p in φ are positive. If $\Gamma := \{\varphi_1, \ldots, \varphi_n\} \subseteq \mathcal{L}_{\mathbb{A}}$, then $\bigwedge\Gamma$ is an abbreviation for $\varphi_1 \wedge \ldots \wedge \varphi_n$.

A *(modal) logic* L for the modal language $\mathcal{L}_{\mathbb{A}}$ is a set of formulas of $\mathcal{L}_{\mathbb{A}}$ called *theorems* which contains all propositional tautologies and which is closed under modus ponens, that is, if $\varphi \in \mathsf{L}$ and $\varphi \to \psi \in \mathsf{L}$, then $\psi \in \mathsf{L}$, and closed under uniform substitution, that is, if φ belongs to L then so do all of its substitution instances (see (Blackburn, de Rijke, and Venema 2001, Def. 1.18) for the definition of a substitution instance).

A modal logic is usually defined by a set of *inference rules* and of formulas called *axioms*. A formula is a theorem of the modal logic if it can be derived by successively applying (some of) the inference rules to (some of) the axioms. We are interested here in *normal modal logics*. These modal logics contain the axiom schema $([i](\varphi \to \psi) \wedge [i]\varphi) \to [i]\psi$, and the inference rule of necessitation: from $\varphi \in \mathsf{L}$, infer $[i]\varphi \in \mathsf{L}$, for all $i \in \mathbb{A}$. Let $A \subseteq \mathcal{L}_{\mathbb{A}}$. A modal logic for $\mathcal{L}_{\mathbb{A}}$ *generated* by the set A is the smallest normal modal logic for $\mathcal{L}_{\mathbb{A}}$ containing A. In that case, the formulas of A are called *axioms*.

If L and L' are two sets of formulas of $\mathcal{L}_{\mathbb{A}}$ (possibly logics), we denote by $\mathsf{L} + \mathsf{L}'$ the modal logic for $\mathcal{L}_{\mathbb{A}}$ generated by $\mathsf{L} \cup \mathsf{L}'$ (it is also called the *fusion* of L and L' (Marx and Venema 1997; Gabbay et al. 1998)). If x is a formula of $\mathcal{L}_{\mathbb{A}}$, then $\mathsf{L} + \mathsf{x}$ abusively denotes $\mathsf{L} + \{\mathsf{x}\}$. Note that $\mathsf{L} + \mathsf{L}'$ may be different from $\mathsf{L} \cup \mathsf{L}'$ in general, because $\mathsf{L} \cup \mathsf{L}'$ may not be closed under modus ponens or uniform substitution.

Let $\mathsf{x} \in \mathcal{L}_{\mathbb{A}}$ and let $X, X' \subseteq \mathcal{L}_{\mathbb{A}}$. We say that x is *derivable from* X *in* L when $\mathsf{x} \in \mathsf{L} + X$ and in that case we write $X \vdash_{\mathsf{L}} \mathsf{x}$. We also write $X \vdash_{\mathsf{L}} X'$ when $X \vdash_{\mathsf{L}} \mathsf{x}'$ for all $\mathsf{x}' \in X'$, and $X >_{\mathsf{L}} X'$ when it holds that $X \vdash_{\mathsf{L}} X'$ but it does not hold that $X' \vdash_{\mathsf{L}} X$.

Kripke Semantics. The Kripke semantics will be used only in the proof of Theorem 2.2. A *(bi-modal) Kripke model* $\mathcal{M}$ is a tuple $\mathcal{M} = (W, R_1, R_2, V)$ where W is a non-empty set of possible worlds, $R_1, R_2 \in 2^{W\times W}$ are binary relations over W called *accessibility relations*, and $V : \Phi \to 2^W$ is called a *valuation* and assigns to each propositional letter $p \in \Phi$ a subset of W. We often denote by $R_i(w)$ the set $R_i(w) := \{v \in W \mid wR_iv\}$ and we abusively write $w \in \mathcal{M}$ when $w \in W$.

Let $\varphi \in \mathcal{L}_{\mathbb{A}}$, let $\mathcal{M}$ be a Kripke model and let $w \in \mathcal{M}$. The *satisfaction relation* $\mathcal{M}, w \models \varphi$ is defined inductively as follows:

$$\begin{array}{lll}
\mathcal{M}, w \models p & \text{iff} & w \in V(p)\\
\mathcal{M}, w \models \varphi \wedge \varphi' & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \varphi'\\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \text{not } \mathcal{M}, w \models \varphi\\
\mathcal{M}, w \models [i]\varphi & \text{iff} & \text{for all } v \in R_i(w), \mathcal{M}, v \models \varphi
\end{array}$$

If $\Gamma \subseteq \mathcal{L}_{\mathbb{A}}$, then we write $\mathcal{M}, w \models \Gamma$ when $\mathcal{M}, w \models \varphi$ for all $\varphi \in \Gamma$.

2.2 Interaction Axioms and Characterization of Axioms

In the sequel, L_1 and L_2 are two modal logics for $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively, and $\mathsf{L}_{1,2}$ is a modal logic for $\mathcal{L}_{1,2}$.

Definition 2.1 (Interaction Axioms). A *set of interaction axioms* w.r.t. a pair of logics $(\mathsf{L}_1, \mathsf{L}_2)$ is a finite set of formulas $\Gamma \subseteq \mathcal{L}_{1,2}$ for which there is no $\chi \in \mathcal{L}_1 \cup \mathcal{L}_2$ such that

$$\chi \leftrightarrow \bigwedge\Gamma \in \mathsf{L}_1 + \mathsf{L}_2.$$

In the sequel, x is a formula of $\mathcal{L}_1$ and Γ is a set of interaction axioms w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$.

Definition 2.2 (Characterization of an Axiom). We say that x is *characterized* by the set of interaction axioms Γ w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$ when

$$\mathsf{L}_1 + \mathsf{x} = (\mathsf{L}_1 + \mathsf{L}_2 + \Gamma) \cap \mathcal{L}_1. \tag{1}$$

Moreover, x is *conservatively characterized* by Γ w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$ when the set of interaction axioms Γ satisfies the following condition as well:

$$\mathsf{L}_2 = (\mathsf{L}_1 + \mathsf{L}_2 + \Gamma) \cap \mathcal{L}_2. \tag{2}$$

Definition 2.2 tells us that an axiom x is characterized by a set of interaction axioms Γ if, when we add the interaction axioms to the base logics, we derive exactly the theorems for the language $\mathcal{L}_1$ obtained by only adding axiom x to L_1, and nothing else. For the case of strong characterization, by adding these interaction axioms Γ, we do not even obtain new theorems for the language $\mathcal{L}_2$ as 'side effects', the theorems for this language just remain the same as initially.

Proposition 2.1. • *If x is conservatively characterized by a set of interaction axioms w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$, then $\mathsf{x} \notin \mathsf{L}_1$.*

• *Conversely, if $\mathsf{x} \notin \mathsf{L}_1$ and Γ is a set of formulas of $\mathcal{L}_{1,2}$ such that Equation (1) holds, then Γ is a set of interaction axioms w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$.*

Proof. It follows easily from Definitions 2.1 and 2.2. □

Finally, we define a notion of minimality among the sets of interaction axioms characterizing an axiom x.

Definition 2.3 (Minimal Characterization). The axiom x is *minimally characterized* by the set of interaction axioms Γ w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$ when x is characterized by Γ w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$ and there is no set of interaction axioms Γ' such that $\Gamma >_{\mathsf{L}_1+\mathsf{L}_2} \Gamma'$ and x is still characterized by Γ' w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$.

2.3 Definability of Modalities and Characterization of Axioms

The definability of modalities in terms of other modalities is studied from a theoretical point of view in (Halpern, Samet, and Segev 2009b). This study is subsequently applied to epistemic logic in (Halpern, Samet, and Segev 2009a). Three notions of definability emerge from this work: explicit definability, implicit definability and reducibility. It has been proven that, for modal logic, explicit definability coincides with the conjunction of implicit definability and reducibility (unlike first-order logic, where the notion of explicit definability coincides with implicit definability only). In this

article, we are interested only in the notion of *explicit* definability, which is also used in (Lenzen 1979).

Definition 2.4 (Explicit Definability of a Modality). Let $\{i,j\} = \{1,2\}$. The modality $\langle i\rangle$ is *explicitly defined* in the logic $\mathsf{L}_{i,j}$ in terms of the modality $\langle j\rangle$ by a formula $\mathsf{def}_i(p) \in \mathcal{L}_j$ if, and only if,

$$\langle i\rangle p \leftrightarrow \mathsf{def}_i(p) \in \mathsf{L}_{i,j}. \qquad (\mathsf{Def}\ \langle i\rangle)$$

The following key theorem will play an important role in the last section.

Theorem 2.2. *Assume that $\langle 2\rangle$ is explicitly defined in $\mathsf{L}_1 + \mathsf{L}_2 + \Gamma$ in terms of $\langle 1\rangle$ by a formula $\mathsf{def}_2(p) \in \mathcal{L}_1$ positive in p. Then, the following are equivalent:*

- *x is characterized by Γ w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$;*
- *$\mathsf{L}_1 + \mathsf{L}_2 + \Gamma = \mathsf{L}_1 + \mathsf{x} + \{\langle 2\rangle p \leftrightarrow \mathsf{def}_2(p)\}$.*

Moreover, assume that $\langle 1\rangle$ is also explicitly defined in $\mathsf{L}_1 + \mathsf{L}_2 + \Gamma$ in terms of $\langle 2\rangle$ by a formula $\mathsf{def}_1(p) \in \mathcal{L}_2$ positive in p. Then, the following are equivalent:

- *x is conservatively characterized by Γ w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$;*
- *$\mathsf{L}_1 + \mathsf{L}_2 + \Gamma = \mathsf{L}_1 + \mathsf{x} + \{\langle 2\rangle p \leftrightarrow \mathsf{def}_2(p)\}$ and $\mathsf{L}_1 + \mathsf{L}_2 + \Gamma = \mathsf{L}_2 + \{\langle 1\rangle p \leftrightarrow \mathsf{def}_1(p)\}$.*

Finally, in both cases, the axiom x is (conservatively) characterized by Γ w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$ if, and only if, it is minimally *(conservatively) characterized by Γ w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$.*

Proof. The proof of the second part of the theorem is similar to the proof of the first part. So, we only prove the first part. Assume that x is characterized by Γ w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$. Then, $\mathsf{L}_1 + \mathsf{x} = (\mathsf{L}_1 + \mathsf{L}_2 + \Gamma) \cap \mathcal{L}_1$, and therefore $\mathsf{L}_1 + \mathsf{x} \subseteq \mathsf{L}_1 + \mathsf{L}_2 + \Gamma$. Moreover, $\langle 2\rangle p \leftrightarrow \mathsf{def}_2(p) \in \mathsf{L}_1 + \mathsf{L}_2 + \Gamma$ by assumption. Thus, $\mathsf{L}_1 + \mathsf{x} + \{\langle 2\rangle p \leftrightarrow \mathsf{def}_2(p)\} \subseteq \mathsf{L}_1 + \mathsf{L}_2 + \Gamma$. Now, we prove the converse inclusion. Assume towards a contradiction that there is $\varphi \in \mathsf{L}_1 + \mathsf{L}_2 + \Gamma$ such that $\varphi \notin \mathsf{L}_1 + \mathsf{x} + \{\langle 2\rangle p \leftrightarrow \mathsf{def}_2(p)\}$. Then, there is $\varphi' \in \mathcal{L}_1$ such that $\varphi \leftrightarrow \varphi' \in \mathsf{L}_1 + \mathsf{L}_2 + \Gamma$, because $\langle 2\rangle$ is explicitly definable in terms of $\langle 1\rangle$ in $\mathsf{L}_1 + \mathsf{L}_2 + \Gamma$. Then, $\varphi' \in (\mathsf{L}_1 + \mathsf{L}_2 + \Gamma) \cap \mathcal{L}_1$, *i.e.*, $\varphi' \in \mathsf{L}_1 + \mathsf{x}$. Then, by performing the inverse translation that we followed to obtain φ' from φ, we conclude that $\varphi \in \mathsf{L}_1 + \mathsf{x} + \{\langle 2\rangle p \leftrightarrow \mathsf{def}_2(p)\}$. This is impossible, and therefore $\mathsf{L}_1 + \mathsf{L}_2 + \Gamma = \mathsf{L}_1 + \mathsf{x} + \{\langle 2\rangle p \leftrightarrow \mathsf{def}_2(p)\}$.

Now, assume that $\mathsf{L}_1 + \mathsf{L}_2 + \Gamma = \mathsf{L}_1 + \mathsf{x} + \{\langle 2\rangle p \leftrightarrow \mathsf{def}_2(p)\}$. We are going to prove that $\mathsf{L}_1 + \mathsf{x} = (\mathsf{L}_1 + \mathsf{L}_2 + \Gamma) \cap \mathcal{L}_1$. The right to left inclusion is immediate, because $\mathsf{x} \in \mathsf{L}_1 + \mathsf{L}_2 + \Gamma$ by assumption. Now, we prove that $(\mathsf{L}_1 + \mathsf{L}_2 + \Gamma) \cap \mathcal{L}_1 \subseteq \mathsf{L}_1 + \mathsf{x}$, *i.e.*, for all Kripke models $\mathcal{M}$, for all $w \in \mathcal{M}$, if $\mathcal{M}, w \models \mathsf{L}_1 + \mathsf{x}$, then $\mathcal{M}, w \models (\mathsf{L}_1 + \mathsf{L}_2 + \Gamma) \cap \mathcal{L}_1$. In order to do so, we are going to build a Kripke model $(\mathcal{M}', w')$ such that $(\mathcal{M}, w)$ and $(\mathcal{M}', w')$ are bisimilar w.r.t. the modality $\langle 1\rangle$ and such that $\mathcal{M}', w' \models \mathsf{L}_1 + \mathsf{x} + \{\langle 2\rangle p \leftrightarrow \mathsf{def}_2(p)\}$ $(*)$, that is, $\mathcal{M}', w' \models \mathsf{L}_1 + \mathsf{L}_2 + \Gamma$ (recall the assumption). This will prove the second inclusion. If $(\mathcal{M}, w) = (W, R_1, R_2, V, w)$, then we define the (pointed) Kripke model $(\mathcal{M}', w') := (W, R_1, R_2', V, w)$, where R_2' is defined as follows. First, we define the (pointed) Kripke model $(\mathcal{M}'', w) := (W, R_1, R_2, V'', w)$ by setting V'' such that for all $q \neq p$, $V''(q) = V(q)$ and such that $V''(p) = \{v\}$. Then, for all $u, v \in W$, we set $uR_2'v$ in $\mathcal{M}'$ if, and only if, $\mathcal{M}'', u \models \mathsf{def}_2(p)$. Then, using the fact that $\mathsf{def}_2(p)$ is positive in p, one can easily show that $(*)$ holds. This proves the second inclusion.

Finally, we prove the last part of the theorem. Assume towards a contradiction that x is characterized by the set of interaction axioms Γ w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$ and that there is a set of interaction axioms Γ' such that $\Gamma >_{\mathsf{L}_1+\mathsf{L}_2} \Gamma'$ and such that x is also characterized by Γ' w.r.t. $(\mathsf{L}_1, \mathsf{L}_2)$. Because $\Gamma >_{\mathsf{L}_1+\mathsf{L}_2} \Gamma'$, we should have that $\mathsf{L}_1 + \mathsf{L}_2 + \Gamma \subset \mathsf{L}_1 + \mathsf{L}_2 + \Gamma'$. However, since x is characterized by Γ and Γ', we should also have that $\mathsf{L}_1 + \mathsf{L}_2 + \Gamma = \mathsf{L}_1 + \mathsf{L}_2 + \Gamma' = \mathsf{L}_1 + \mathsf{x} + \{\langle 2\rangle p \leftrightarrow \mathsf{def}_2(p)\}$ by the result of the first part of the theorem. This is impossible. □

3 Epistemic Logic and Interaction Axioms

We introduce the basics of epistemic logic (see (Fagin et al. 1995; Meyer and van der Hoek 1995) for more details).

3.1 Logics of Knowledge and Belief

We define the epistemic-doxastic language $\mathcal{L}_{KB}$ inductively as follows:

$$\mathcal{L}_{KB} \ : \ \varphi \ ::= \ p \ \mid \ \neg\varphi \ \mid \ \varphi \wedge \varphi \ \mid \ B\varphi \ \mid \ K\varphi$$

where p ranges over Φ. The language $\mathcal{L}_K$ is the language $\mathcal{L}_{KB}$ without the belief operator B, and the language $\mathcal{L}_B$ is the language $\mathcal{L}_{KB}$ without the knowledge operator K. The formula $B\varphi$ reads as 'the agent Believes φ' and $K\varphi$ reads as 'the agent Knows φ'. Their dual operators $\langle B\rangle\varphi$ and $\langle K\rangle\varphi$ are abbreviations of $\neg B\neg\varphi$ and $\neg K\neg\varphi$ respectively. Below, we give a list of axioms that will be used in the rest of the article.

D	$: K\varphi \to \langle K\rangle\varphi$	$\mathsf{4}$	$: K\varphi \to KK\varphi$
$\mathsf{5}$	$: \neg K\varphi \to K\neg K\varphi$	T	$: K\varphi \to \varphi$
$.\mathsf{2}$	$: \langle K\rangle K\varphi \to K\langle K\rangle\varphi$	$.\mathsf{4}$	$: (\varphi \wedge \langle K\rangle K\varphi) \to K\varphi$.

The logic $\mathsf{KD45}_B$ is the modal logic for $\mathcal{L}_B$ generated by the set of axioms $\{\mathsf{D}, \mathsf{4}, \mathsf{5}\}$. For any $\mathsf{x} \in \{.\mathsf{2}, .\mathsf{4}\}$, the logic $\mathsf{S4.x}_K$ is the modal logic for $\mathcal{L}_K$ generated by the set of axioms $\{\mathsf{T}, \mathsf{4}, \mathsf{x}\}$. We have the following relationship between these logics:

$$\mathsf{S4}_K \subset \mathsf{S4.2}_K \subset \mathsf{S4.4}_K \subset \mathsf{S5}_K.$$

3.2 Interaction Axioms for Epistemic Logic

The following interaction axioms were suggested in (Hintikka 1962):

$$K\varphi \to B\varphi \qquad (\mathrm{I}_1)$$
$$B\varphi \to KB\varphi \qquad (\mathrm{I}_2)$$

Axiom I_1 is a cornerstone of epistemic logic. Just as axiom T, it follows from the classical analysis of knowledge of Plato presented in the Theaetethus. Axiom I_2 highlights the fact that the agent has "privileged access" to his doxastic state.

$$B\varphi \to BK\varphi \qquad (\mathrm{I}_3)$$

Axiom I_3 above was suggested in (Lenzen 1978). It characterizes a notion of belief corresponding to some sort of conviction or certainty. This kind of belief is therefore different

from the notion of *weak* belief which can be represented by a probability superior to 0.5, like my belief that "it will rain tomorrow".

The last interaction axiom we will consider is in fact a definition of knowledge in terms of belief:

$$K\varphi \leftrightarrow (\varphi \wedge B\varphi) \qquad (\mathrm{I}_4)$$

This list of interaction axioms is incomplete, see (Aucher 2014) for more information about interaction axioms and axioms of epistemic logic.

4 Applying our Theory to Epistemic Logic

In (Halpern, Samet, and Segev 2009a), only the interaction axioms I_1 and I_2 suggested in (Hintikka 1962) are considered. In this section, we also add the interaction axiom I_3. Lenzen is the first to note that the belief modality can be defined in terms of knowledge if we adopt $\{I_1, I_2, I_3\}$ as interaction axioms.

Theorem 4.1. *(Lenzen 1979)*

The belief modality B is explicitly defined in the logic $\mathsf{L} = \mathsf{S4}_K + \mathsf{KD45}_B + \{I_1, I_2, I_3\}$ by the following definition:

$$B\varphi \leftrightarrow \langle K\rangle K\varphi \in \mathsf{L} \qquad (\mathsf{Def\ B})$$

Consequently, the belief modality B is also defined by (Def B) *in any logic containing* L.

This result can be contrasted with Theorem 4.8 of (Halpern, Samet, and Segev 2009a), from which it follows that the belief modality can*not* be explicitly defined in the logic $(\mathsf{S4.x})_K + (\mathsf{KD45})_B + \{I_1, I_2\}$, for any $\mathsf{x} \in \{.2, .4\}$. On the other hand, as Theorem 4.2 below shows, knowledge can be defined in terms of belief only if the logic of knowledge is S4.4, but not if the logic of knowledge is S4 or S4.2.

Theorem 4.2. *(Aucher 2014)*

- *The knowledge modality K is explicitly defined in the logic $\mathsf{L.4} := \mathsf{S4.4}_K + \mathsf{KD45}_B + \{I_1, I_2, I_3\}$ by the following definition:*

$$K\varphi \leftrightarrow \varphi \wedge B\varphi \in \mathsf{L.4} \qquad (\mathsf{Def\ K})$$

- *The knowledge modality K cannot be explicitly defined in the logic $\mathsf{S4.2}_K + \mathsf{KD45}_B + \{I_1, I_2, I_3\}$.*

Again, this result can be contrasted with Theorem 4.1 of (Halpern, Samet, and Segev 2009a), from which it follows that the knowledge modality can*not* be explicitly defined in the logic $(\mathsf{S4.4})_K + (\mathsf{KD45})_B + \{I_1, I_2\}$. We see that in both cases the increase in expressivity due to the addition of the interaction axiom I_3 plays an important role in bridging the gap between belief and knowledge.

Theorem 4.3.

- *The axiom .2 is characterized w.r.t. the pair $(\mathsf{S4}_K, \mathsf{KD45}_B)$ by the set of interaction axioms $\{I_1, I_2, I_3\}$.*
- *The axiom .4 is conservatively characterized w.r.t. the pair $(\mathsf{S4}_K, \mathsf{KD45}_B)$ by the interaction axiom I_4.*

Proof. It follows from a direct application of Theorem 2.2 to the results of (Lenzen 1979), namely the facts that $\mathsf{S4}_K + \mathsf{KD45}_B + \{I_1, I_2, I_3\} = \mathsf{S4.2}_K + \{B\varphi \leftrightarrow \langle K\rangle K\varphi\}$ and $\mathsf{S4}_K + \mathsf{KD45}_B + \{I_4\} = \mathsf{KD45}_B + \{I_4\} = \mathsf{S4.4}_K + \{B\varphi \leftrightarrow \langle K\rangle K\varphi\}$. □

5 Conclusion

Theorem 4.3 tells us that if we use S4.4 as the logic of knowledge (and KD45 as the logic of belief), then we implicitly assume that knowledge is in fact true belief (a rather strong assumption for knowledge). Although it was acknowledged by all epistemic logicians that axiom .4 characterized knowledge as true belief, this could never be justified and explained rigorously. We claim that our meta-theory of modal logic fills this conceptual gap. Likewise, if we use S4.2 as the logic of knowledge (and KD45 as the logic for belief), then Theorem 4.3 tells us that, by doing so, we are only assuming that the agent knows his beliefs and disbeliefs and that his beliefs are in fact certainties, convictions, and not simply weak beliefs.

Overall, our meta-theory of modal logic enables to carry out a rigorous and fine-grained analysis of the intuitive assumptions underlying the logics of knowledge between S4 and S5. This theory provides a meaningful logical foundation for these analysis and can serve as a means to justify and explain the intuitive arguments employed.

References

Aucher, G. 2014. *Interdisciplinary Works in Logic, Epistemology, Psychology and Linguistics*, volume 3 of *Logic, Argumentation and Reasoning*. Springer. chapter Principles of Knowledge, Belief and Conditional Belief.

Blackburn, P.; de Rijke, M.; and Venema, Y. 2001. *Modal Logic*, volume 53 of *Cambridge Tracts in Computer Science*. Cambridge University Press.

Fagin, R.; Halpern, J.; Moses, Y.; and Vardi, M. 1995. *Reasoning about knowledge*. MIT Press.

Gabbay, D.; Kurucz, A.; Wolter, F.; and Zakharyaschev, M. 1998. *Multi-dimensional modal logic: Theory and application*, volume 148 of *Studies in logic and the foundations of mathematics*. Elsevier.

Halpern, J. Y.; Samet, D.; and Segev, E. 2009a. Defining knowledge in terms of belief: the modal logic perspective. *The Review of Symbolic Logic* 2:469–487.

Halpern, J. Y.; Samet, D.; and Segev, E. 2009b. On definability in multimodal logic. *The Review of Symbolic Logic* 2:451–468.

Hintikka, J. 1962. *Knowledge and Belief, An Introduction to the Logic of the Two Notions*. Ithaca and London: Cornell University Press.

Kutschera, F. v. 1976. *Einführung in die intensional Semantik*. Berlin: W. de Gruyter.

Lenzen, W. 1978. *Recent Work in Epistemic Logic*. Acta Philosophica Fennica 30. North Holland Publishing Company.

Lenzen, W. 1979. Epistemologische betractungen zu [S4;S5]. *Erkenntnis* 14:33–56.

Marx, M., and Venema, Y. 1997. *Multi-dimensional modal logic*. Springer.

Meyer, J.-J. C., and van der Hoek, W. 1995. *Epistemic Logic for AI and Computer Science*. Cambridge: Cambridge University Press.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Stable Models of Multi-Valued Formulas: Partial versus Total Functions

Michael Bartholomew and Joohyung Lee
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ, USA
{*mjbartho,joolee*}*@asu.edu*

Abstract

Recent extensions of the stable model semantics that allow *intensional* functions—functions that can be specified by logic programs using other functions and predicates—can be divided into two groups. One group defines a stable model in terms of minimality on the values of partial functions, and the other defines it in terms of uniqueness on the values of total functions. We show that, in the context of multi-valued formulas, these two different approaches can be reduced to each other, and further, each of them can be viewed in terms of propositional formulas under the stable model semantics. Based on these results, we present a prototype implementation of different versions of functional stable model semantics by using existing answer set solvers.

Introduction

The original stable model semantics (Gelfond and Lifschitz 1988) and many extensions have been restricted to Herbrand models, where the role of functions is quite limited. Recently a few extensions of the stable model semantics were proposed to allow *intensional* functions—functions that can be specified by logic programs using other functions and predicates. Despite the different forms in which these semantics were defined, they can be essentially divided into two groups. One group defines a stable model in terms of "minimality on the values of partial functions" (Cabalar 2011; Balduccini 2013) and the other defines it in terms of "uniqueness on the values of total functions" (Lifschitz 2012; Bartholomew and Lee 2012). While it is known that they coincide on some class of formulas (Bartholomew and Lee 2013), it does not look obvious if they can be reduced to each other in full generality. Further, it is not obvious how mathematical results in answer set programming that have been established in the absence of intensional functions would carry over to these extensions.

In this note, we address such issues in the context of multi-valued formulas—a simple extension of standard propositional formulas where atoms can express functions mapping to finite domains. The convenience of using multi-valued formulas for knowledge representation is demonstrated in the context of nonmonotonic causal theories and action languages $\mathcal{C}+$ (Giunchiglia et al. 2004) and $\mathcal{BC}$ (Lee, Lifschitz, and Yang 2013). For this paper, multi-valued formulas serve as a simple but useful special case of first-order formulas to compare different extensions of the functional stable model semantics.

The total function based stable model semantics for multi-valued formulas is defined in (Bartholomew and Lee 2012). Here, following (Cabalar 2011; Balduccini 2013), we define the partial function based stable model semantics, which we call the CB-stable model semantics. This is essentially a generalization of the semantics from (Balduccini 2013). We show that multi-valued formulas under these functional stable model semantics can be viewed in terms of propositional formulas under the stable model semantics, with a slight difference to each other. This finding reveals the close relationship between the functional stable model semantics and their relationships to the propositional stable model semantics, and allows us to easily relate the mathematical results established for propositional formulas, such as the theorem on strong equivalence (Lifschitz, Pearce, and Valverde 2001) and the splitting theorem (Ferraris et al. 2009), to multi-valued formulas. In (Lee, Lifschitz, and Yang 2013), action language $\mathcal{BC}$ is defined by a translation to multi-valued formulas and by a translation to logic programs. The equivalence between the two translations follows from our finding.

Given that both versions of the functional stable model semantics can be reduced to the propositional stable model semantics, one may wonder about the relationship between the two versions of the functional stable model semantics. Interestingly, we show that the functional stable model semantics that is based on partial functions can be fully embedded into the one that is based on total functions.

These results provide a way to implement the functional stable model semantics using existing ASP solvers. We present system MVSM based on this idea. The system is essentially a preprocessor to F2LP (Lee and Palla 2009), which in turn is a preprocessor to the ASP grounder GRINGO.

Multi-Valued Formulas under the Stable Model Semantics

Review: Stable Models of Multi-Valued Formulas

A (multi-valued) signature is a finite set σ of symbols called *constants*, along with a finite set $Dom(c)$ of symbols that is disjoint from σ and contains at least two elements, assigned

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

to each constant c. We call $Dom(c)$ the *domain* of c. A *multi-valued atom* of σ is $\bot$, or an expression of the form $c=v$ ("the value of c is v") where $c \in \sigma$ and $v \in Dom(c)$. A *(multi-valued) formula* of σ is a propositional combination of multi-valued atoms.

A *(multi-valued) interpretation* of σ is a function that maps every element of σ to an element in its domain. A multi-valued interpretation I *satisfies* an atom $c=v$ (symbolically, $I \models c=v$) if $I(c) = v$. The satisfaction relation is extended from atoms to arbitrary formulas according to the usual truth tables for the propositional connectives. We often identify an interpretation with the set of atoms of σ that are satisfied by I. We say that I is a *(multi-valued) model* of F if it satisfies F.

We understand an expression of the form $c = d$, where both c and d are constants, as an abbreviation for the formula

$$\bigvee_{v \in Dom(c) \cap Dom(d)} (c=v \wedge d=v). \tag{1}$$

Let F be a multi-valued formula of signature σ, and let I be a multi-valued interpretation of σ. The *reduct* of F relative to I (denoted $F^{\underline{I}}$) is the formula obtained from F by replacing each (maximal) subformula that is not satisfied by I with $\bot$. We call I a *multi-valued stable model* of F if I is the *only* multi-valued interpretation of σ that satisfies $F^{\underline{I}}$.

Example 1 *Take $\sigma = \{c\}$ and $Dom(c) = \{1,2,3\}$, and let F_1 be $c = 1 \vee \neg(c = 1)$ and let I_i ($i = 1,2,3$) be the interpretation that maps c to i. All three interpretations satisfy F_1, but I_1 is the only stable model of F_1: the reduct $F_1^{\underline{I_1}}$ is $c = 1 \vee \bot$, and I_1 is the only model of the reduct; the reduct of F_1 relative to other interpretations is $\bot \vee \neg\bot$, which does not have a unique model.*

If we conjoin $c = 2$ with F_1, we can check that the only stable model is $c=2$, which illustrates the nonmonotonicity of the semantics.

As shown in Example 1, formulas of the form $F \vee \neg F$ under the stable model semantics are useful for representing the concept of defaults involving functions. We abbreviate $F \vee \neg F$ as $\langle F \rangle$. For example, F_1 above can be written as $\langle c = 1 \rangle$.

Reducing Multi-Valued SM to Propositional SM

In this section we show that the multi-valued stable model semantics can be viewed as a special case of the propositional stable model semantics. Let σ be a multi-valued signature, and let σ^{prop} be the propositional signature consisting of all propositional atoms $c = v$ where $c \in \sigma$ and $v \in Dom(c)$. For example, for σ in Example 1, σ^{prop} is the set $\{c=1, c=2, c=3\}$, where each element is understood as a propositional atom. We identify a multi-valued interpretation of σ with the corresponding set of propositional atoms from σ^{prop}. It is clear that a multi-valued interpretation I of signature σ satisfies a multi-valued formula F iff I satisfies F when F is viewed as a propositional formula of signature σ^{prop}. Also, it is not difficult to show that multi-valued formulas can be turned into standard propositional formulas having the same *models*. Less obvious is whether such a translation exists while keeping same *stable models*. Theorem 1 below shows such a translation.

Given a multi-valued signature σ, by UC_σ ("Uniqueness Constraint") we denote the conjunction of

$$\bigwedge_{v \neq w \mid v,w \in Dom(c)} \neg(c = v \wedge c = w) \tag{2}$$

for all $c \in \sigma$, and by EC_σ ("Existence Constraint") we denote the conjunction of

$$\neg\neg \bigvee_{v \in Dom(c)} c = v\,, \tag{3}$$

for all $c \in \sigma$. By UEC_σ we denote the conjunction of (2) and (3) for all $c \in \sigma$. For instance, in Example 1, UEC_σ is

$$\neg(c=1 \wedge c=2) \wedge \neg(c=2 \wedge c=3) \wedge \neg(c=1 \wedge c=3) \wedge \neg\neg(c=1 \vee c=2 \vee c=3)\,.$$

Theorem 1 *Let F be a multi-valued formula of signature σ, which can be also viewed as a propositional formula of signature σ^{prop}.*

(a) If an interpretation I of σ is a multi-valued stable model of F, then I can be viewed as an interpretation of σ^{prop} that is a propositional stable model of $F \wedge UEC_\sigma$ in the sense of (Ferraris 2005).

(b) If an interpretation I of σ^{prop} is a propositional stable model of $F \wedge UEC_\sigma$ in the sense of (Ferraris 2005), then I can be viewed as an interpretation of σ that is a multi-valued stable model of F.

Note that the presence of $\neg\neg$ in (3) is essential for Theorem 1 to be valid. For instance, consider the signature containing only one constant d whose domain is $\{1,2\}$ and F to be $\top$. F has no multi-valued stable models, but $F \wedge \neg(d=1 \wedge d=2) \wedge (d=1 \vee d=2)$ has two propositional stable models: $\{d=1\}$ and $\{d=2\}$.

Multi-Valued Formulas under the CB-Stable Model Semantics

CB-Stable Models of Multi-Valued Formulas

In this section we introduce a variant of the stable model semantics in the previous section, which we call the CB-stable model semantics. Unlike the previous section, this section allows interpretations to be partially defined. That is, some constants might not be mapped to any values. By *complete* interpretations, we mean a special case of partial interpretations where all constants are defined, which can be identified with "total" interpretations in the previous section.

We consider the same syntax of a multi-valued formula as in the previous section. As with total interpretations, a partial interpretation I *satisfies* an atom $c = v$ if $I(c)$ is defined and is mapped to v. This implies that an interpretation that is undefined on c does not satisfy any atom of the form $c = w$ for any $w \in Dom(c)$. As before, it is convenient to identify a partial interpretation I with the set of atoms of σ that are satisfied by this interpretation. For instance, an interpretation of $\sigma = \{c\}$ which is undefined on c is identified with the empty set. Again, the satisfaction relation is extended from atoms to arbitrary formulas according to the usual truth tables

for the propositional connectives. We call I a *model* of F if it satisfies F.

The reduct $F^{\underline{I}}$ is defined to be the same as before. We say that a partial interpretation I is a *CB-stable model* of F if I satisfies F and no proper subset J of I satisfies $F^{\underline{I}}$.

Example 1 Continued *Under the CB-stable model semantics, $\langle c=1\rangle$ does not mean that c is mapped to 1 by default. Instead, it means that c can be mapped to 1 or undefined. As before, the reduct $F_1^{\underline{I_1}}$ relative to $I_1 = \{c=1\}$ is $c=1 \vee \bot$, and I_1 is the minimal model of the reduct.*[1] *Further, for I_0 that leaves c undefined, the reduct $F_1^{\underline{I_0}}$ is $\bot \vee \neg\bot$, and I_0 is the minimal model of the reduct.*

This difference in understanding $\langle F\rangle$ tells us that multi-valued stable models are more convenient for representing the commonsense law of inertia. For instance,

$$Loc_0 = Kitchen \rightarrow \langle Loc_1 = Kitchen\rangle$$

represents under the multi-valued stable model semantics that the location does not change by default, but under the CB-stable model semantics, the location may become unknown as well.

Reducing CB-Stable Models to Propositional SM

Similar to Theorem 1, the following theorem tells us that the CB-stable models of a multi-valued formula can be identified with the stable models of a propositional formula. The only difference is that we impose UC_σ in place of UEC_σ.

Theorem 2 *Let F be a multi-valued formula of signature σ, which can be also viewed as a propositional formula of signature σ^{prop}.*

(a) If a partial interpretation I of σ is a CB-stable model of F, then I can be viewed as an interpretation of σ^{prop} that is a propositional stable model of $F \wedge UC_\sigma$ (in the sense of (Ferraris 2005)).

(b) If an interpretation I of σ^{prop} is a propositional stable model of $F \wedge UC_\sigma$ (in the sense of (Ferraris 2005)), then I can be viewed as a partial interpretation of σ that is a CB-stable model of F.

Reducing CB Semantics to Multi-Valued SM

Reducing Multi-Valued SM to CB-Stable Models

The following corollary immediately follows from Theorems 1 and 2. It tells us that the multi-valued stable model semantics can be fully embedded into the CB-stable model semantics.

Corollary 1 *For any multi-valued formula F of signature σ and any partial interpretation I, we have that I is a multi-valued stable model of F iff I is a CB-stable model of $F \wedge EC_\sigma$.*

Unlike the way we treat $c = d$ as an abbreviation of (1), in (Balduccini 2013), it was called a *t-atom*, for which the notion of satisfaction was defined directly: *I satisfies $c = d$* if I is defined on both c and d, and maps them to the same value. This is essentially equivalent to the way we understand $c = d$ as shorthand for formula (1).[2]

Since I satisfies $c = c$ iff I is defined on c, the assertion in Corollary 1 remains valid when we replace EC_σ in the statement with $\neg\neg\Big(\bigwedge_{c\in\sigma} c = c\Big)$.

Reducing CB-Stable Models to Multi-Valued SM

Any multi-valued stable model of a formula is a CB-stable model, but not vice versa because an incomplete partial interpretation has no direct counterpart as a total interpretation. It may not look obvious how the CB-stable model semantics (based on partial functions) can be reduced to the multi-valued stable model semantics (based on total functions). Nevertheless, we show that it is possible.

Let σ be a multi-valued signature, and let σ^{none} be the signature that is the same as σ except that the domain of each constant has an additional new value NONE. Given a partial interpretation I of σ, by I^{none} we denote an interpretation of σ^{none} that agrees with I on all defined constants, and maps undefined constants to NONE. Recall that expression $\langle F\rangle$ stands for the formula $F \vee \neg F$.

Theorem 3 *Let F be a multi-valued formula of signature σ.*

(a) If an interpretation I of σ is a CB-stable model of F, then I^{none} is a stable model of $F \wedge \bigwedge_{c\in\sigma}\langle c=\text{NONE}\rangle$.

(b) If an interpretation J of σ^{none} is a stable model of $F \wedge \bigwedge_{c\in\sigma}\langle c = \text{NONE}\rangle$ then $J = I^{none}$ for some CB-stable model I of F.

For instance, in Example 1, the CB-stable models of F_1 are in a 1-1 correspondence with the stable models of $F_1 \wedge \langle c=\text{NONE}\rangle$.

System MVSM

System `mvsm`[3] is a prototype implementation of multi-valued formulas under the stable model semantics. In fact, it is a script that invokes several software, such as MVPF2LPCOMPILER, F2LP, GRINGO, CLASPD, and AS2TRANSITION. MVPF2LPCOMPILER is an implementation of the translations in Theorem 1 and Theorem 2, which translates multi-valued formulas under the stable model semantics into standard propositional formulas under the stable model semantics. As the theorems show, the translations are very similar, and the user can choose which translation to use. F2LP then transforms the propositional formula into an ASP program in the input language of GRINGO v3. AS2TRANSITION takes the output of CLASPD and outputs propositional atoms in the form of multi-valued atoms. The composition of these software is depicted in Figure 1.

Shown below is a description of the blocks world domain in the language of MVSM assuming the multi-valued stable model semantics. The syntax of declarations follows the one in the input language of the Causal Calculator v2.[4] Compared

[1] Minimality is understood in terms of set inclusion.

[2] In a sense, our treatment is more general because it allows the domains of c and d to be different.

[3] http://sourceforge.net/projects/aspmt/

[4] http://www.cs.utexas.edu/~tag/cc/

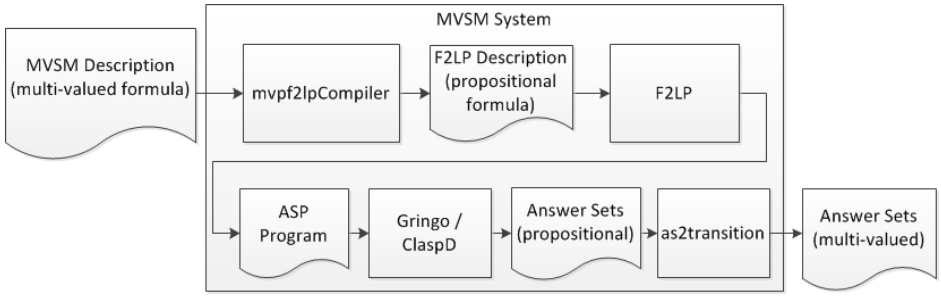


Figure 1: Architecture of MVSM

to the usual ASP encoding, explicit declaration of sorts and type checking help reduce the programmer's mistakes. The inertia and exogeneity assumptions in the last three rules have simple reading, once we understand $\langle F \rangle$ as representing defaults ($\{\ldots\}$ was used in place of $\langle \ldots \rangle$). There is no need to use strong negation.

```
% File 'bw': The blocks world

:- sorts
  step; astep; location >> block.

:- objects
  0..maxstep              :: step;
  0..maxstep-1            :: astep;
  1..6                    :: block;
  table                   :: location.

:- variables
  ST                      :: step;
  T                       :: astep;
  Bool                    :: boolean;
  B,B1                    :: block;
  L                       :: location.

:- constants
  loc(block,step)                 :: location;
  move(block,location,astep)      :: boolean.

% two blocks can't be on the same block at the same time
<- loc(B1,ST)=B & loc(B2,ST)=B & B1!=B2.

% effect of moving a block
loc(B,T+1)=L <- move(B,L,T).

% a block can be moved only when it is clear
<- move(B,L,T) & loc(B1,T)=B.

% a block can't be moved onto a block that is being
% moved also
<- move(B,B1,T) & move(B1,L,T).

% initial location is exogenous
{loc(B,0)=L}.

% actions are exogenous
{move(B,L,T)=Bool}.

% fluents are inertial
{loc(B,T+1)=L} <- loc(B,T)=L.
```

Acknowledgements: This work was partially supported by NSF under Grant IIS-1319794 and by the South Korea IT R&D program MKE/KIAT 2010-TD-300404-001.

References

Balduccini, M. 2013. ASP with non-Herbrand partial functions: a language and system for practical use. *TPLP* 13(4-5):547–561.

Bartholomew, M., and Lee, J. 2012. Stable models of formulas with intensional functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2–12.

Bartholomew, M., and Lee, J. 2013. On the stable model semantics for intensional functions. *TPLP* 13(4-5):863–876.

Cabalar, P. 2011. Functional answer set programming. *TPLP* 11(2-3):203–233.

Ferraris, P.; Lee, J.; Lifschitz, V.; and Palla, R. 2009. Symmetric splitting in the general theory of stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 797–803. AAAI Press.

Ferraris, P. 2005. Answer sets for propositional theories. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 119–131.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R., and Bowen, K., eds., *Proceedings of International Logic Programming Conference and Symposium*, 1070–1080. MIT Press.

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2004. Nonmonotonic causal theories. *Artificial Intelligence* 153(1–2):49–104.

Lee, J., and Palla, R. 2009. System F2LP – computing answer sets of first-order formulas. In *Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 515–521.

Lee, J.; Lifschitz, V.; and Yang, F. 2013. Action language $\mathcal{BC}$: Preliminary report. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic* 2:526–541.

Lifschitz, V. 2012. Logic programs with intensional functions. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 24–31.

Rough Set Semantics for Identity on the Web

Wouter Beek and **Stefan Schlobach** and **Frank van Harmelen**
Vrije Universiteit Amsterdam
De Boelelaan 1081a
1081HV Amsterdam
The Netherlands

Abstract

Identity relations are at the foundation of many logic-based knowledge representations. We argue that the traditional notion of equality, is unsuited for many realistic knowledge representation settings. The classical interpretation of equality is too strong when the equality statements are re-used outside their original context. On the Semantic Web, equality statements are used to interlink multiple descriptions of the same object, using `owl:sameAs` assertions. And indeed, many practical uses of `owl:sameAs` are known to violate the formal Leibniz-style semantics.

We provide a more flexible semantics to identity by assigning meaning to the subrelations of an identity relation in terms of the predicates that are used in a knowledge-base. Using those indiscernability-predicates, we define upper and lower approximations of equality in the style of rought-set theory, resulting in a quality-measure for identity relations.

1 Introduction

Identity relations are a cornerstone of logic-based knowledge representation. They allow to state and relate properties of an object using multiple names for that object, and conversely, they allow to infer that different names actually refer to the same object.

Identity relations are at the foundation of the Linked Open Data initiative and the Semantic Web (SW) in general. The SW consists of sets of assertions that are published on the Web by different authors operating in different contexts, often using different names for the same object. Identity relations allow the interlinking of these multiple descriptions of the same thing.

Identity is often understood as the sharing of all properties between two objects with different names (principle of indiscernibility). In the SW this traditional notion of identity is expressed by the `owl:sameAs` property.
According to the traditional semantics of the identity relation, identical terms can be replaced for one another in all non-modal contexts *salva veritate*. Practical uses of `owl:sameAs` are known to violate this strict condition (Halpin et al. 2010a; 2010b).

On the SW, identity assertions are extra strong because of the Open World Assumption. Stating that two objects are the same implies that from now on no new property can be stated about only one of those objects Moreover, whether or not two objects share the absence of a property cannot be concluded based on the absence of a property assertion.

Improving on the existing semantics of identity, we have the following research goals:

1. In an identity relation the pairs all look the same. We want to characterize subrelations of an identity relation in terms of the predicates that are important in a particular context.
2. Based on an existing identity relation we want to give semantically motivated suggestions for extending or limiting the identity relation.
3. We want to assess the quality of an identity relation based on the consistency with which it is applied to the data.

2 Related work

Existing research suggests six different solutions for the problem of identity on the SW.

[1] Introduce weaker versions of `owl:sameAs` (Halpin et al. 2010a; McCusker and McGuinness 2010). Candidates for replacement are the SKOS concepts `skos:related` and `skos:exactMatch`. The former is not transitive, thereby limiting the possibilities for reasoning. The latter is transitive, but can only be used in certain contexts. It is not defined in what contexts it can be used.

[2] Restrict the applicability of identity relations to specific contexts. In terms of SW technology, identities are expected to hold within a named graph or within a namespace, but not necessarily outside of it (Halpin et al. 2010a). (de Melo 2013) has successfully used the Unique Names Assumption within namespaces in order to identify many (arguably) spurious identity statements.

[3] Introduce additional vocabulary that does not weaken but extend the existing identity relation. (Halpin et al. 2010a) mention an explicit distinction that can be made between mentioning a term and using a term. Other possible extensions of `owl:sameAs` take the Fuzzyness and/or uncertainty of identity statements into account.

[4] Use domain-specific identity relations (McCusker and McGuinness 2010). Such domain-specific links are only locally valid, thereby limiting knowledge reuse.

[5] Change the modeling practice (Halpin et al. 2010a; Ding et al. 2010a). Introducing checks on editing operations

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

violates one of the fundamental underpinnings of the SW: that anybody is allowed to say anything about anything (Antoniou et al. 2012).

(Ding et al. 2010b) who show that network analysis of the occurrence of `owl:sameAs` in datasets can provide insights into the ways in which identity is used. These latter endeavors have not yet been related to the semantics of identity.

3 Approach

First we give a short outline of our approach and then we provide a more detailed description of the individual steps.

Outline of the approach

We start by assuming that we are given an identity relation $\approx$. We will then reinterpret this relation as an indiscernibility relation relative to different sets of predicates. Pairs that have the same indiscernibility predicates are simi-discernible, i.e.: they discern resources based on the same criteria. Simi-discernibility is an equivalence relation which partitions all pairs and thus also the identity relation $\approx$. The members of the indiscernibility partition have a certain overlap with the original identity relation. The overlap between an indiscernibility subset and the identity relation is called an *identity subrelation*. Each identity subrelation is characterized in terms of predicates from the domain vocabulary. Different forms of identity can therefore be distinguished and meaningfully described. Based on whether there is a complete or a partial overlap between the simi-discernible partition members and the identity subrelations, these partition members belong either to the lower ($\underline{\approx}$) or to the higher approximation ($\overline{\approx}$) of $\approx$. Besides setting a lower and a higher bound to the identity relation, we can also calculate the quality of the identity relation and the precision of each identity subrelation.

Preliminaries

G denotes an RDF graph. It consists of a set of ground binary predicates $p(s, o)$, called "triples" in SW jargon, and often written as $\langle s, p, o\rangle$. These triples form a graph with all subjects s and objects o as nodes, and each assertion $p(s, o)$ corresponds to a directed edge labelled p between s and o.

We identify subsets of RDF terms based on their positional occurrence in triples in G: S_G, P_G, and O_G denote the subject, predicate and object terms in G respectively.

The interpretation I maps RDF terms onto resources, and triples onto truth values. The extension function Ext maps resources onto pairs of resources. $I(\langle s, p, o\rangle)$ is true iff $\langle I(s), I(o)\rangle \in Ext(I(p))$.

Reinterpreting identity as indiscernibility

We start by assuming that we are given an identity relation $\approx$, which partitions the subject terms S_G according to equation 1.

$$[x]_{\approx} = \{y \in S_G \mid x \approx y\} \tag{1}$$

Identity can be defined as the smallest equivalence relation, i.e. the most fine-grained partition of S_G. For reasoning purposes, the fact that $\approx$ is an equivalence relation is important, allowing symmetrical and transitive inferences. Identity implies indiscernibility with respect to all properties.

We can generalize the notion of indiscernibility by parameterizing the set of properties with respect to which indiscernibility is determined. According to this generalization, resources x and y are indiscernible with respect to a set of properties $PO \subseteq P_G \times O_G$ iff $\forall po \in PO(po(x) \leftrightarrow po(y))$ is the case. Every indiscernibility relation is also an equivalence relation, although not necessarily the smallest one. Moreover, every indiscernibility relation defined over the domain S_G is also an identity relation, but over a different domain.

We now reinterpret the identity relation $\approx$ as if it were an indiscernibility relation whose set of properties PO is implicit. Based on the extensional specification of the identity relation, we make the set of properties with respect to which it is indiscernible explicit. Definition 1 makes explicit the properties relative to which the terms x_i are indiscernibile.

Definition 1 (Indiscernibility properties).

$$\mathbb{IND}\text{-}\mathbb{PO}_{\approx}(\{x_1, \ldots, x_n\}) = \{\langle p, o\rangle \in P_G \times O_G \mid \bigwedge_{1 \le i \le n} \exists p_i \in [p]_{\approx}, \exists o_i \in [o]_{\approx}(\langle x_i, p_i, o_i\rangle \in G)\}$$

Notice that in definition 1 we close both the predicate terms p and the object terms o under identity. Performing these closures is important in order to identify the relevant indiscernibility properties.

In the above, we were interested in the properties that resources share with one other. But we are also interested in the predicates that are shared by a set of resources. This amounts to a simple abstraction of definition 1, equating the sets of objects (closed under identity) and only returning the set of shared RDF predicate terms (see definition 2).

Definition 2 (Indiscernibility predicates).

$$\mathbb{IND}\text{-}\mathbb{P}_{\approx}(\{x_1, \ldots, x_n\}) = \{p \in P_G \mid \exists_{p_1,\ldots,p_n \in [p]_{\approx}}([\{o \in O_G \mid \langle x_1, p_1, o\rangle\}]_{\approx} = \ldots = [\{o \in O_G \mid \langle x_n, p_n, o\rangle\}]_{\approx})\}$$

Discerning the same

In the previous section we saw that resources are *indiscernible* with respect to PO iff they cannot be told apart in a language that only contains the properties denoted by PO (the so-called indiscernibility properties):

In the same vein, and builing upon definition 2, we say that two pairs of resources are *simi-discernible* iff their indiscernibility predicates $P \subseteq P_G$ are the same.

When we look at the pairs that constitute (the extension of) an identity relation, all identity assertions look the same. But when we take the considerations of the previous section into account, we see that within a given identity relation there are pairs that assert indiscernibility based on different domain predicates. Stating this formally, simi-discernibility is an equivalence relation on pairs of resources, which induces a partition of the Cartesian product of the domain. Definition 3 makes this concrete in terms of the earlier definitions.

Definition 3 (Simi-discernibility relation).

$$\equiv_{\mathbb{IND}\text{-}\mathbb{P}_{\approx}} = \{\langle\langle x_1, x_2\rangle, \langle y_1, y_2\rangle\rangle \in (S_G^2)^2 \mid \mathbb{IND}\text{-}\mathbb{P}_{\approx}(\{x_1, x_2\}) = \mathbb{IND}\text{-}\mathbb{P}_{\approx}(\{y_1, y_2\})\}$$

Partitioning identity

The members of the partition induced by $\equiv_{\mathbb{IND}\text{-}\mathbb{P}_{\approx}}$ are sets of resource pairs that share the same sharing properties.

Notice that the partitioned pairs contain but are not limited to the identity pairs. Therefore, for sets of pairs closed under simi-discernibility we have the following three possibilities:

1. All pairs are identity pairs. This characterizes a consistent portion of the identity relation, since no simi-discernible pair is left out of this set.
2. Some pairs are identity pairs. This characterizes a portion of the identity relation which is not applied consistently with respect to the simi-discernibility relation.
3. No pairs are identity pairs. This characterizes a portion of the collection of pairs that is consistently kept out of the identity relation.

Each member of the simi-discernibility partition that is not of the third kind, i.e. every set of pairs that contains some identity pair, can be though of as an identity subrelation. The simi-discernibility partition also partitions the identity relation into *identity subrelations*. Each identity subrelation can be described in terms of its discernibility predicates, i.e. in meaningful terms drawn from the domain vocabulary.

Quality & Approximation

Not all identity subrelations have the same quality. Indeed, when we look at the subdivision into three 'categories' above, we are able to distinguish between a lower approximation of identity, as the union of subrelations from the first category (definition 4), and a higher approximation of identity, as the union of subrelations from both the first and the second category (definition 5).

Definition 4 (Lower approximation).

$$x \in \underline{\approx} \iff \{y \mid x \equiv_{\mathbb{IND}\text{-}\mathbb{P}_{\approx}} y\} \subseteq \approx$$

Definition 5 (Higher approximation).

$$x \in \overline{\approx} \iff \{y \mid x \equiv_{\mathbb{IND}\text{-}\mathbb{P}_{\approx}} y\} \cap \approx \neq \emptyset$$

Based on these approximations we can give the rough set representation $\langle \underline{\approx}, \overline{\approx} \rangle$ of identity relation $\approx$. The quality of a rough set representation is given in definition 6. The intuition behind this quality measure is that the crispness of a set should be proportional to the quality of the identity relation on which it is based. Since a consistently applied identity relation has relatively many partition sets that contain either no identity pairs (small value for $\underline{\approx}$) or only identity pairs (large value for $\overline{\approx}$), a more consistent identity relation has a higher accuracy.

Definition 6 (Quality).

$$\alpha(\approx) = \frac{|\underline{\approx}|}{|\overline{\approx}|}$$

4 Conclusion

In this paper we presented a new approach for characterizing, extending, retracting, and assessing identity relations. Our approach does this in purely qualitative terms, using schema semantics.

In section 1 we enumerated three research goals. The first goal is met, since an indiscernibility partition characterizes identity subrelations based on the predicates P (closed under identity) for which the pairs in that sets are indiscernible. In this way we can distinguish between different types of identity by treating P as a description of a (sub)set of identity pairs. We suggest that the meaning of an identity relation and its subrelations is partially defined in its use, i.e., in the indiscernibility criteria it embodies.

The second goal is met, since the notion of a rough set allows us to distinguish between pairs that must be (lower approximation) and those that may be (higher approximation) in the identity relation. If we want to add/remove pairs of the identity relation, we should not consider pairs of the former but only pairs of the latter kind.

The third goal is met, since the measure for rough set accuracy is based on the discernibility criteria of an identity set. The crispness of the set is proportional to the quality of the identity relation, based on its semantic consistency.

References

Antoniou, G.; Groth, P.; van Harmelen, F.; and Hoekstra, R. 2012. *A Semantic Web Primer (Third Edition)*. The MIT Press.

de Melo, G. 2013. Not quite the same: Identity constraints for the web of linked data. In *Proceedings of the American Association for Artificial Intelligence 2013*.

Ding, L.; Shinavier, J.; Finin, T.; and McGuinness, D. L. 2010a. owl:sameas and linked data: an empirical study.

Ding, L.; Shinavier, J.; Shangguan, Z.; and McGuinness, D. 2010b. SameAs networks and beyond: Analyzing deployment status and implications of `owl:sameAs` in linked data. In Patel-Schneider, P. F.; Pan, Y.; Hitzler, P.; Mika, P.; Zhang, L.; Pan, J.; Horrocks, I.; and Glimm, B., eds., *The Semantic Web – ISWC 2010*, volume 6496 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 145–160.

Halpin, H.; Hayes, P.; McCusker, J.; McGuinness, D.; and Thompson, H. 2010a. When `owl:sameAs` isn't the same: An analysis of identity in linked data. In Patel-Schneider, P. F.; Pan, Y.; Hitzler, P.; Mika, P.; Zhang, L.; Pan, J. Z.; Horrocks, I.; and Glimm, B., eds., *The Semantic Web – ISWC 2010*, volume 6496 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 305–320.

Halpin, H.; Hayes, P.; McCusker, J.; McGuinness, D.; and Thompson, H. S. 2010b. *When `owl:sameAs` Isn't the Same: An Analysis of Identity in Linked Data*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg. 305–320.

McCusker, J., and McGuinness, D. 2010. Towards identity in linked data. *Proceedings of OWL Experiences and Directions Seventh Annual Workshop*.

On the Progression of Knowledge in Multiagent Systems

Vaishak Belle*
Dept. of Computer Science
University of Toronto
Toronto, Ontario M5S 3H5, Canada
vaishak@cs.toronto.edu

Gerhard Lakemeyer†
Dept. of Computer Science
RWTH Aachen University
52056 Aachen, Germany
gerhard@cs.rwth-aachen.de

Abstract

In a seminal paper, Lin and Reiter introduced the progression of basic action theories in the situation calculus. In this paper, we study the progression of knowledge in multiagent settings, where after actions, an agent updates her beliefs but also updates what she believes other agents know given what has occurred. By appealing to the notion of only knowing, we are able to avoid limitations of earlier work on multiagent progression, and obtain a new general account: we show that after an action, knowledge bases are updated in a Lin and Reiter fashion at every nesting of modalities. Consequently, recent results on the first-order definability of progression carry over to a multiagent setting without too much effort.

Introduction

Long-lived agents, such as autonomous robots, have to operate purposefully after thousands of actions. As argued in (Reiter 2001), there is only one computational methodology that addresses this problem at large: the idea of *progressing* the current world state, investigated in a general way for the language of the situation calculus in (Lin and Reiter 1997). Indeed, applying repeated regression over so many actions would simply not be feasible. STRIPS technology is a simple form of progression, Lin and Reiter (LR) observe, but in an open-world setting with non-trivial action types, appealing to second-order logic is necessary (Vassos and Levesque 2008). Nonetheless, LR identified two simple cases where progression is first-order definable, and other companion results have been identified since (Liu and Lakemeyer 2009).

However, LR did not consider knowledge. The progression of epistemic theories is very easily motivated in a world where the agent can both *act* and *sense*, and the agent would deliberate on her actions based on what she knows and does not know. An equally compelling case for the progression of knowledge is in multiagent settings, such as the ones robots would inhabit. Here, it is reasonable that after actions, for computational reasons among others, an agent updates her beliefs but also updates her beliefs about what others know given that these actions have occurred.[1] Imagine, for example, a simple domain, involving agents a and b looking for gold (Figure 1). When a observes b moving to Room 2 and looking through its window, her beliefs would be as follows:

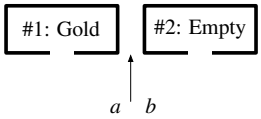


Figure 1: Agents a and b look for a room with gold.

- she still does not know where the gold is;
- a knows that b knows whether Room 2 has the gold, but cannot say what b now knows.

In general, then, progression should say precisely how a's knowledge and a's beliefs about b's knowledge would be updated after any sequence of physical and sensing actions.[2]

The progression of knowledge in the single agent case is considered in (Liu and Wen 2011), but under syntactic restrictions to the initial theory, such as limitations on quantifying-in, and that knowledge does not appear negatively. The progression of knowledge is also considered in (Lakemeyer and Levesque 2009). The results here are general. By appealing to Levesque's (1990) logic of *only knowing*, they obtain a simple specification: given an agent who only knows an action theory, the result of doing an action is that the agent only knows the LR progression of this theory.

The progression of knowledge in the multiagent case is also considered in (Liu and Wen 2011). In addition to the above limitations, formulas such as $\boldsymbol{K}_i p$, where p is objective, can be progressed, but not $\boldsymbol{K}_i \boldsymbol{K}_j \boldsymbol{K}_i p$. That is, among other things, they cannot deal with an agent's beliefs about what others believe about her. In this work, we substantially improve and extend these results. We provide a new general account of the progression of knowledge in multiagent systems that is essentially equivalent to the results in (Lakemeyer and Levesque 2009) when only a single agent is concerned. As a simple consequence of this work, earlier advances on tractable progression apply easily to our results.

*Vaishak Belle thanks the Natural Sciences and Engineering Research Council of Canada for financial support.

†Gerhard Lakemeyer is also affiliated with the Dept. of Computer Science at the University of Toronto.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] We use the terms "knowledge" and "belief" interchangeably, and never require that knowledge needs to be true in the real world.

[2] For simplicity, executed actions are assumed to be publicly observable, even if the information that agents receive may be private and unobservable.

Logical Foundations

The language is a second-order modal dialect with equality, and can be thought of as a situation-suppressed epistemic version of Reiter's (2001) situation calculus, as defined in (Lakemeyer and Levesque 2009). We will not go over the details here, except to note:

- the symbols of the language consist of first-order variables, second-order predicate variables, rigid functions, fluent predicate symbols, as well as these connectives and other symbols: $=, \wedge, \neg, \forall, \Box$, epistemic modalities, parentheses, period, comma;
- the language includes a countably infinite supply of *rigid terms*, which we denote R, that are taken to be isomorphic to the *domain of discourse*;
- *multiagent knowledge* and *only knowing*: the language includes modalities $\boldsymbol{K}_i$ and $\boldsymbol{O}_i$ to reason about what i knows as well as what i only knows;
- *sensing*: every action is assumed to have a binary sensing result. After an action, the agent comes to know that the action was executable (via a special symbol *Poss*), and whether the corresponding sensing outcome holds (via a special symbol SF_i). For simplicity, we keep sensing actions separate from physical ones which only affect the world but do not provide any knowledge to the agent.

A semantics is provided for well-formed formulas in this language by means of *possible worlds*. Formally, let $\mathcal{Z}$ be the set of all finite sequences of elements of R including $\langle\rangle$, the empty sequence. $\mathcal{Z}$ should be understood as the set of all finite sequences of actions. Then a *world* $w \in \mathcal{W}$ is any function from G (the set of ground atoms) and $\mathcal{Z}$ to $\{0, 1\}$.

To interpret epistemic operators, and only knowing in particular, we appeal to the notion of k-structures (Belle and Lakemeyer 2010). For ease of exposition, assume that there are only 2 agents: a (Alice) and b (Bob); a generalization to more agents is straightforward. Then,

Definition 1: A k-structure ($k \geq 1$), say e^k, for an agent is defined inductively as:

- $e^1 \subseteq \mathcal{W} \times \{\{\}\}$,
- $e^k \subseteq \mathcal{W} \times \mathbb{E}^{k-1}$, where $\mathbb{E}^m$ is the set of all m-structures.

Intuitively, a k-structure for a, denoted e^k_a, determines what a believes about the world, but also at each world, what a believes b to know, to depth $k-1$. Such an epistemic state captures the initial beliefs of an agent, and of course, as the agent acts and senses in the world, some possibilities would be discarded. To enable this, we define $w' \simeq^i_z w$ (read: "i considers w' to agree with w on the sensing results throughout action sequence z") inductively by the following:

- $w' \simeq^i_{\langle\rangle} w$ for all w';
- $w' \simeq^i_{z \cdot t} w$ iff $w' \simeq^i_z w$, $w'[Poss(t), z] = 1$ and $w'[SF_i(t), z] = w[SF_i(t), z]$.

Putting all these together, we now turn to the semantic rules for sentences of the logic. These will be defined wrt a world w, an action sequence z, a variable map μ to interpret quantification over second-order variables (see (Lakemeyer and Levesque 2009)) a structure of some depth for a, say e^k_a, and a structure of some depth for b, say e^l_b.[3] The rules for the formulas in the language are given inductively, as usual. The cases for formulas not mentioning epistemic operators is as discussed in (Lakemeyer and Levesque 2009). For epistemic operators, we use the following definition:

Definition 2: Let w be a world, e^k_i a k-structure and z any sequence of actions. Then:

- w_z is a world such that $w_z[p, z'] = w[p, z \cdot z']$ for all ground atoms p and action sequences z';
- $(e^k_i)^w_z$ is defined inductively:
  - $(e^1_i)^w_z = \{(w'_z, \{\}) \mid (w', \{\}) \in e^1_i \text{ and } w' \simeq^i_z w\}$;
  - $(e^k_i)^w_z = \left\{(w'_z, (e^{k-1}_j)^{w'}_z) \mid (w', e^{k-1}_j) \in e^k_i \text{ and } w' \simeq^i_z w\right\}$.

Here, w_z is exactly like w after z. Thus, it is intuitive to think of w_z as the *progression* of w wrt z, and analogously, $(e^k_i)^w_z$ is the progression of e^k_i wrt z given the (real) world w. Indeed, when we progress the structures in e^k_i we are insisting that only those worlds which are compatible with w are considered.[4] Then, the rules for epistemic operators are as follows:

- $e^k_a, e^l_b, w, z, \mu \models \boldsymbol{K}_a\alpha$ iff for all w', for all e^{k-1} for b, if $(w', e^{k-1}_b) \in (e^k_a)^w_z$ then $(e^k_a)^w_z, e^{k-1}_b, w', \langle\rangle, \mu \models \alpha$;
- $e^k_a, e^l_b, w, z, \mu \models \boldsymbol{O}_a\alpha$ iff for all w', for all e^{k-1} for b, $(w', e^{k-1}_b) \in (e^k_a)^w_z$ iff $(e^k_a)^w_z, e^{k-1}_b, w', \langle\rangle, \mu \models \alpha$.

The semantics for $\boldsymbol{K}_b$ and $\boldsymbol{O}_b$ is defined analogously.

Roughly speaking, then, i is said to know α after z at w iff all the structures in i's progressed epistemic state wrt (w, z) are those where α holds initially. Analogously, α is said to be only known after z at w if structures where α holds are precisely the ones in the progressed epistemic state.[5]

We say a sentence α is *satisfiable* if there is a model of "appropriate" depth where α is true (Belle and Lakemeyer 2010). We say α is *valid*, written $\models \alpha$, with the understanding that α is true wrt all models of the appropriate depth.[6]

Finally, the usual properties on introspection can be shown to hold for all action sequences. That is, both $\Box(\boldsymbol{K}_i\alpha \supset \boldsymbol{K}_i\boldsymbol{K}_i\alpha)$ and $\Box(\neg\boldsymbol{K}_i\alpha \supset \boldsymbol{K}_i\neg\boldsymbol{K}_i\alpha)$ are valid. Of course, from only knowing (Levesque 1990), we also get knowledge, that is, $\Box(\boldsymbol{O}_i\alpha \supset \boldsymbol{K}_i\alpha)$ is valid.

[3] The idea is that only formulas with some number of epistemic operators are interpreted wrt structures of a corresponding depth; see (Belle and Lakemeyer 2010) on how to make this precise.

[4] We remark that when constructing $(e^k_i)^w_z$, we use worlds w' that are compatible with w wrt i's sensing results, but then we progress the structures e^{k-1}_j that i considers possible for j at w' as if w' is the real world to obtain $(e^{k-1}_j)^{w'}_z$. The reason is that i simply does not know which world is the real one, and so at each of her possible worlds w', she would "progress" her version of j's epistemic state using w' instead of w.

[5] The difference between the rules for $\boldsymbol{K}_i\alpha$ and $\boldsymbol{O}_i\alpha$ is the "if" in the second line of the former *vs.* "iff" in the latter.

[6] In particular, it is possible to show that if a sentence is satisfiable wrt every model of the appropriate depth, it is also satisfiable wrt every model of higher depths (Belle and Lakemeyer 2010).

Basic Action Theories

We now consider the equivalent of Reiter's *action theories*, the components of which we illustrate using an example:[7]

Example 3: We develop a simple basic action theory for the gold domain in Figure 1, where the gold is to be found in either $r1$ (Room 1) or $r2$ (Room 2). We imagine a single physical action $mv(x, y)$ which gets the agent x to Room y, and a single sensing action $see(x, y)$ which allows the agent x to see through Room y's window and the action reports whether the gold is present. We also use a fluent $Gold(x)$ to say that Room x has the gold, and a fluent $At(x, y)$ to indicate that the agent x is at Room y. More precisely, the basic action theory we are imagining has the following components:

1. $\Box Poss(o) \equiv \exists x, y.\ o = see(x, y) \wedge At(x, y) \ \vee$
   $o = mv(x, y) \wedge true.$
2. $\Box[o]Gold(x) \equiv Gold(x).$
3. $\Box[o]At(x, y) \equiv o = mv(x, y) \ \vee$
   $At(x, y) \wedge \neg\exists u(o = mv(x, u)).$
4. $\Box SF_a(o) \equiv \exists x, y.\ o = see(x, y) \wedge x = a \wedge Gold(y) \ \vee$
   $o = see(x, y) \wedge x \neq a \wedge true.$
5. $\Box SF_b(o) \equiv \exists x, y.\ o = see(x, y) \wedge x = b \wedge Gold(y) \ \vee$
   $o = see(x, y) \wedge x \neq b \wedge true.$

Item 1 is a precondition axiom, items 2 and 3 are successor state axioms, and the remaining are sensing axioms. These latter axioms say this: if x is at y then its sensor would return *true* if y has the gold, and false otherwise. We assume that actions are always public in this work, and so we also need to describe what i learns when he senses j looking through the window. It is this definition that eventually leads to information asymmetry; here, we want that i does not obtain new knowledge on j's sensing. For example, SF_a will simply be *true* when $see(b, r1)$ is performed.

Finally, we want to also say that the gold is in Room 1, and so for an initial theory, we will have $\mathrm{KB}_0 = Gold(r1)$. Moreover, to say that the agents do not know where the gold is, let KB denote the conjunction of: $Gold(r1) \vee Gold(r2)$, $Gold(r1) \equiv \neg Gold(r2)$, and $\forall x, y \neg At(x, y)$. Then, lumping items 1-5 from the example as $\Box\beta$, the following sentence:

$$\mathrm{KB}_0 \wedge \boldsymbol{O}_a(\mathrm{KB} \wedge \Box\beta \wedge \boldsymbol{O}_b(\mathrm{KB} \wedge \Box\beta)) \qquad (1)$$

is a domain where Room 1 has the gold, a does not know where the gold is, and believes the same of b. In contrast, $(1) \wedge \boldsymbol{O}_b(\mathrm{KB}_0 \wedge \Box\beta)$ is a domain where, in fact, b does know where the gold is located and so, a has false beliefs about b.

Let us conclude the section with some entailments which will guide us when accounting for progression.

Proposition 4: *The following are entailed by* (1)*:*

1. $[mv(b, r2)][see(b, r2)]\neg\exists x \boldsymbol{K}_a(Gold(x))$;
2. $[mv(b, r2)][see(b, r2)]\boldsymbol{K}_a\exists x \boldsymbol{K}_b(Gold(x))$;
3. $[mv(b, r2)][see(b, r2)][mv(a, r1)][see(a, r1)]$
   $(\boldsymbol{K}_a Gold(r1) \wedge \boldsymbol{K}_a\boldsymbol{K}_b Gold(r1))$.

[7] See (Lakemeyer and Levesque 2009) for the general definition. Most significantly, since situation terms do not appear in the language, Reiter's (2001) second-order induction axiom, among other things, will not be needed, but the remaining parts of an action theory have an equivalent formulation.

Progression

In this section, we turn to the main results of this paper. The question we would like to ask is this: suppose agents are given basic action theories as their initial knowledge bases, how are we to characterize the knowledge bases after an action is performed? Only knowing will give us the answer.

What we desire for a progression theorem are transformations obeying the following properties:

1. a general characterization of the new knowledge bases regardless of the syntactic restrictions to action types;
2. progression must be allowed to iterate.

We present results supporting this desiderata for a fragment of the language referred to as *progressable sentences*:

Definition 5: A Ocnf formula is any conjunction of formulas of the form $\phi \vee \boldsymbol{O}_i\psi$, where ψ is any formula and ϕ is any fluent formula.

Definition 6: We define *progressable sentences* of depth $k \geq 1$ inductively as follows:

- *i-progressable* and *j-progressable* sentences of depth 1 are basic action theories;
- a *i-progressable* sentence of depth $k > 1$ is the conjunction of any basic action theory Σ and a Ocnf formula $\bigwedge(\phi \vee \boldsymbol{O}_j\psi)$, where each ψ is a j-progressable sentence of depth $< k$;
- a *j-progressable* sentence of depth $k > 1$ is the conjunction of any basic action theory Σ and a Ocnf formula $\bigwedge(\phi \vee \boldsymbol{O}_i\psi)$, where each ψ is a i-progressable sentence of depth $< k$.

If α is a i-progressable sentence (of any depth) then $\boldsymbol{O}_i\alpha$ is referred to a *progressable sentence.* If α is a j-progressable sentence (of any depth) then $\boldsymbol{O}_j\alpha$ is referred to as a *progressable sentence.*

Roughly, progressable sentences are those where i's beliefs about the world are precisely determined by an action theory, and i has beliefs about possible knowledge bases that j only knows.[8] Clearly, (1) would be classified as progressable, as would a sentence of the form $\boldsymbol{O}_a(\phi \wedge \Box\beta \wedge (\varphi \supset \boldsymbol{O}_b\psi \wedge \varphi' \supset \boldsymbol{O}_b\psi'))$ where, by requirement, ψ and ψ' are b-progressable, and thus, perhaps action theories themselves.

Progression for Physical Actions In the following we assume π to refer to the RHS of the precondition axioms, and γ_F to refer to the RHS of successor state axioms. Also, let $\alpha^{\vec{F}}_{\vec{P}}$ denote the formula α with every occurrence of fluent predicate symbols F_i replaced by second-order variables P_i of the same arity. The first main result is a characterization, in general terms, of what is known after a physical action.[9]

[8] Note that, then, the single agent case is essentially identical to (Lakemeyer and Levesque 2009).

[9] To avoid notational clutter, we make two simplifying assumptions: (a) we state theorems for progressable sentences of depth 2, and (b) we assume that the dynamic components $\Box\beta$ are the same at all depths. Generalizing these is straightforward but tedious.

Theorem 7: *Suppose* $\Upsilon = \boldsymbol{O}_i(\phi \wedge \Box\beta \wedge \bigwedge(\varphi_u \vee \boldsymbol{O}_j(\psi_u \wedge \Box\beta)))$*, where* $u \in \{1, \ldots, n\}$*, is a progressable theory, where* ϕ, φ_u *and* ψ_u *are fluent sentences and* $\Box\beta$ *are the dynamic components. Suppose* t *is a physical action. Then:*

$$\models \Upsilon \supset [t]\boldsymbol{O}_i(\mathcal{P}_\phi \wedge \Box\beta \wedge \bigwedge(\varphi_u \vee \boldsymbol{O}_j(\mathcal{P}_{\psi_u} \wedge \Box\beta)))$$

where for a fluent formula α*:*

$$\mathcal{P}_\alpha = \exists \vec{P}.\ [(\alpha \wedge \pi_t^o)^{\vec{F}}_{\vec{P}} \wedge \bigwedge \forall \vec{x}.\ F(\vec{x}) \equiv \gamma_{F_t^o}^{\vec{F}}].$$

This theorem says that, at all depths, if all that i knows is a basic action theory, then after an action, he knows another basic action theory where the knowledge base is the *progression* of the previous initial knowledge base. The resulting theory is still *progressable*, and so progression can iterate.

On the correctness of belief expansion, we have:

Theorem 8: *Suppose* Υ *is as above. Then*

$$\Upsilon \models [t]\boldsymbol{K}_i\alpha \ \ \textit{iff}\ \ \Upsilon' \models \boldsymbol{K}_i\alpha$$

where Υ' *is the new progressed theory from Theorem 7.*

This result says that whatever is believed by the progressed theory is equivalently believed after t from the initial knowledge base. That is, the two theories agree on the future.

Example 9: Consider the effect of $mv(b, r2)$ on the progressable theory (1). From Theorem 7, we have:

$$\models (1) \supset [mv(b, r2)]\boldsymbol{O}_a(\mathcal{P}_{\mathrm{KB}} \wedge \Box\beta \wedge \boldsymbol{O}_b(\mathcal{P}_{\mathrm{KB}} \wedge \Box\beta))$$

where $\mathcal{P}_{\mathrm{KB}}$ can be shown to be equivalent to $\{Gold(r1) \vee Gold(r2), Gold(r1) \equiv \neg Gold(r2), At(b, r2), \forall x, y.\ x \neq b \supset \neg At(x, y)\}$. Then, readers may note that the following sentences are entailed by $\boldsymbol{O}_a(\mathcal{P}_{\mathrm{KB}} \wedge \Box\beta \wedge \boldsymbol{O}_b(\mathcal{P}_{\mathrm{KB}} \wedge \Box\beta))$: $\boldsymbol{K}_a At(b, r2)$, $\boldsymbol{K}_a\boldsymbol{K}_b At(b, r2)$ and $\boldsymbol{K}_a\boldsymbol{K}_b\neg At(a, r2)$.

Finally, we remark that $\mathcal{P}_\phi$ for any initial theory ϕ is essentially the second-order account of progression given by LR; see (Lakemeyer and Levesque 2009) for discussions.

From this general definition, standard progression results on first-order definability (Liu and Lakemeyer 2009) apply in a straightforward manner. For example, we have:

Theorem 10: *Suppose* Υ *is as above, where* ϕ *and* ψ_u *are first-order formulas. If* Υ *is local-effect, then* $\mathcal{P}_\phi$ *and* $\mathcal{P}_{\psi_u}$ *from Theorem 7 are definable as first-order formulas.*

See (Liu and Lakemeyer 2009) for a definition of local-effect action theories; analogously, first-order progression for other classes of actions carry over without too much effort. See a long version of this paper for details.

Progression for Sensing Actions Letting θ_i refer to the RHS of SF_i in a basic action theory, we have the second main result of this paper:

Theorem 11: *Suppose* $\Upsilon = \boldsymbol{O}_i(\phi \wedge \Box\beta \wedge \bigwedge(\varphi_u \vee \boldsymbol{O}_j(\psi_u \wedge \Box\beta)))$*, where* $u \in \{1, \ldots, n\}$*, is a progressable theory as before. Suppose* t *is a sensing action. Then,*

$$\begin{aligned}\models \Upsilon \supset\ & (SF_i(t) \supset [t]\boldsymbol{O}_i(\phi \wedge \pi_t^o \wedge \theta_{i_t}^o \wedge \Box\beta \wedge \bigwedge(\varphi_u \vee \Psi_u))) \wedge \\ & (\neg SF_i(t) \supset [t]\boldsymbol{O}_i(\phi \wedge \pi_t^o \wedge \neg\theta_{i_t}^o \wedge \Box\beta \wedge \bigwedge(\varphi_u \vee \Psi_u)))\end{aligned}$$

where Ψ_u *is the following sentence:*

$$\begin{aligned}&\theta_{j_t}^o \supset \boldsymbol{O}_j(\psi_u \wedge \pi_t^o \wedge \theta_{j_t}^o \wedge \Box\beta) \wedge \\ &\neg\theta_{j_t}^o \supset \boldsymbol{O}_j(\psi_u \wedge \pi_t^o \wedge \neg\theta_{j_t}^o \wedge \Box\beta).\end{aligned}$$

This theorem says that after a sensing action, what i learns for herself is contingent on the outcome of $SF_i(t)$. But as far as i's beliefs about j goes, she is uncertain about the outcome of sensing for j, and so her beliefs about j would differ at one place: $\theta_{j_t}^o$ *vs.* $\neg\theta_{j_t}^o$. Let us illustrate this using an example:

Example 12: Consider the progression of the theory from Example 9 wrt the sensing action $see(b, r2)$, denoted t. Let ζ denote $\mathcal{P}_{\mathrm{KB}}$ from that example, and so for an initial theory, we have $\Upsilon = Gold(r1) \wedge \boldsymbol{O}_a(\zeta \wedge \Box\beta \wedge \boldsymbol{O}_b(\zeta \wedge \Box\beta))$. After t, from Theorem 11 we have $\Upsilon \models SF_a(t) \supset [t]\boldsymbol{O}_a(\zeta^* \wedge \Box\beta\ \wedge \theta_{b_t}^o \supset \boldsymbol{O}_b(\zeta' \wedge \Box\beta)\ \wedge \neg\theta_{b_t}^o \supset \boldsymbol{O}_b(\zeta'' \wedge \Box\beta))$ because $SF_a(t)$ is *true*, and where $\zeta^* = \zeta \wedge \pi_t^o \wedge \theta_{a_t}^o$, which is equivalent to ζ; $\zeta' = \zeta \wedge \pi_t^o \wedge \theta_{b_t}^o$, where $\theta_{b_t}^o$ is $\exists x, y.\ (t = see(x, y) \wedge x = b \wedge Gold(y) \vee (t = see(x, y) \wedge x \neq b))$, which is equivalent to $Gold(r2)$; and $\zeta'' = \zeta \wedge \pi_t^o \wedge \neg\theta_{b_t}^o$. Thus, ζ' is equivalent to $\zeta \wedge Gold(r2)$ and ζ'' is equivalent to $\zeta \wedge \neg Gold(r2)$. Thus, a now knows that b knows whether Room 2 has the gold. Formally, the progressed theory can be shown to entail $\boldsymbol{K}_a(\boldsymbol{K}_b Gold(r2) \vee \boldsymbol{K}_b\neg Gold(r2))$, $\boldsymbol{K}_a\exists x\boldsymbol{K}_b Gold(x)$, and $\neg\exists x\boldsymbol{K}_a\boldsymbol{K}_b Gold(x)$.

Example 13: As a final example on progression, consider the sentence $\mathrm{KB}_0 \wedge \boldsymbol{O}_a(\mathrm{KB} \wedge \Box\beta \wedge \boldsymbol{O}_b(\mathrm{KB} \wedge \Box\beta \wedge \boldsymbol{O}_a(\mathrm{KB} \wedge \Box\beta)))$. Its progression wrt $mv(b, r2) \cdot see(b, r2)$ simplifies to:

$$\begin{aligned}&\boldsymbol{O}_a(\zeta \wedge \Box\beta \wedge \\ &\quad (Gold(r2) \supset \boldsymbol{O}_b(\zeta \wedge Gold(r2) \wedge \Box\beta \wedge \boldsymbol{O}_a(\zeta \wedge \Box\beta))) \wedge \\ &\quad (\neg Gold(r2) \supset \boldsymbol{O}_b(\zeta \wedge \neg Gold(r2) \wedge \Box\beta \wedge \boldsymbol{O}_a(\zeta \wedge \Box\beta))))\end{aligned}$$

where ζ is from the previous example. The new knowledge base entails $\boldsymbol{K}_a\boldsymbol{K}_b\boldsymbol{K}_a At(b, r2)$ and $\boldsymbol{K}_a\boldsymbol{K}_b\neg\exists x\boldsymbol{K}_a Gold(x)$.

Conclusions

We obtained new *general* results on the progression of knowledge in multiagent systems that substantially extended previous work: we showed that given any action theory, knowledge bases at all depths can be progressed in a LR fashion after actions. For the future, we would like to consider *private* actions, and go beyond Ocnf formulas.

References

Belle, V., and Lakemeyer, G. 2010. Multi-agent only-knowing revisited. In *Proc. KR*, 49–60.

Lakemeyer, G., and Levesque, H. J. 2009. A semantical account of progression in the presence of defaults. In *IJCAI*, 842–847.

Levesque, H. J. 1990. All I know: a study in autoepistemic logic. *Artificial Intelligence* 42(2-3):263–309.

Lin, F., and Reiter, R. 1997. How to progress a database. *Artificial Intelligence* 92(1-2):131–167.

Liu, Y., and Lakemeyer, G. 2009. On first-order definability and computability of progression for local-effect actions and beyond. In *Proc. IJCAI*, 860–866.

Liu, Y., and Wen, X. 2011. On the progression of knowledge in the situation calculus. In *Proc. IJCAI*, 976–982.

Reiter, R. 2001. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT Press.

Vassos, S., and Levesque, H. 2008. On the Progression of Situation Calculus Basic Action Theories: Resolving a 10-year-old Conjecture. In *Proc. AAAI*, 1004–1009.

Interval Methods for Judgment Aggregation in Argumentation

Richard Booth[1], **Edmond Awad**[2] and **Iyad Rahwan**[2,3]
[1]University of Luxembourg, Luxembourg
[2]Masdar Institute of Science & Technology, UAE
[3]University of Edinburgh, UK

Abstract

Given a set of conflicting arguments, there can exist multiple plausible opinions about which arguments should be accepted, rejected, or deemed undecided. Recent work explored some operators for deciding how multiple such judgments should be aggregated. Here, we generalize this line of study by introducing a family of operators called *interval aggregation methods*, which contain existing operators as instances. While these methods fail to output a complete labelling in general, we show that it is possible to *transform* a given aggregation method into one that *does* always yield collectively rational labellings. This employs the *down-admissible* and *up-complete* constructions of Caminada and Pigozzi. For interval methods, collective rationality is attained at the expense of a strong *Independence* postulate, but we show that an interesting weakening of the Independence postulate is retained.

Introduction

A conflicting knowledge base can be viewed abstractly as a set of arguments (defeasible derivations), and a binary relation capturing conflicts among them, forming an *argumentation framework* (AF) (Dung 1995). Given a set of conflicting arguments, there can exist multiple plausible ways to identify (or *label*) which arguments should be accepted, rejected, or deemed undecided (Baroni, Caminada, and Giacomin 2011). The question we explore here is how to aggregate the judgments of multiple agents who have different opinions about how to evaluate a given set of arguments.

This problem of *Judgment Aggregation* (JA) has been explored extensively in classical logic (List and Puppe 2009). But it was only recently that JA has been applied to collective argument evaluation (Caminada and Pigozzi 2011; Rahwan and Tohmé 2010). Early results showed that argument-wise plurality voting cannot guarantee that the outcome of aggregation is always rational (consistent)—thus simple voting violates *Collective Rationality* (Rahwan and Tohmé 2010). On the other hand, the aggregation operators of Caminada and Pigozzi are able to guarantee collective rationality, but do so at the expense of the *Independence* property (Caminada and Pigozzi 2011).

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In the present paper, we embark on a broader study of JA in argumentation. We define a general family of aggregation operators called *interval methods* and show that they contain existing operators as instances. Interval methods always satisfy a strong version of Independence, but will usually fail Collective Rationality. But despite this important barrier, we are able to fully axiomatize interval methods in terms of a set of fundamental postulates. Then, building on Caminada and Pigozzi's *down-admissible* + *up-complete* (DAUC) construction, we present an approach to transform any interval method into one satisfying *Collective Rationality* while preserving a weaker and more reasonable form of independence known as *Directionality*.

Preliminaries

We assume a countably infinite set U of argument names, from which all possible argumentation frameworks are built.

Definition 1 *An* argumentation framework *(AF for short)* $\mathcal{A} = (Args, \rightharpoonup)$ *is a pair consisting of a finite set* $Args \subseteq U$ *of arguments and an* attack *relation* $\rightharpoonup \subseteq Args \times Args$. *Sometimes we use* $Args_{\mathcal{A}}$ *and* $\rightharpoonup_{\mathcal{A}}$ *to denote the arguments and attack relation of a given AF* $\mathcal{A}$.

An AF is evaluated by assigning one of the labels `in`, `out` or `undec` to each argument in $Args_{\mathcal{A}}$, standing for *accepted*, *rejected* and *undecided* respectively (Caminada 2006). We define a unary "negation" operator on the set of labels by setting $\neg\texttt{in} = \texttt{out}$, $\neg\texttt{out} = \texttt{in}$ and $\neg\texttt{undec} = \texttt{undec}$. Given an AF $\mathcal{A}$, an $\mathcal{A}$*-labelling* is a function $L : Args \rightarrow \{\texttt{in}, \texttt{out}, \texttt{undec}\}$. For each $\texttt{x} \in \{\texttt{in}, \texttt{out}, \texttt{undec}\}$ we denote by $L^{-1}(\texttt{x})$ the inverse image of x under L, and given $A \subseteq Args_{\mathcal{A}}$ we denote by $L[A]$ the restriction of L to A.

A *rational* evaluation should somehow respect the attack relation, as captured by the notion of *complete* labelling.

Definition 2 *Let* $\mathcal{A} = (Args, \rightharpoonup)$ *be an AF and* L *be an* $\mathcal{A}$*-labelling.* L *is a* complete $\mathcal{A}$*-labelling iff, for all* $a \in Args$*:*

- *If* $L(a) = \texttt{in}$ *then* $L(b) = \texttt{out}$ *for all* $b \in Args$ *s.t.* $b \rightharpoonup a$.
- *If* $L(a) = \texttt{out}$ *then* $L(b) = \texttt{in}$ *for some* $b \in Args$ *s.t.* $b \rightharpoonup a$.
- *If* $L(a) = \texttt{undec}$ *then* $L(b) \neq \texttt{in}$ *for all* $b \in Args$ *s.t.* $b \rightharpoonup a$ *and* $L(c) = \texttt{undec}$ *for some* $c \in Args$ *s.t.* $c \rightharpoonup a$.

An admissible $\mathcal{A}$*-labelling is one that satisfies the first two conditions above. We denote the set of complete* $\mathcal{A}$*-labellings by* $Comp(\mathcal{A})$.

We assume a set of agents $Ag = \{1, \ldots, n\}$ (with $n \geqslant 2$) is fixed. An $\mathcal{A}$-*profile* is a sequence $\mathbf{L} = (L_1, \ldots, L_n)$ assigning a **complete** $\mathcal{A}$-labelling to each $i \in Ag$. Given $A \subseteq Args_{\mathcal{A}}$ we denote by $\mathbf{L}[A]$ the profile $(L_1[A], \ldots, L_n[A])$ (writing $\mathbf{L}[a]$ rather than $\mathbf{L}[\{a\}]$ for the singleton case). For each label $\mathtt{x} \in \{\mathtt{in}, \mathtt{out}, \mathtt{undec}\}$ and $a \in Args_{\mathcal{A}}$ we denote the set of agents who *voted* for label $\mathtt{x}$ for a by $V^{\mathbf{L}}_{a:\mathtt{x}}$, i.e., $V^{\mathbf{L}}_{a:\mathtt{x}} = \{i \in Ag \mid L_i(a) = \mathtt{x}\}$.

The central concept of this paper is the following.

Definition 3 *An* aggregation method *is a function F that assigns to every AF $\mathcal{A}$ and $\mathcal{A}$-profile $\mathbf{L}$ an $\mathcal{A}$-labelling $F_{\mathcal{A}}(\mathbf{L})$.*

Postulates for aggregation methods

We start with some postulates for a good aggregation method. Some are inspired by postulates in (Caminada and Pigozzi 2011; Rahwan and Tohmé 2010), which in turn were inspired by those familiar from the JA literature. We modify them to account for allowing the AF to vary. Note that free occurrences of $\mathcal{A}$ and $\mathbf{L}$ within the postulates are implicitly universally quantified. Ideally, of course, we would like the output too to be complete.

Collective Rationality $F_{\mathcal{A}}(\mathbf{L}) \in Comp(\mathcal{A})$.

Full *Collective Rationality* will turn out to be beyond the reach of the the simplest aggregation methods. However, a very weak version turns out to be relatively easy to satisfy. We call $\mathcal{A}$ a *2-loop AF* if it consists only of two arguments that mutually attack each other, i.e., $Args_{\mathcal{A}} = \{a, b\}$ and $\rightharpoonup_{\mathcal{A}} = \{(a, b), (b, a)\}$ for some distinct $a, b \in U$.

Minimal Collective Rationality For any 2-loop AF $\mathcal{A}$ we have $F_{\mathcal{A}}(\mathbf{L}) \in Comp(\mathcal{A})$.

Given an $\mathcal{A}$-profile $\mathbf{L} = (L_1, \ldots, L_n)$, we say $\mathbf{L}'$ is a *permutation* of $\mathbf{L}$ if $\mathbf{L}' = (L_{\sigma(1)}, \ldots, L_{\sigma(n)})$ for some permutation σ on Ag.

Anonymity If $\mathbf{L}'$ is a permutation of $\mathbf{L}$ then $F_{\mathcal{A}}(\mathbf{L}) = F_{\mathcal{A}}(\mathbf{L}')$.

Unanimity If there is some $\mathcal{A}$-labelling L such that $L_i = L$ for all $i \in Ag$ then $F_{\mathcal{A}}(\mathbf{L}) = L$.

The idea behind the next postulate is that AFs that are *isomorphic* should be treated the same when aggregating. Given $\mathcal{A}_1 = (Args_1, \rightharpoonup_1)$ and $\mathcal{A}_2 = (Args_2, \rightharpoonup_2)$, an isomorphism from $\mathcal{A}_1$ to $\mathcal{A}_2$ is a bijection $g : Args_1 \to Args_2$ such that, for all $a, b \in Args_1$ we have $a \rightharpoonup_1 b$ iff $g(a) \rightharpoonup_2 g(b)$. Such a g extends to a mapping between the $\mathcal{A}_1$-labellings and the $\mathcal{A}_2$-labellings. For any $\mathcal{A}_1$-labelling L we define the $\mathcal{A}_2$-labelling $g(L)$ by setting, for all $a \in \mathcal{A}_2$, $[g(L)](a) = L(g^{-1}(a))$. The function g further extends naturally to a mapping between $\mathcal{A}_1$-profiles and $\mathcal{A}_2$-profiles by setting, for any $\mathcal{A}_1$-profile $\mathbf{L} = (L_1, \ldots, L_n)$, $g(\mathbf{L}) = (g(L_1), \ldots, g(L_n))$.

Isomorphism Suppose $\mathcal{A}_1$ and $\mathcal{A}_2$ are connected by isomorphism g. Then, for any $\mathcal{A}_1$-profile $\mathbf{L}$ we have $g(F_{\mathcal{A}_1}(\mathbf{L})) = F_{\mathcal{A}_2}(g(\mathbf{L}))$.

A standard idea in aggregation is that the group evaluation concerning some item should depend only on the individuals' evaluations over that item and no others. Given we allow the AF to vary, we strengthen this property somewhat.

AF-Independence If $\mathbf{L}_1$ and $\mathbf{L}_2$ are profiles over $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively and $a \in Args_{\mathcal{A}_1} \cap Args_{\mathcal{A}_2}$ then $\mathbf{L}_1[a] = \mathbf{L}_2[a]$ implies $[F_{\mathcal{A}_1}(\mathbf{L}_1)](a) = [F_{\mathcal{A}_2}(\mathbf{L}_2)](a)$.

This postulate implies the more commonly used version of Independence (just put $\mathcal{A}_1 = \mathcal{A}_2$). It roughly says that the collective label of a depends only on $\mathbf{L}[a]$ *no matter what other arguments might be present or absent in* $\mathcal{A}$.

Our first *monotonicity* postulate, in/out-*Monotonicity*, says that if some agents change their individual labels of some arguments in profile $\mathbf{L}$ so that they agree with the collective labelling $F_{\mathcal{A}}(\mathbf{L})$, assuming those collective labels are in $\{\mathtt{in}, \mathtt{out}\}$, then the collective labelling does not change.

in/out-Monotonicity Let $\mathbf{L}, \mathbf{L}'$ be $\mathcal{A}$-profiles such that for all $a \in Args_{\mathcal{A}}$ and all $i \in Ag$, $(L'_i(a) \neq L_i(a)$ implies $L'_i(a) = [F_{\mathcal{A}}(\mathbf{L})](a) \in \{\mathtt{in}, \mathtt{out}\})$. Then $F_{\mathcal{A}}(\mathbf{L}') = F_{\mathcal{A}}(\mathbf{L})$.

The intuition behind *Strong* in/out-*Monotonicity* is that if some agents in $\mathbf{L}$ move their individual labels of some arguments *closer* towards the collective label (and those collective labels belong to $\{\mathtt{in}, \mathtt{out}\}$), then the resulting collective labelling remains unchanged. To formulate it we use the notion of one label being *between* another two labels. Given $\mathtt{x}, \mathtt{y}, \mathtt{z} \in \{\mathtt{in}, \mathtt{out}, \mathtt{undec}\}$ we say that $\mathtt{y}$ is between $\mathtt{x}$ and $\mathtt{z}$ iff either $\mathtt{y} = \mathtt{x}$ or $\mathtt{y} = \mathtt{z}$ or [$\mathtt{y} = \mathtt{undec}$ and $\mathtt{x} \neq \mathtt{z}$].

Strong in/out-Monotonicity Let $\mathbf{L}$, $\mathbf{L}'$ be $\mathcal{A}$-profiles such that for all $a \in Args_{\mathcal{A}}$ such that $[F_{\mathcal{A}}(\mathbf{L})](a) \in \{\mathtt{in}, \mathtt{out}\}$ and all $i \in Ag$, $L'_i(a)$ is between $L_i(a)$ and $[F_{\mathcal{A}}(\mathbf{L})](a)$. Then $F_{\mathcal{A}}(\mathbf{L}') = F_{\mathcal{A}}(\mathbf{L})$.

The next postulate says the collective label on any argument never goes against the individual label of any agent (Caminada and Pigozzi 2011).

Compatibility For all $i \in Ag$ and $a \in Args_{\mathcal{A}}$ we have $[F_{\mathcal{A}}(\mathbf{L})](a) = \neg L_i(a)$ implies $[F_{\mathcal{A}}(\mathbf{L})](a) = \mathtt{undec}$.

Given any n-tuple (l_i) of labels the in/out-*winner in* (l_i) is the label among $\{\mathtt{in}, \mathtt{out}\}$ which appears more frequently in (l_i) (if such a label exists). E.g. the in/out-winner in $(\mathtt{in}, \mathtt{undec}, \mathtt{out}, \mathtt{undec}, \mathtt{in})$ is $\mathtt{in}$. If $\mathtt{x}$ is the in/out-winner in (l_i) then we call $\neg\mathtt{x}$ the in/out-*loser*. A weaker version of *Compatibility* can then be formulated as follows:

in/out-Plurality If $\mathtt{x}$ is the in/out-loser in $(L_i(a))_{i \in Ag}$ then $[F_{\mathcal{A}}(\mathbf{L})](a) \neq \mathtt{x}$

Proposition 1 *Let F be an aggregation method satisfying* Compatibility. *Then*
(i). *F satisfies* in/out-Plurality.
(ii). *If F satisfies* in/out-Monotonicity *then it satisfies* Strong in/out-Monotonicity.

Interval aggregation methods

Now we describe the family of *interval aggregation methods*, which will include a number of interesting special cases (and which are closely-related to the *quota rules* considered in JA by (Dietrich and List 2007)). Formally, let Int_n be the set of *intervals* of non-zero length in $\{0, 1, \ldots, n\}$ (recall n is the number of agents), i.e., $Int_n = \{(k, l) \mid k <$

$l,\ k,l \in \{0,1,\ldots,n\}\}$. Let $Y \subseteq Int_n$ be some subset of distinguished intervals in Int_n. Then we define aggregation method F^Y by setting, for each $\mathcal{A}$, $\mathcal{A}$-labelling profile $\mathbf{L}$ and $a \in Args_{\mathcal{A}}$:

$$[F^Y_{\mathcal{A}}(\mathbf{L})](a) = \begin{cases} \texttt{x} & \text{if } \texttt{x} \in \{\texttt{in}, \texttt{out}\} \text{ and} \\ & \quad (|V^{\mathbf{L}}_{a:\neg\texttt{x}}|, |V^{\mathbf{L}}_{a:\texttt{x}}|) \in Y \\ \texttt{undec} & \text{otherwise} \end{cases}$$

Definition 4 *An* interval aggregation method *is an aggregation method F such that $F = F^Y$ for some $Y \subseteq Int_n$ satisfying* **(I1)***: $(0,n) \in Y$.*

By making different choices of Y we find some special instances of interval methods.

Argument-wise plurality: Take the collective label of a to be the label among $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$ that gets the most votes. If there is a tie then take undec. This corresponds to $Y_{\mathrm{AWP}} = \{(k,l) \in Int_n \mid n-(k+l) < l\}$. We use F^{AWP} to denote $F^{Y_{\mathrm{AWP}}}$.

Majority: Take the collective label of a to be x if more than half of the agents voted for it, otherwise take undec. $Y_{\mathrm{Maj}} = \{(k,l) \in Int_n \mid l > n/2\}$. We use F^{Maj} to denote $F^{Y_{\mathrm{Maj}}}$.

Sceptical initial: (Caminada and Pigozzi 2011) Take the in/out winner if it is the unanimous choice among the agents, otherwise undec. $Y_{\mathrm{Scept}} = \{(0,n)\}$. We use F^{Scept} to denote $F^{Y_{\mathrm{Scept}}}$.

Credulous initial: (Caminada and Pigozzi 2011) Take the in/out-winner x whenever no agent voted for $\neg$x, otherwise undec. $Y_{\mathrm{Cred}} = \{(0,l) \in Int_n \mid l \geqslant 1\}$. We use F^{Cred} to denote $F^{Y_{\mathrm{Cred}}}$.

in/out-*winner:* Take the in/out-winner whenever it exists. $Y_{\mathrm{iow}} = Int_n$. We use F^{iow} to denote $F^{Y_{\mathrm{iow}}}$.

We obtain the following axiomatic characterisation.

Theorem 1 *Let F be an aggregation method. Then F is an interval aggregation method iff it satisfies:* Minimal Collective Rationality*,* Anonymity*,* Unanimity*,* Isomorphism*,* AF-Independence *and* in/out-Plurality.

Thus we see that most of the postulates from the previous section are sound for the interval methods. The postulates missing from Thm. 1 are the two *Monotonicity* postulates, *Compatibility* and, most significantly, *Collective Rationality*. None of these will hold in general for interval methods, at least not without placing some extra restrictions on Y beyond only **(I1)**. Looking first at in/out-*Monotonicity* we can say the following:

Proposition 2 (i). *There exists an interval method that does not satisfy* in/out-Monotonicity.
(ii). *F^{AWP}, F^{Maj}, F^{Scept}, F^{Cred} and F^{iow} all satisfy* in/out-Monotonicity.

We obtain *Strong* in/out-*Monotonicity* for an interval method F^Y if we assume Y satisfies an extra condition saying that Y is closed under *widening* intervals:

(I2) If $(k,l) \in Y$ and $s \leqslant k,\ l \leqslant t$ then $(s,t) \in Y$.

Proposition 3 *Let F^Y be an interval method. Then F^Y satisfies* Strong in/out-Monotonicity *iff Y satisfies* **(I2)**.

Definition 5 *If $Y \subseteq Int_n$ satisfies both* **(I1)** *and* **(I2)** *then we say Y is* widening*. A* widening interval method *is an aggregation method F such that $F = F^Y$ for some widening Y.*

Putting Thm. 1 and Prop. 3 together we can see that the class of widening interval methods is characterised by the six postulates of Thm. 1 plus *Strong* in/out-*Monotonicity*.

It can be checked that each of our previous examples of interval methods, apart from Y_{AWP}, are widening and so yield interval methods that satisfy *Strong* in/out-*Monotonicity*. However if we want *Compatibility* to hold then we need to place a further restriction on Y:

(I3) If $(k,l) \in Y$ then $k = 0$.

Proposition 4 *Let F^Y be an interval method. Then F^Y satisfies* Compatibility *iff Y satisfies* **(I3)**.

Clearly, among our examples, Y_{Scept} and Y_{Cred} are the only Y that satisfy **(I3)**, which means that F^{Scept} and F^{Cred} are the only interval methods among our examples that satisfy *Compatibility*. Looking more generally, combining the previous proposition with Thm. 1 and Prop. 3 (and recalling the facts about *Compatibility* in Prop. 1) gives us the following result.

Theorem 2 *Let F be an aggregation method. Then the following are equivalent:*
(i). *$F = F^Y$ for some Y of the form $\{(0,t) \mid t \geqslant l\}$ for some $1 \leqslant l \leqslant n$.*
(ii). *F satisfies* Minimal Collective Rationality*,* Anonymity*,* Unanimity*,* Isomorphism*,* AF-Independence*,* Compatibility *and* in/out-Monotonicity

Regarding *Collective Rationality*, we know already from (Caminada and Pigozzi 2011; Rahwan and Tohmé 2010) that our examples of interval methods above fail to satisfy it. Is there any requirement we can place on Y to ensure it? Unfortunately the answer is no, as the following impossibility result (whose proof has a flavour of similar impossibility results commonly seen in JA, e.g., Thm. 1 of (List and Pettit 2002)) shows.

Theorem 3 *There is no aggregation method (for any $n > 1$) satisfying all of* Isomorphism*,* Anonymity*,* Unanimity*,* AF-Independence *and* Collective Rationality.

Thus, given the basic requirements *Isomorphism*, *Anonymity* and *Unanimity*, there is no hope to obtain *both* of *AF-Independence* and *Collective Rationality*. We now look at relaxing *AF-Independence*.

Weakening AF-Independence

One might argue that *AF-Independence* cannot be expected to hold when part of the input to the aggregation explicitly contains information (in the form of the attack relation $\rightharpoonup_{\mathcal{A}}$) regarding dependencies between arguments. Instead we might expect the following weaker version, inspired by a similar postulate for argumentation semantics from (Baroni and Giacomin 2007). The idea is that if we have a set of arguments in $\mathcal{A}$ that is *unattacked* then we can aggregate just that part without looking at the arguments outside the

set. Note $\mathcal{A} \subseteq_{\mathrm{f}} \mathcal{A}'$ indicates that $Args_{\mathcal{A}} \subseteq Args_{\mathcal{A}'}$ and $\rightharpoonup_{\mathcal{A}} = \rightharpoonup_{\mathcal{A}'} \cap (Args_{\mathcal{A}} \times Args_{\mathcal{A}})$.

Directionality Suppose $\mathcal{A} \subseteq_{\mathrm{f}} \mathcal{A}'$ and suppose $Args_{\mathcal{A}}$ is unattacked in $\mathcal{A}'$. Then for any $\mathcal{A}'$-profile $\mathbf{L}$ and $a \in Args_{\mathcal{A}}$ we have $[F_{\mathcal{A}'}(\mathbf{L})](a) = [F_{\mathcal{A}}(\mathbf{L}[Args_{\mathcal{A}}])](a)$.

Proposition 5 *Every aggregation method F that satisfies* AF-Independence *also satisfies* Directionality.

Can we construct an aggregation method that satisfies *Directionality*, *Collective Rationality* and some other desirable postulates? We show the answer is yes, using the *down-admissible* and *up-complete* constructions of (Caminada and Pigozzi 2011). We begin with the down-admissible construction, which uses the definition of the 'committedness' relation $\sqsubseteq$ according to which $L_1 \sqsubseteq L_2$ iff both $L_1^{-1}(\mathtt{in}) \subseteq L_2^{-1}(\mathtt{in})$ and $L_1^{-1}(\mathtt{out}) \subseteq L_2^{-1}(\mathtt{out})$.

Definition 6 ((Caminada and Pigozzi 2011)) *Given an $\mathcal{A}$-labelling L, the* down-admissible *labelling of L, denoted by $\downarrow L$, is the (unique) greatest element (under $\sqsubseteq$) of the set of all admissible $\mathcal{A}$-labellings M such that $M \sqsubseteq L$.*

As described in (Caminada and Pigozzi 2011), it can be arrived at by iteratively relabelling every argument that is illegally `in` or illegally `out` with `undec` until no illegal `in` or `out` labels remain. The result is a labelling that is admissible, but not necessarily complete. To ensure a complete labelling we need to additionally apply the up-complete operator.

Definition 7 ((Caminada and Pigozzi 2011)) *Given an admissible $\mathcal{A}$-labelling L, the* up-complete *labelling of L, denoted by $\uparrow L$, is the (unique) smallest element (under $\sqsubseteq$) of the set of all complete $\mathcal{A}$-labellings M such that $L \sqsubseteq M$.*

To obtain $\uparrow L$ we iteratively change every illegally `undec` argument to `in` or `out` as appropriate (Caminada and Pigozzi 2011). We denote by $\Updownarrow L$ the composite operation of performing the down-admissible followed by the up-complete procedures on an $\mathcal{A}$-labelling L.

Definition 8 *Given any aggregation method F, the* DAUC version of F *is the aggregation method $\widehat{F}$ defined by setting, for any AF $\mathcal{A}$ and $\mathcal{A}$-labelling profile $\mathbf{L}$, $\widehat{F}_{\mathcal{A}}(\mathbf{L}) = \Updownarrow(F_{\mathcal{A}}(\mathbf{L}))$.*

For the special cases of interval methods F^{Scept} and F^{Cred} this procedure was studied in detail in (Caminada and Pigozzi 2011). Their DAUC versions were called the *sceptical* and *super-credulous* aggregation methods respectively there. We lose *AF-Independence* as expected. But we can show that some postulates satisfied by the initial method F can be *inherited* by $\widehat{F}$.

Proposition 6 *Let F be any aggregation method. For each of the following postulates, if F satisfies that postulate then so does $\widehat{F}$:* Anonymity, Unanimity, Isomorphism, Directionality, Compatibility.

Corollary 1 *Let F be an interval method. Then $\widehat{F}$ satisfies* Collective Rationality, Anonymity, Unanimity, Isomorphism *and* Directionality.

Hence we have established that, for every interval method F, $\widehat{F}$ satisfies four of the six postulates that characterised the interval methods in Thm. 1, plus a weaker version (*Directionality*) of a fifth (*AF-Independence*). What about the remaining postulate from there, i.e., `in`/`out`-*Plurality*? From Props. 4 and 6 we know that if Y satisfies **(I3)** then $\widehat{F}^Y$ will satisfy *Compatibility* and hence `in`/`out`-*Plurality*. Thus **(I3)** is sufficient to obtain `in`/`out`-*Plurality*. Surprisingly, it turns out this condition is also necessary.

Proposition 7 *Let F^Y be an interval method. The $\widehat{F}^Y$ satisfies* `in`/`out`-Plurality *iff Y satisfies* **(I3)**.

One last question concerns the circumstances under which $\widehat{F}^Y$ will satisfy *(Strong)* `in`/`out`-*Monotonicity*. Since for interval methods we have that *Strong* `in`/`out`-*Monotonicity* holds iff Y is widening, one might expect that an analogous equivalence is preserved for the class of DAUC versions of the interval methods. However this remains open for now.

Acknowledgments

Booth is supported by FNR (DYNGBaT project). Awad benefited from a SINTELNET grant. Thanks to Martin Caminada and anonymous referees for comments.

References

Baroni, P., and Giacomin, M. 2007. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence* 171(10):675–700.

Baroni, P.; Caminada, M.; and Giacomin, M. 2011. An introduction to argumentation semantics. *Knowledge Engineering Review* 26(4):365–410.

Caminada, M., and Pigozzi, G. 2011. On judgment aggregation in abstract argumentation. *Autonomous Agents and Multi-Agent Systems* 22(1):64–102.

Caminada, M. 2006. On the issue of reinstatement in argumentation. In *Proc. JELIA*, 111–123.

Dietrich, F., and List, C. 2007. Judgment aggregation by quota rules: Majority voting generalized. *Journal of Theoretical Politics* 19(4):391–424.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence* 77(2):321–357.

List, C., and Pettit, P. 2002. Aggregating sets of judgments: An impossibility result. *Economics and Philosophy* 18(01):89–110.

List, C., and Puppe, C. 2009. Judgment aggregation: A survey. In *Handbook of Rational and Social Choice*. Oxford University Press.

Rahwan, I., and Tohmé, F. 2010. Collective argument evaluation as judgement aggregation. In *Proc. AAMAS*, 417–424.

Strong Equivalence of Non-Monotonic Temporal Theories*

Pedro Cabalar and **Martín Diéguez**
Dept. of Computer Science
University of Corunna, Spain
{*cabalar,martin.dieguez*}*@udc.es*

Abstract

In this paper we solve the following open problem: we prove that equivalence in the logic of *Temporal Here-and-There* (THT) is not only a sufficient, but also a *necessary* condition for strong equivalence of two *Temporal Equilibrium Logic* (TEL) theories. This result has allowed constructing a tool, `ABSTEM`, that can be used to check different types of equivalence between two arbitrary temporal theories. More importantly, when the theories are not THT-equivalent, the system provides a context theory that makes them behave differently, together with a Büchi automaton showing the temporal stable models that arise from that difference.

Introduction

With the consolidation of Answer Set Programming (ASP) (Brewka, Eiter, and Truszczyński 2011) as a successful paradigm for practical knowledge representation, many examples and benchmarks formalising dynamic scenarios became available. ASP usually treats time as an integer index restricting all reasoning tasks to finite narratives. As a piece of example, consider an extremely simple ASP program where a fluent p represents that a switch is on and q represents that it is off. Moreover, suppose we have freedom to add p arbitrarily at any moment and that either p or q holds initially. A typical ASP representation could be:

$$p(0) \vee q(0) \tag{1}$$
$$p(I{+}1) \leftarrow p(I), \mathit{not}\ q(I{+}1), sit(I) \tag{2}$$
$$q(I{+}1) \leftarrow q(I), \mathit{not}\ p(I{+}1), sit(I) \tag{3}$$
$$p(I) \vee \mathit{not}\ p(I) \leftarrow sit(I) \tag{4}$$

where (1) is the initial state, (2) and (3) are the inertia rules and (4) is a choice rule for p at any situation. Predicate sit would have a finite domain $0 \ldots n$ for some constant $n \geq 0$. Planning for a goal like $p \wedge \neg q$ implies including two constraints for the last situation, $\bot \leftarrow \mathit{not}\ p(n)$ and $\bot \leftarrow q(n)$, and go increasing n until a solution is found. This strategy falls short for many temporal reasoning problems that involve dealing with infinite time such as proving the non-existence of a plan or checking the satisfaction of temporal properties of a given dynamic system. For instance, questions such as "is there a reachable state in which both p and q are false?" or "can we show that whenever p is true it will remain so forever?" can be answered by an analytical inspection of our simple program, but cannot be solved in an automated way.

To overcome these limitations, (Aguado et al. 2013) proposed a temporal extension of *Equilibrium Logic* (Pearce 1996), the best-known logical formalisation of ASP. This extension, which received the name of *Temporal Equilibrium Logic* (TEL), is defined as follows. First, it extends the monotonic basis of Equilibrium Logic, the intermediate logic of *Here-and-There* (HT) (Heyting 1930), by introducing the full syntax of the well-known *Linear-time Temporal Logic* (LTL) (Pnueli 1977). The result of this combination is called *Temporal Here-and-There* (THT). Then, a selection criterion on THT models is imposed, obtaining non-monotonicity in this way. As a result, TEL constitutes a full non-monotonic temporal logic that allows a proper definition of *temporal stable models* for any arbitrary theory in the syntax of LTL. For instance, the ASP program (1)-(4) would be represented in TEL as:

$$p \vee q \tag{5}$$
$$\Box(p \wedge \neg \bigcirc q \rightarrow \bigcirc p) \tag{6}$$
$$\Box(q \wedge \neg \bigcirc p \rightarrow \bigcirc q) \tag{7}$$
$$\Box(p \vee \neg p) \tag{8}$$

where, as usual in LTL, '$\Box$' stands for "always" and '$\bigcirc$' stands for "next." Checking whether p and q can be eventually false would correspond to look for a plan satisfying:

$$\neg\Diamond(\neg p \wedge \neg q) \rightarrow \bot \tag{9}$$

with '$\Diamond$' meaning "eventually." Similarly, to test whether p remains true after becoming true we would add:

$$\Box(p \rightarrow \Box p) \rightarrow \bot \tag{10}$$

and check that, indeed, no temporal stable model exists.

In the past years, several interesting results about TEL were obtained – see survey (Aguado et al. 2013) – but a few important questions about TEL remained unsolved. One of them has to do with the property of *strong equivalence* in TEL. In any Non-Monotonic Reasoning approach, we say that Γ_1 and Γ_2 are *strongly equivalent* when, for any arbitrary theory Γ, both $\Gamma_1 \cup \Gamma$ and $\Gamma_2 \cup \Gamma$ have the same

*This research was partially supported by Spanish MEC project TIN2009-14562-C05-04.
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

selected models (stable models in ASP). (Lifschitz, Pearce, and Valverde 2001) proved that checking equivalence in the logic of HT is a necessary and sufficient condition for strong equivalence in Equilibrium Logic, that is, Γ_1 and Γ_2 are strongly equivalent iff $\Gamma_1 \equiv_{HT} \Gamma_2$. A pair of strong equivalence checkers are, for instance, (Valverde 2004) and (Chen, Lin, and Li 2005). This result for propositional HT was extended to arbitrary first-order theories in (Lifschitz, Pearce, and Valverde 2007). It must be noticed that one direction of this result, the sufficient condition, is actually trivial. As HT is monotonic, $\Gamma_1 \equiv_{HT} \Gamma_2$ implies $\Gamma_1 \cup \Gamma \equiv_{HT} \Gamma_2 \cup \Gamma$ and so, their selected models will also coincide. The real significant result is the opposite direction, namely, that HT-equivalence is also a necessary condition for strong equivalence, as it shows that HT is strong enough as a monotonic basis for Equilibrium Logic. In the case of TEL, (Aguado et al. 2008) implemented a prototype checker and used it on some examples exploiting the trivial direction, i.e., that THT-equivalence is obviously a sufficient condition for strong equivalence in TEL. However, during the past six years, the question whether THT-equivalence was also necessary or not remained unanswered.

In this paper we adapt a result from (Lifschitz, Pearce, and Valverde 2007) to prove that indeed THT-equivalence is a necessary condition for TEL strong equivalence and use this proof, combined with previous theoretical results, to construct a tool, ABSTEM[1], that allows the formal study of arbitrary temporal theories in different ways. First, it implements the technique in (Cabalar and Demri 2011) to compute the temporal stable models of an arbitrary theory Γ, displaying them as a Büchi automaton. Second, given two theories Γ_1 and Γ_2, it allows checking different types of equivalence: LTL-equivalence, weak equivalence (coincidence in temporal stable models) and THT-equivalence which, as said before, corresponds to strong equivalence.

The rest of the paper is organised as follows. In the next section we recall the basic definitions of TEL. After that, we overview the automata-based techniques from (Cabalar and Demri 2011). The next section contains the main theorem. This is followed by an explanation of the implementation together with a practical example. Finally, we include some conclusions.

Temporal Equilibrium Logic

We begin defining the (monotonic) logic of THT as follows. The syntax is defined as in propositional LTL. A temporal *formula* φ can be expressed following the grammar shown below:

$$\varphi ::= \bot \mid p \mid \alpha \wedge \beta \mid \alpha \vee \beta \mid \alpha \rightarrow \beta \mid \bigcirc \alpha \mid \alpha \,\mathcal{U}\, \beta \mid \alpha \,\mathcal{R}\, \beta$$

where p is an atom of some finite signature At, and α and β are temporal formulas in their turn.

The formula $\alpha \,\mathcal{U}\, \beta$ stands for "α *until* β" whereas $\alpha \,\mathcal{R}\, \beta$ is read as "α *release* β" and is the dual of "until." Derived operators such as $\Box$ ("always") and $\Diamond$ ("at some future time") are defined as $\Box\varphi \stackrel{\text{def}}{=} \bot \,\mathcal{R}\, \varphi$ and $\Diamond\varphi \stackrel{\text{def}}{=} \top \,\mathcal{U}\, \varphi$. Other usual propositional operators are defined as follows: $\neg\varphi \stackrel{\text{def}}{=} \varphi \rightarrow \bot$, $\top \stackrel{\text{def}}{=} \neg\bot$ and $\varphi \leftrightarrow \psi \stackrel{\text{def}}{=} (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.

Given a finite propositional signature At, an LTL-*interpretation* $\mathbf{T}$ is an infinite sequence of sets of atoms, $T_0, T_1, \ldots$ with $T_i \subseteq At$ for all $i \geq 0$. Given two LTL-interpretations $\mathbf{H}, \mathbf{T}$ we define $\mathbf{H} \leq \mathbf{T}$ as: $H_i \subseteq T_i$ for all $i \geq 0$. A THT-*interpretation* $\mathbf{M}$ for At is a pair of LTL-interpretations $\langle \mathbf{H}, \mathbf{T} \rangle$ satisfying $\mathbf{H} \leq \mathbf{T}$. A THT-interpretation is said to be *total* when $\mathbf{H} = \mathbf{T}$.

Definition 1 (Satisfaction) *We define when an interpretation $\mathbf{M} = \langle \mathbf{H}, \mathbf{T} \rangle$ satisfies a formula φ at a state $i \geq 0$, written $\mathbf{M}, i \models \varphi$, recursively as follows:*

1. $\mathbf{M}, i \models p$	*iff*	$p \in H_i$, *with p an atom.*
2. $\wedge, \vee, \bot$		*as usual*
3. $\mathbf{M}, i \models \varphi \rightarrow \psi$	*iff*	*for all* $x \in \{\mathbf{H}, \mathbf{T}\}$, $\langle x, \mathbf{T} \rangle, i \not\models \varphi$ *or* $\langle x, \mathbf{T} \rangle, i \models \psi$.
4. $\mathbf{M}, i \models \bigcirc \varphi$	*iff*	$\mathbf{M}, i{+}1 \models \varphi$
5. $\mathbf{M}, i \models \varphi \,\mathcal{U}\, \psi$	*iff*	$\exists k \geq i$ *such that* $\mathbf{M}, k \models \psi$ *and* $\forall j \in \{i, \ldots, k\text{-}1\}, \mathbf{M}, j \models \varphi$.
6. $\mathbf{M}, i \models \varphi \,\mathcal{R}\, \psi$	*iff*	$\forall k \geq i$ *such that* $\mathbf{M}, k \not\models \psi$ *then* $\exists j \in \{i, \ldots, k\text{-}1\}, \mathbf{M}, j \models \varphi$. ■

We say that $\langle \mathbf{H}, \mathbf{T} \rangle$ is a *model* of a theory Γ, written $\langle \mathbf{H}, \mathbf{T} \rangle \models \Gamma$, iff $\langle \mathbf{H}, \mathbf{T} \rangle, 0 \models \alpha$ for all formulas $\alpha \in \Gamma$. It is easy to see that restricting the study to total interpretations, THT-satisfaction collapses to LTL-satisfaction, i.e.:

Proposition 1 (from (Aguado et al. 2013)) $\langle \mathbf{T}, \mathbf{T} \rangle, i \models \varphi$ *in THT iff* $\mathbf{T}, i \models \varphi$ *in LTL.*

An interpretation $\mathbf{M}$ is a *temporal equilibrium model* of a theory Γ if it is a total model of Γ, that is, $\mathbf{M} = \langle \mathbf{T}, \mathbf{T} \rangle \models \Gamma$, and there is no $\mathbf{H} < \mathbf{T}$ such that $\langle \mathbf{H}, \mathbf{T} \rangle \models \Gamma$. An LTL-interpretation $\mathbf{T}$ is a *temporal stable model* (TS-model) of a theory Γ iff $\langle \mathbf{T}, \mathbf{T} \rangle$ is a temporal equilibrium model of Γ. By Proposition 1 it is easy to see that any TS-model of a temporal theory Γ is also an LTL-model of Γ.

As happens in LTL, the set of TS-models of a theory Γ can be captured by a Büchi automaton (Büchi 1962), a kind of ω-automaton (that is, a finite automaton that accepts words of infinite length). In this case, the alphabet of the automaton would be the set of states (classical propositional interpretations) and the acceptance condition is that a word (a sequence of states) is accepted iff it corresponds to a run of the automaton that visits some acceptance state an infinite number of times. As an example, Figure 1 shows the TS-models for the theory (5)-(8) which coincide with sequences of states of the forms $\{q\}^*\{p\}^\omega$ or $\{q\}^\omega$.

Computing Temporal Stable Models of Arbitrary Theories

Based on the techniques introduced in (Cabalar and Demri 2011), the tool ABSTEM constitutes the first implementation capable of computing TS-models for arbitrary temporal formulas, without syntactic restrictions. The method obtains the TS-models of a formula φ by performing several operations on a pair of automata derived from φ. The first automaton, denoted as A_φ, accepts the total THT-models

[1] http://kr.irlab.org/?q=abstem

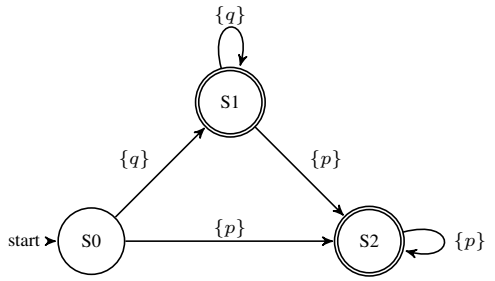


Figure 1: Temporal stable models of theory (5)-(8).

$\langle \mathbf{T}, \mathbf{T} \rangle$ of φ. By Proposition 1 this simply amounts to compute the LTL models $\mathbf{T}$ of φ using an automata construction method for LTL. The second automaton, denoted as $A_{\varphi''}$, accepts the non-total THT-models $\langle \mathbf{H}, \mathbf{T} \rangle$ of φ. The final set of TS-models is obtained from the composition $A_{\varphi} \cap \overline{h(A_{\varphi''})}$ where $h(A_{\varphi''})$ filters out the $\mathbf{H}$ component of non-total models, $\overline{h(A_{\varphi''})}$ is the complementary of $h(A_{\varphi''})$ and finally $\cap$ denotes the automata product.

The computation of $A_{\varphi''}$ is done exploiting a translation of THT into LTL first presented in (Aguado et al. 2008) and directly extrapolating the translation of HT into classical logic in (Pearce, Tompits, and Woltran 2001). It uses an extended signature $\{p' \mid p \in At\}$ so that p' represents the truth of p in $\mathbf{H}$ while p is used for $\mathbf{T}$. The translation of φ, written φ^*, is recursively defined as follows:

- $(p)^* \stackrel{\text{def}}{=} p'$, for any atom $p \in At$
- $(\alpha \rightarrow \beta)^* \stackrel{\text{def}}{=} (\alpha \rightarrow \beta) \wedge (\alpha^* \rightarrow \beta^*)$

being homomorphic for the rest of logical connectives. To impose the restriction $\mathbf{H} < \mathbf{T}$ we further include the axiom:

$$\left(\bigwedge_{p \in At} \Box(p' \rightarrow p) \right) \wedge \left(\bigvee_{p \in At} \Diamond(\neg p' \wedge p) \right) \qquad \textbf{(Ax1)}$$

Automaton $A_{\varphi''}$ is built from formula $\varphi'' \stackrel{\text{def}}{=} \varphi^* \wedge (\textbf{Ax1})$. The operation $h(A_{\varphi''})$ returns a new automaton that results from removing the atoms p' from transitions in $A_{\varphi''}$. This captures the $\mathbf{T}$-components of non-total models; in this way, its complementary automaton $\overline{h(A_{\varphi''})}$ accepts the $\mathbf{T}$ sequences that do not form a non-total model, but perhaps they are not models either. Thus, the final product $A_{\varphi} \cap \overline{h(A''_{\varphi})}$ captures those $\mathbf{T}$ such that $\langle \mathbf{T}, \mathbf{T} \rangle$ is a total model of φ and no non-total model $\langle \mathbf{H}, \mathbf{T} \rangle$ can be formed.

Temporal Strong Equivalence

For simplicity, we assume finite theories and we indistinctly represent them as the conjunction of their formulas.

Theorem 1 (Sufficient condition (Aguado et al. 2008)) *If two temporal formulas α and β are THT-equivalent then they are strongly equivalent in TEL.* ■

To prove the other direction, namely, that THT-equivalence is also a necessary condition for strong equivalence, we begin defining γ_0 as the conjunction of all formulas $\Box(p \vee \neg p)$ for all atoms $p \in At$.

Proposition 2 *Let $\langle \mathbf{H}, \mathbf{T} \rangle$ be a THT interpretation for signature At. If $\langle \mathbf{H}, \mathbf{T} \rangle \models \gamma_0$ then $\mathbf{H} = \mathbf{T}$.*

Corollary 1 *For any formula α for signature At, the LTL-models of $\alpha \wedge \gamma_0$ coincide with its TS-models.* ■

Lemma 1 *Let α and β be two LTL-equivalent formulas and let $\gamma = (\beta \rightarrow \gamma_0)$. Then, the following conditions are equivalent:*

(i) *There exists some $\mathbf{H} < \mathbf{T}$ such that $\langle \mathbf{H}, \mathbf{T} \rangle \not\models \alpha \rightarrow \beta$;*

(ii) *$\mathbf{T}$ is TS-model of $\beta \wedge \gamma$ but not TS-model of $\alpha \wedge \gamma$.* ■

Theorem 2 (Main theorem: necessary condition) *If two temporal formulas α and β are strongly equivalent in TEL then they are THT-equivalent.* ■

Proof We will prove that if α and β are not THT-equivalent then there is some context formula γ for which $\alpha \wedge \gamma$ and $\beta \wedge \gamma$ have different TS-models. Assume first that α and β have different total models, i.e., different LTL-models. Then, the LTL-models of $\alpha \wedge \gamma_0$ and $\beta \wedge \gamma_0$ also differ (since γ_0 is an LTL tautology). But by Corollary 1, LTL-models of these theories are exactly their TS-models, and so, they also differ.

Suppose now that α and β are LTL-equivalent but not THT-equivalent. Then, there is some THT-countermodel $\langle \mathbf{H}, \mathbf{T} \rangle$ of either $(\alpha \rightarrow \beta)$ or $(\beta \rightarrow \alpha)$, and given LTL-equivalence of α and β, the countermodel is non-total, $\mathbf{H} < \mathbf{T}$. Without loss of generality, assume $\langle \mathbf{H}, \mathbf{T} \rangle \not\models \alpha \rightarrow \beta$. By Lemma 1, taking the formula $\gamma = (\beta \rightarrow \gamma_0)$, we get that $\mathbf{T}$ is TS-model of $\beta \wedge \gamma$ but not TS-model of $\alpha \wedge \gamma$. ■

Algorithm 1 StrongEquivalenceTest(α, β)

Require: Two propositional temporal formulas α, β.
Ensure: If α and β are THT-equivalent, it returns **true**. Otherwise, it returns a triple $\langle \gamma, A_1, A_2 \rangle$.

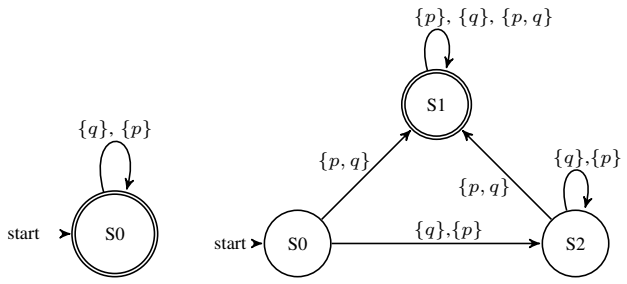
```
A1 := ltl_to_Büchi(α ∧ ¬β)
A2 := ltl_to_Büchi(β ∧ ¬α)
if A1 ≠ ∅ or A2 ≠ ∅ then
    return ⟨γ0, A1, A2⟩
end if
A=ltl_to_Büchi(¬(α → β)* ∧ (Ax1))
if A ≠ ∅ then
    A2 := h(A)
    return ⟨(β → γ0), ∅, A2⟩
end if
A=ltl_to_Büchi(¬(β → α)* ∧ (Ax1))
if A ≠ ∅ then
    A1 := h(A)
    return ⟨(α → γ0), A1, ∅⟩
end if
return true
```

Implementation and a practical example

The procedure for checking strong equivalence in `ABSTEM` is shown in Algorithm 1. It takes two arbitrary propositional temporal formulas α and β and returns either **true**, if they are strongly equivalent, or a triple with a formula γ and two

automata A_1, A_2 otherwise[2]. The meaning of this information is that A_1 captures TS-models of $\alpha \wedge \gamma$ that are not TS-models of $\beta \wedge \gamma$ and, analogously, A_2 captures TS-models of $\beta \wedge \gamma$ that are not TS-models of $\alpha \wedge \gamma$. The procedure uses an auxiliary routine ltl_to_Büchi(φ) to obtain a Büchi automaton from an LTL-formula φ.

As an example of use, let β_1 be our "switch" domain (5)-(8) plus the rule $\Box(\neg p \rightarrow q)$ trying to capture the idea that, when no information on p is available, q becomes true. This new rule is actually a new default for q that interacts with inertia rules (6),(7) destroying somehow their effect. Using ABSTEM to check the TS-models of β_1 we obtain the automaton in Figure 2(a) which corresponds to arbitrary sequences formed with states $\{p\}$ and $\{q\}$. This set of TS-models actually coincides with what one would expect from a formula of the form $\Box(p \vee q)$ since, as happens in ASP, truth minimality converts the disjunction into an exclusive or. So, we may conjecture that β_1 is equivalent to $\alpha_1 = \Box(p \vee q)$. Using ABSTEM we can check that, in fact, α_1 and β_1 have the same TS-models and, furthermore, they are also LTL-equivalent. However, α_1 and β_1 are not THT-equivalent and so, they are not strongly equivalent. The answer displayed this time by ABSTEM is negative and shows the context formula $\gamma_1 = \beta_1 \rightarrow \gamma_0$ plus a file containing the automaton in Figure 2(b). This automaton captures all the TS-models of $\beta_1 \wedge \gamma_1$ that are not TS-models of $\alpha_1 \wedge \gamma_1$.



(a) TS-models of α_1 and β_1.

(b) TS-models of $\beta_1 \wedge \gamma_1$ not of $\alpha_1 \wedge \gamma_1$.

Figure 2: Temporal stable models related to α_1 and β_1.

Conclusions

In this paper we have adapted (Lifschitz, Pearce, and Valverde 2007) to the temporal case, to prove that equivalence in the logic of Temporal Here-and-There (THT), is not only a sufficient but also a *necessary* condition for strong equivalence in Temporal Equilibrium Logic (TEL). Using this proof, we have implemented a system, ABSTEM, for analysing TEL arbitrary theories in different ways. First, ABSTEM constitutes the first tool for computing temporal equilibrium models of any arbitrary temporal theory. Second, it also allows checking three types of equivalence between two arbitrary theories: LTL, weak and strong equivalence. When equivalence is not satisfied, ABSTEM shows counterexamples in the form of a Büchi automaton. Regarding efficiency and scalability, this prototype works satisfactorily for small theories like the ones presented in the paper. It must be noted, however, that checking THT-equivalence is a PSPACE-complete problem.

[2]For further details, see the extended version of this document at http://kr.irlab.org/sites/default/files/papers/kr2014-ex.pdf.

References

Aguado, F.; Cabalar, P.; Pérez, G.; and Vidal, C. 2008. Strongly equivalent temporal logic programs. In *JELIA'08*, volume 5293 of *Lecture Notes in Computer Science*, 8–20.

Aguado, F.; Cabalar, P.; Diéguez, M.; Pérez, G.; and Vidal, C. 2013. Temporal equilibrium logic: a survey. *Journal of Applied Non-Classical Logics* 23(1-2):2–24.

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. Answer set programming at a glance. *Commun. ACM* 54(12):92–103.

Büchi, R. 1962. On a decision method in restricted second-order arithmetic. In *Intl. Congress on Logic, Method and Philosophical Science'60*, 1–11.

Cabalar, P., and Demri, S. 2011. Automata-based computation of temporal equilibrium models. In Vidal, G., ed., *LOPSTR'11*, Lecture Notes in Computer Science. Odense, Denmark: Springer.

Chen, Y.; Lin, F.; and Li, L. 2005. SELP - a system for studying strong equivalence between logic programs. In *LPNMR'05*, volume 3662 of *Lecture Notes in Computer Science*, 442–446.

Heyting, A. 1930. Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse* 42–56.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *Computational Logic* 2(4):526–541.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2007. A characterization of strong equivalence for logic programs with variables. In *LPNMR'07*, volume 4483 of *Lecture Notes in Computer Science*. Springer.

Pearce, D.; Tompits, H.; and Woltran, S. 2001. Encodings for equilibrium logic and logic programs with nested expressions. In *EPIA 2001*, volume 2258 of *Lecture Notes in Computer Science*, 306–320. Springer.

Pearce, D. 1996. A new logical characterisation of stable models and answer sets. In *NMELP'96*, volume 1216 of *Lecture Notes in Artificial Intelligence*. Springer. 57–70.

Pnueli, A. 1977. The temporal logic of programs. In *FOCS'77*, 46–57. IEEE.

Valverde, A. 2004. tabeql: A tableau based suite for equilibrium logic. In *JELIA'04*, volume 3229 of *Lecture Notes in Artificial Intelligence*, 734–737.

Using Answer Set Programming for Solving Boolean Games*

Sofie De Clercq and **Martine De Cock**
Dept. of Applied Mathematics, CS and Statistics
Ghent University, Ghent, Belgium
{SofieR.DeClercq, Martine.DeCock}@ugent.be

Kim Bauters
School of Electronics, Electrical Engineering and CS
Queen's University, Belfast, UK
K.Bauters@qub.ac.uk

Steven Schockaert
School of Computer Science and Informatics
Cardiff University, Cardiff, UK
S.Schockaert@cs.cardiff.ac.uk

Ann Nowé
Computational Modeling Lab
Vrije Universiteit Brussel, Brussels, Belgium
Ann.Nowe@vub.ac.be

Abstract

Boolean games are a framework for reasoning about the rational behaviour of agents, whose goals are formalized using propositional formulas. They offer an attractive alternative to normal-form games, because they allow for a more intuitive and more compact encoding. Unfortunately, however, there is currently no general, tailor-made method available to compute the equilibria of Boolean games. In this paper, we introduce a method for finding the pure Nash equilibria based on disjunctive answer set programming. Our method is furthermore capable of finding the core elements and the Pareto optimal equilibria, and can easily be modified to support other forms of optimality, thanks to the declarative nature of disjunctive answer set programming. Experimental results clearly demonstrate the effectiveness of the proposed method.

Introduction

Boolean games or BGs are a framework for analyzing the behaviour of competing rational agents, whose goals are formalized as propositional formulas. They offer an alternative to normal-form games (NFGs), in which the goals of agents are encoded implicitly in the form of a pay-off function. A BG is a tuple (N, V, π, Φ) with $N = \{1, \ldots, n\}$ a set of agents, V a set of propositional (action) variables, $\pi : N \rightarrow 2^V$ a control assignment function such that $\{\pi(1), \ldots, \pi(n)\}$ forms a partition of V, and Φ a set $\{\varphi_1, \ldots, \varphi_n\}$ of formulas in L_V, i.e. the logical language associated with V (Bonzon et al. 2006). The set $\pi(i)$ or π_i contains the action variables controlled by agent i. Note that each $p \in V$ is controlled by exactly one agent. Every agent i can choose to set $p \in \pi_i$ to true (undertake the action p) or to false (not undertake the action). The formula φ_i is agent i's goal. An interpretation of π_i is called a *strategy* s_i of agent i. A *strategy profile* of a BG is an n-tuple $S = (s_1, \ldots, s_n)$, with s_i a strategy of agent i for every $i \in N$. Since π partitions V and $s_i \subseteq \pi_i, \forall i \in N$, we unambiguously use the set notation $\cup_{i=1}^{n} s_i \subseteq V$ for a strategy profile $S = (s_1, \ldots, s_n)$. With s_{-i} we denote the projection of the strategy profile $(s_1, \ldots, s_n)$ on $N \setminus \{i\}$, i.e. $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$. If s_i' is a strategy of agent i, then (s_{-i}, s_i') is a shorthand for $(s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n)$. In a natural way, agent i's goal induces a Boolean *utility function* u_i for agent i: for every strategy profile S, $u_i(S) = 1$ iff $S \models \varphi_i$, i.e. agent i reaches its goal in S, and $u_i(S) = 0$ otherwise. A straightforward solution concept in game theory is the notion of *pure Nash equilibrium* (PNE). A strategy profile $S = (s_1, \ldots, s_n)$ is a PNE iff, for every agent $i \in N$, s_i is a best response to s_{-i}, i.e. $u_i(S) \geq u_i(s_{-i}, s_i')$ for all strategies $s_i' \subseteq \pi_i$. An alternative solution concept of BGs is the *core*, i.e. the set of strategy profiles that are not blocked by any non-empty coalition $C \subseteq N$. A strategy profile S is blocked by a coalition $C \subseteq N$ ($C \neq \emptyset$) if there exists a strategy profile S' such that all agents outside C undertake the same actions in S and S' and if all agents in C have a strictly higher utility in S' than in S. In this paper we have restricted the discussion to PNEs, cores and Pareto optimality, since these are very common, intuitive solution concepts in the context of BGs, but our reduction to disjunctive answer set programming (ASP) can readily be generalized to alternative solution concepts (Bonzon, Lagasquie-Schiex, and Lang 2012; Dunne and Wooldridge 2012; Dunne et al. 2008).

BGs form a young and active research area (Harrenstein et al. 2001; Bonzon et al. 2006; Bonzon, Lagasquie-Schiex, and Lang 2007; Dunne et al. 2008; Bonzon, Lagasquie-Schiex, and Lang 2012; Wooldridge et al. 2013). The strength of BGs lies in their transparent and compact representation, since the propositional goals make it redundant to explicitly mention the utility for every strategy profile. Therefore, however, computing solutions such as PNEs or core elements is harder than for most other game representations. This is undoubtedly part of the reason why, to the best of our knowledge, no methods for computing the PNEs or core elements of general BGs have been proposed so far.

Currently available methods for solving BGs fall in two categories. First, given that translations from BGs to NFGs

*This research was funded by a Research Foundation-Flanders project.
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

are available, we can readily use solvers for NFGs. However, since such translations are exponential, this circuitous method does not exploit the compactness of BGs. Moreover, its inefficiency increases rapidly with the number of action variables per agent, as shown in the Experiments section. Second, several authors have proposed methods for sub-classes of BGs. For instance, in (Bonzon, Lagasquie-Schiex, and Lang 2007), an algorithm is provided to compute PNEs, but only for BGs with an acyclic irreflexive part of their dependency graph. In (Dunne et al. 2008), a bargaining protocol for BGs allows agents to negotiate in rounds. If the logical operators in the goals are restricted to $\wedge$ and $\vee$, specific negotiation strategies yield a Pareto optimal strategy profile during the first round. Otherwise, obtaining a solution is not guaranteed. More important, none of these techniques can compute the solutions of general BGs. In (De Clercq et al. 2014), the stochastic Win-Stay Lose-probabilistic-Shift algorithm (WSLpS) is used to find solutions of BGs. In case all agents are able to reach their goal simultaneously, WSLpS converges to a Pareto optimal PNE. In the Experiments section, we show that our disjunctive ASP technique outperforms a generalized version of WSLpS for sufficiently difficult problems.

In NFGs, deciding whether a PNE exists is NP-complete when the game is represented by: (i) a set of agents, (ii) a finite set of actions per agents, (iii) a function defining for each agent which other agents may influence their utility, and (iv) the utility of each agent, explicitly given for all joint strategies of the agent and the agents influencing it (Gottlob, Greco, and Scarcello 2003). Deciding whether a BG has a PNE, on the other hand, is Σ_2^{P}-complete[1], even for 2-player zero-sum BGs (Bonzon et al. 2006). Deciding whether the core of a BG is non-empty is also Σ_2^{P}-complete (Dunne et al. 2008). In particular, the problems we tackle in this paper are at the 2nd level of the Polynomial Hierarchy.

The aim of this paper is to introduce a solver that can compute PNEs and core elements of BGs, with the option of additionally enforcing Pareto optimality or taking into account constraints (Bonzon, Lagasquie-Schiex, and Lang 2012) and costs (Dunne et al. 2008). Our solver, available online[2], is based on disjunctive ASP, a form of declarative programming (Baral 2003). In recent years, a range of Σ_2^{P} problems have been identified on which ASP solvers outperform other state-of-the-art methods (Faber, Leone, and Ricca 2005). For our BG solver, we use the state-of-the-art ASP solver DLV to compute answer sets.

Computing PNEs and Core Elements of BGs with Disjunctive ASP Programs

We first show how to compute PNEs of a BG by associating it with a disjunctive ASP program. To this end, we use the saturation technique (Eiter, Gottlob, and Mannila 1997; Baral 2003) to compare answer sets and select the desired solutions. In our case, answer sets correspond to strategy profiles. Intuitively, the idea is to create a program with 3 parts: (i) a part describing strategy profiles and checking the satisfaction of the agents' goals, (ii) a part describing alternative strategies for all agents and checking the corresponding satisfaction of the agents' goals, (iii) a part checking whether or not the agents can improve their utility by deviating from the strategy profile in the first part and selecting the PNEs by saturation.

Let $G = (N, V, \pi, \Phi)$ be a BG. We start by adding the facts $\mathit{agent}(i) \leftarrow$ for every $i \in N$ and $\mathit{action}(p) \leftarrow$ for every $p \in V$ to the first program part $\mathcal{P}_1$. Next, we add a rule to express that every $p \in V$ is either undertaken or not:

$$\mathit{act}(P) \vee \neg \mathit{act}(P) \leftarrow \mathit{action}(P) \tag{1}$$

The capital letter P is a variable, and the rule means that if there exists a constant p such that the literal $\mathit{action}(p)$ is true, then either $\mathit{act}(p)$ or $\neg\mathit{act}(p)$ should be true. For every agent $i \in N$, we add a 'rule' that checks whether its goal is satisfied or not:

$$\mathit{goal}(i) \leftarrow \varphi_i(\mathit{act}(V)) \tag{2}$$

The notation $\varphi_i(\mathit{act}(V))$ represents the formula φ_i in which every $p \in V$ is replaced by $\mathit{act}(p)$. Note that the syntax of ASP does not allow an arbitrary formula in the body, but we translate these 'rules' into valid ASP rules. First our solver transforms the goals of the agents into negation-normal form (NNF). Then 'rules' of the form $\mathit{goal} \leftarrow \varphi$, with φ a propositional formula in NNF, are recursively translated to ASP rules by introducing new atoms. For instance, a 'rule' $\mathit{goal} \leftarrow ((a \wedge b) \vee \neg c) \wedge d$ is replaced with the set of rules $\{\mathit{goal} \leftarrow z, d; z \leftarrow a, b; z \leftarrow \neg c\}$, with z a newly introduced atom. We refer to the solver's implementation[2] for more details.

The second program part is denoted as $\mathcal{P}_2$. We add the following rules to $\mathcal{P}_2$, stating that every action is either undertaken or not in the alternative strategies:

$$\mathit{act'}(P) \vee \mathit{nact'}(P) \leftarrow \mathit{action}(P) \tag{3}$$

Using saturation requires simulating the strong negation $\neg$ with a prefix '*n*'. The intuitive idea is to make all the literals of $\mathcal{P}_2$ true in the third program part, therefore using $\neg\mathit{act'}(P)$ instead of $\mathit{nact'}(P)$ would lead to contradictions. By definition, the utility of every agent who individually deviates from a PNE cannot be strictly higher than its utility in the PNE. To know if the strategy s_i is a best response to s_{-i} for agent i, it suffices that either $u_i(s_{-i}, s_i) = 1$ or $u_i(s_{-i}, s_i') = 0$, $\forall s_i' \subseteq \pi_i$. Therefore it suffices to know whether φ_i is false for i's alternative strategy in $\mathcal{P}_2$. To this end, we add the following 'rules' to $\mathcal{P}_2$ for every $i \in N$:

$$\mathit{ngoal'}(i) \leftarrow n\varphi_i(\mathit{act}(\pi_{-i}), \mathit{nact'}(\neg\pi_i), \mathit{act'}(\pi_i)) \tag{4}$$

with $n\varphi_i$ the notation for a formula, equivalent to $\neg\varphi_i$, in NNF. The notation $n\varphi_i(\mathit{act}(\pi_{-i}), \mathit{nact'}(\neg\pi_i), \mathit{act'}(\pi_i))$ is the formula $n\varphi_i$ in which every occurrence of $p \in \pi_{-i}$ is replaced by $\mathit{act}(p)$, every occurrence of $\neg p$ with $p \in \pi_i$ is replaced by $\mathit{nact'}(p)$ and every other occurrence of $p \in \pi_i$ is replaced by $\mathit{act'}(p)$. As for (2), we translate (4) into valid ASP rules. With X' we denote the set of all newly introduced atoms during the translation of (4).

[1] This is also known as $\mathrm{NP}^{\mathrm{NP}}$-complete, a complexity class at the 2nd level of the Polynomial Hierarchy.

[2] http://www.cwi.ugent.be/BooleanGamesSolver.html.

The third program part $\mathcal{P}_3$ checks whether the strategy of agent i chosen in $\mathcal{P}_1$ is a better response than its alternative strategy in $\mathcal{P}_2$, given the strategies of the other agents in $\mathcal{P}_1$. If so, we derive $\mathit{pleased}(i)$ with the following rules of $\mathcal{P}_3$:

$$\mathit{pleased}(I) \leftarrow \mathit{goal}(I); \quad \mathit{pleased}(I) \leftarrow \mathit{ngoal'}(I) \qquad (5)$$

If all agents have a better response in $\mathcal{P}_1$ than in $\mathcal{P}_2$, we derive *sat* using the following rule of $\mathcal{P}_3$, in which the body literals are compactly denoted in a set. Moreover, we exclude answer sets in which *sat* is not derived:

$$\mathit{sat} \leftarrow \{\mathit{pleased}(i) \mid i \in N\}; \qquad \leftarrow \mathit{not\ sat} \qquad (6)$$

As part of the saturation technique, we set the literals introduced in $\mathcal{P}_2$ to true if *sat* is derived, by adding the following rules to $\mathcal{P}_3$ for every $x \in X'$:

$$\begin{aligned} \mathit{act'}(P) &\leftarrow \mathit{sat}, \mathit{action}(P); \quad \mathit{ngoal'}(I) \leftarrow \mathit{sat}, \mathit{agent}(I) \\ \mathit{nact'}(P) &\leftarrow \mathit{sat}, \mathit{action}(P); \qquad x \leftarrow \mathit{sat} \end{aligned} \qquad (7)$$

Together with (5), (7) implies that all literals of the form *pleased* will also be made true if *sat* is derived. The entire program $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3$ is called the *PNE-program induced by* G. Note that the Σ_2^{P}-complexity of deciding whether there is a PNE in a BG matches the Σ_2^{P}-complexity of our grounded disjunctive ASP programs (Baral 2003). We can prove that this ASP encoding is correct, i.e. that there is a one-to-one correspondence between the answer set of the induced PNE-program and the PNEs of the BG.

Adjusting the ASP program, we obtain an analogous result for core elements. Both encodings are generalized to an extended BG framework, allowing constraints on the possible strategies of the agents (Bonzon, Lagasquie-Schiex, and Lang 2012) and costs, imposed on agents depending on their chosen strategy (Dunne et al. 2008). Similarly, we have extended our encoding to additionally enforce Pareto optimality on the solutions. Due to space constraints, we refer to the implementation of our solver for more details.

Experiments

Experimental Set-up

We evaluate the performance of our ASP based method on instances of a structured problem, which we call *project BGs*. The problem consists of n persons, who can work on several projects. Depending on the project, each person has his preferences concerning the people he might collaborate with — partners — and the people he does not want to collaborate with — anti-partners. Someone who is not a partner, is a non-partner. Partnership and anti-partnership relations are anti-reflexive, not necessarily symmetric and project dependent. The project BGs are inspired by a BG about people being invited to a party (Bonzon, Lagasquie-Schiex, and Lang 2007; Dunne et al. 2008). We include 3 parameters for project BGs: (i) the number of agents, (ii) the number of projects and (iii) the probability of an agent being another agent's partner and the probability of a non-partner being an anti-partner. In a project BG, every agent i controls an action variable p_i^m per project m, interpreted as joining the project. For every project, every agent is randomly assigned one of 13 types, determined by three parameters: (i) personal preference: (a) join the project, (b) do not join, and (c) no preference; (ii) positive partnership (only relevant if level 1 is not (b)): (a) no condition, (b) only join a project if at least one partner joins, and (c) only join a project if all partners join; and (iii) negative partnership (only relevant if level 1 is not (b)): (a) no condition, and (b) only join the project if no anti-partners join. For each agent, we (uniformly) choose the personal preference, and based on this choice we (uniformly) choose among the allowed positive and negative partnerships. The type of an agent influences its goal. For example, agent i of type (c,b,b) has two partners j and k and one anti-partner l for a project m. Then the sub-goal of agent i corresponding to project m becomes $(p_i^m \wedge (p_j^m \vee p_k^m) \wedge \neg p_l^m) \vee \neg p_i^m$. So either agent i joins the project with j or k and without l, or agent i does not join. The overall goal of an agent is formed as the conjunction of the sub-goals corresponding to all the projects. All problem instances, problem generators and solvers have been made available at http://www.cwi.ugent.be/BooleanGamesSolver.html. The measurements have been performed on a dual CPU system with two 2.4GHz Intel Xeon six core E5-4610 processors and 8GB RDIMM.

Since there are currently no solvers for general BGs, there are only few options for a baseline. We consider two baselines: an NFG solver and a heuristic approach. In this paper we use on the one hand a standard approach to compute PNEs of NFGs in ASP (De Vos and Vermeir 1999), which we call the NFG baseline. To use it on BGs, we need to translate the BGs to NFGs, compute the PNEs and translate these PNEs back to BG format. Our second baseline uses the stochastic and iterative algorithm WSLpS on BGs as suggested in (De Clercq et al. 2014), but after each iteration, we verify whether the obtained strategy profile is a PNE or core element. For some classes of BGs, it can be shown that this algorithm is guaranteed to converge to a PNE without checking the best response condition (De Clercq et al. 2014) but in general it should be regarded as a heuristic method. Since WSLpS is stochastic, we run this baseline 25 times per BG.

Results

In a first experiment, we compute 1 PNE for project BGs with a timeout of 5 minutes. The number of agents varies from 5 to 50. For each fixed number of agents, we generate 100 BGs with 1, 2 or 3 projects and a 50% probability that an agent is another agent's partner and that a non-partner agent is an anti-partner. The median computation times, within a 95% confidence interval, are plotted in Figure 1. The results show that dASP strongly outperforms the NFG baseline. It also outperforms WSLpS if the number of agents is sufficiently high. The higher the number of projects, the smaller the number of agents at which dASP starts outperforming WSLpS. Note that for a wide range of parameters, dASP still succeeds in finding a solution for all problem instances, whereas both baselines consistently time out.

Secondly, we compute core elements instead of PNEs. Since the NFG baseline is not capable of computing core elements, we limit the experiments to WSLpS and dASP. For the same range of parameters as before, we compute

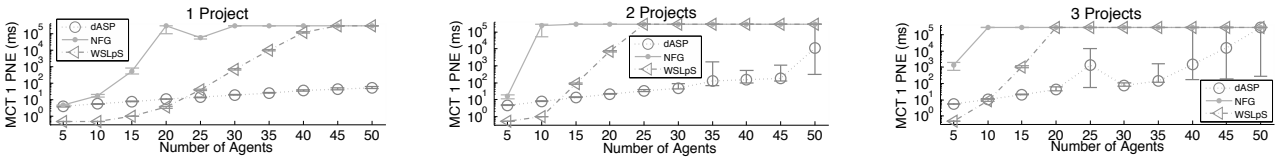


Figure 1: Median computation time (ms) of 1 PNE in project BGs.

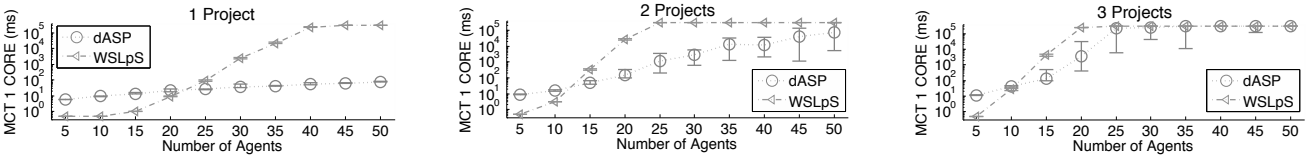


Figure 2: Median computation time (ms) of 1 core element in project BGs.

1 core element in 100 project BGs. The results are plotted in Figure 2. We observe that dASP consistently outperforms WSLpS for the more difficult problems instances.

Finally we investigate the scalability of our approach on project BGs with 2 projects, extended with costs and constraints. The constraints enforce that an agent can join at most 1 project. For both projects, an individual cost of 0, 1 or 2 is randomly assigned to joining the project. Not joining involves no costs. There are between 5 and 50 agents and we consider 100 BGs for each parameter setting. With a timeout of 10 minutes, respectively 1 PNE, 1 Pareto optimal PNE (POPNE), 1 core element or 1 Pareto optimal core element (POCORE) is computed. Note that no comparison with the baselines is made, since those are not equipped to deal with costs and constraints, nor to compute Pareto optimal solutions. The results are plotted in Figure 3. As expected, computing a PNE takes less time than computing a core element and enforcing Pareto optimality further increases the computation time of a PNE. For core elements, enforcing Pareto optimality does not demand more time. Although the computational cost is generally higher for stronger solution concepts or for BGs involving costs and constraints, our method remains suitable for medium-sized problems.

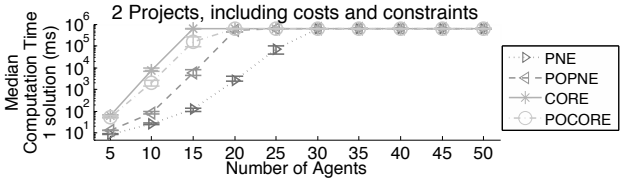


Figure 3: Computation of 1 solution in project BGs.

Conclusion

We proposed a method based on disjunctive ASP for finding the solutions of a BG. To the best of our knowledge, this is the first solver which is capable of handling general BGs (apart from methods based on an exponential translation to NFGs). Our solver can compute (Pareto optimal or arbitrary) PNEs and core elements, even in the presence of costs and constraints. Using a class of structured problems, called project BGs, we showed that our disjunctive ASP based method outperforms both a NFG based method and a heuristic method called WSLpS on sufficiently difficult problems, although for small instances WSLpS was faster on average. We also demonstrated the effectiveness of our method to find solutions under the presence of costs and constraints, and when Pareto optimality is required.

References

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. New York: Cambridge University Press.

Bonzon, E.; Lagasquie-Schiex, M.-C.; Lang, J.; and Zanuttini, B. 2006. Boolean games revisited. In *Proc. ECAI*, 265–269. ACM.

Bonzon, E.; Lagasquie-Schiex, M.-C.; and Lang, J. 2007. Dependencies between players in Boolean games. In *Proc. ECSQARU*, 743–754. Springer.

Bonzon, E.; Lagasquie-Schiex, M.-C.; and Lang, J. 2012. Effectivity functions and efficient coalitions in Boolean games. *Synthese* 187:73–103.

De Clercq, S.; Bauters, K.; Schockaert, S.; Mihaylov, M.; De Cock, M.; and Nowé, A. 2014. Decentralized computation of Pareto optimal pure Nash equilibria of Boolean games with privacy concerns. In *Proc ICAART '14*. Forthcoming.

De Vos, M., and Vermeir, D. 1999. Choice logic programs and Nash equilibria in strategic games. In *Proc. CSL*, 266–276. Springer.

Dunne, P. E., and Wooldridge, M. 2012. Towards tractable Boolean games. In *Proc. AAMAS*, 939–946. IFAAMAS.

Dunne, P.; van der Hoek, W.; Kraus, S.; and Wooldridge, M. 2008. Cooperative Boolean games. In *Proc. AAMAS*, volume 2, 1015–1022. IFAAMAS.

Eiter, T.; Gottlob, G.; and Mannila, H. 1997. Disjunctive datalog. *ACM T. Database Syst.* 22(3):364–418.

Faber, W.; Leone, N.; and Ricca, F. 2005. Solving hard problems for the 2nd level of the Polynomial Hierarchy: Heuristics and benchmarks. *Intelligenza Artificiale* 2(3):21–28.

Gottlob, G.; Greco, G.; and Scarcello, F. 2003. Pure Nash equilibria: hard and easy games. In *Proc. TARK*, 215–230. ACM.

Harrenstein, P.; van der Hoek, W.; Meyer, J.-J.; and Witteveen, C. 2001. Boolean games. In *Proc. TARK*, 287–298. Morgan Kaufmann Publishers Inc.

Wooldridge, M.; Endriss, U.; Kraus, S.; and Lang, J. 2013. Incentive engineering for Boolean games. *Artif. Intell.* 195:418–439.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Towards a Knowledge Level Analysis of Forgetting

James P. Delgrande
School of Computing Science
Simon Fraser University
Burnaby, B.C.
V5A 1S6 Canada
jim@cs.sfu.ca

Abstract

Forgetting has been addressed in various areas in KR, including classical logic, logic programming, modal logic, and description logics. Here, we view forgetting as an abstract operator, independent of the underlying logic. We argue that forgetting amounts to a reduction in the signature of a language of a logic, and that the result of forgetting elements of a signature in a theory is the set of logical consequences over the reduced language. This definition offers several advantages. It provides a uniform approach to forgetting, applicable to any logic with a well-defined consequence relation. Obtained results are thus applicable to all subsumed formal systems, and typically are obtained much more straightforwardly. The approach also leads to insights with respect to specific logics: forgetting in first-order logic is somewhat different from the accepted approach; and the definition applied to logic programs yields a new syntax-independent notion of forgetting.

Introduction

Forgetting has been studied in the context of various logical systems, including classical propositional logic (PC), first-order logic (FOL), logic programming, modal logics, and description logics. While there is no generally-agreed upon definition, a common intuition is that in forgetting an agent becomes ignorant of, or unaware of, some part of its language.

Forgetting potentially has several pragmatic uses. For example, forgetting the part of a knowledge base (KB) that is irrelevant to a query may yield more efficient query-answering. Forgetting may provide a formal account of predicate hiding. As well, forgetting may be used in summarising a KB or reusing part of a KB or in clarifying relations between predicates.

A typical approach to addressing forgetting is to specify a host logic, give intuitions, and then provide a formal definition. Consequently, forgetting has been addressed with respect to various logics and with differing intuitions, but with no consensus as to what it means as a general concept.

In this paper, forgetting is addressed as an abstract belief change operator, independent of any specific formal system. The goal is to investigate forgetting at the *knowledge level* and thus independent of syntax. Our thesis is that forgetting amounts to a reduction in the *signature* of a language of a logic. This approach to forgetting is not new; in fact it is, implicitly or explicitly, the most common definition for the term.

This means that, for a given logic over a language with signature σ, forgetting $\sigma' \subseteq \sigma$ from a KB K is the set of consequences of K expressible over $\sigma \setminus \sigma'$. This definition is very simple but offers several advantages. Foremost, it provides a uniform approach, applicable to any logic with a well-defined consequence relation. By taking a general view, various issues are clarified, unified, and simplified. Obtained results are obviously applicable to all subsumed formal systems, and many results follow immediately. An inference-based perspective leads to simpler techniques for computing forgetting in some cases. Since forgetting results in a theory, the main computational challenge is to, if possible, determine a finite representation of forgetting, or a *uniform interpolant*. Notably, a uniform interpolant may not exist (e.g., in FOL) whereas forgetting remains well-defined.

The next section reviews related work while the third section presents notation and definitions. The fourth section presents the general approach. The next section briefly discusses applying the approach to specific logics. This is followed by a concluding section.

Background

The best-know definition of forgetting is due to George Boole, wherein forgetting an atom p from formula ϕ is given by $\phi[p/\top] \vee \phi[p/\bot]$. [Lin and Reiter, 1994] addresses forgetting in FOL. In logic programming, work has focussed on computing forgetting via program transformations. E.g.,[1] [Eiter and Wang, 2008] give an algorithm for computing forgetting in disjunctive logic programming. These approaches are syntactic, in that forgetting may produce different results for two programs that are strongly equivalent.

In modal logic, most work on forgetting *per se* has dealt with epistemic logic, although some work (e.g. [Ghilardi, 1995]) addresses uniform interpolation. [van Ditmarsch *et al.*, 2009] present an approach wherein "becoming ignorant" is modelled as an event in a dynamic epistemic logic. According to [van Ditmarsch *et al.*, 2009], there is no hard notion of what it means to forget something, whereas in the

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] The set of references is limited due to space contraints.

present approach, forgetting is uniquely defined.

In description logic, work has also focussed on uniform interpolation. E.g. [Konev *et al.*, 2009] addresses uniform interpolation in $\mathcal{EL}$. Characterizations of uniform interpolants are given via bisimulations between models. Work specifically on forgetting includes [Wang *et al.*, 2010], which addresses the problem in $\mathcal{ALC}$.

Formal Preliminaries

For an understood signature, a logic for a language is given in terms of a consequence relation $\vdash$. The relation $\vdash$ is assumed to be a Tarskian consequence relation; that is, it satisfies reflexivity, cut, and monotony. We allow sets of formulas on the right hand side of $\vdash$. $\Gamma \leftrightarrow \Psi$ abbreviates $\Gamma \vdash \Psi$ and $\Psi \vdash \Gamma$. As well, $\mathcal{C}n(\Gamma) = \{\phi \mid \Gamma \vdash \phi\}$.

The set of interpretations is denoted by Ω, where the logic and signature are understood from the context. $w \models \phi$ denotes that ϕ is true in interpretation w. $Mod(\phi)$ is the set of models of ϕ. $\Gamma \models \phi$ just if $Mod(\Gamma) \subseteq Mod(\phi)$. For the logics we consider, we have that $\Gamma \models \phi$ iff $\Gamma \vdash \phi$.

σ (σ', etc.) denotes a signature for a (given) logic. $\mathcal{L}_\sigma$ is the language defined over σ, where the language and consequence relation are clear from the context. $Sig(\Gamma)$ denotes the signature of Γ, i.e. the nonlogical symbols mentioned in formulas in Γ. We also use $Sig(w)$ to denote the signature associated with interpretation w.

The above notions may where appropriate be subscripted by a signature. E.g. for an understood logic, Ω_σ denotes the set of interpretations over, or restricted to, σ. In particular, $\mathcal{C}n_\sigma(\Gamma) = \{\phi \mid \Gamma \vdash \phi \text{ where } \phi \in \mathcal{L}_\sigma\}$

Let σ and σ' be signatures where $\sigma \subset \sigma'$. Then σ is a *reduct*[2] of σ', and σ' is an *expansion* of σ. If also $w \in \Omega_\sigma$, $w' \in \Omega_{\sigma'}$, and w and w' agree on the interpretation of symbols in σ then w is the *σ-reduct* of w', and w' is a *σ'-expansion* of w. $w'_{|\sigma}$ is the reduct of w' with respect to σ whereas $w_{\uparrow\sigma'}$ is the set of expansions of w with respect to σ'. This notation extends to sets of models and formulas in the obvious way. E.g. for $\sigma \subseteq \sigma'$, $\Gamma_{|\sigma}$ is the set of formulas in Γ with signature in σ.

The Approach

Our central definition is the following.

Definition 1 *Assume some fixed logic with language $\mathcal{L}$ and signature σ. Let $\Gamma \subseteq \mathcal{L}_\sigma$. The result of* forgetting *σ' in Γ, denoted $\mathcal{F}(\Gamma, \sigma')$, is given by:*

$$\mathcal{F}(\Gamma, \sigma') = \mathcal{C}n_\sigma(\Gamma) \cap \mathcal{L}_{\sigma \setminus \sigma'}.$$

While this definition is very simple, it specifies a unique, unambiguous operator, applicable to any "reasonable" logic. Notably, it applies to logics for which an interpolation theorem does not hold, such as classical FOL, S4, and various relevance logics and description logics. The characterization in Definition 1 is at the *knowledge level* and yields a theory. A key question then is to determine those instances of $\mathcal{F}$ that may be finitely characterized, i.e. for which a uniform interpolant exists.

$\mathcal{F}(\Gamma, \sigma')$ results in a reduced language (unless $\sigma \cap \sigma' = \emptyset$). We can re-express forgetting in the original language (as many approaches do) via:

$$\mathcal{F}_O(\Gamma, \sigma') = \mathcal{C}n_\sigma(\mathcal{F}(\Gamma, \sigma'))$$

The following results are elementary, but show that the definition of forgetting has the "right" properties.

Proposition 1 *Let Γ and Γ' be sets of sentences.*

1. $\Gamma \vdash \mathcal{F}(\Gamma, \sigma)$
2. *If* $\Gamma \leftrightarrow \Gamma'$ *then* $\mathcal{F}(\Gamma, \sigma) \leftrightarrow \mathcal{F}(\Gamma', \sigma)$
3. $\mathcal{F}(\Gamma, \sigma) = \mathcal{C}n_{\sigma'}(\mathcal{F}(\Gamma, \sigma))$ *where* $\sigma' = Sig(\Gamma) \setminus \sigma$.
4. $\mathcal{F}(\Gamma, \sigma_1 \cup \sigma_2) = \mathcal{F}(\mathcal{F}(\Gamma, \sigma_1), \sigma_2))$
5. Γ *is a* conservative extension *of* $\mathcal{F}(\Gamma, \sigma)$.

Thus, forgetting yields no new consequences; it is independent of syntax and results in a deductively-closed theory. Part 4 shows that forgetting is decomposable with respect to a signature; hence it is commutative with respect to its second argument. Last, in forgetting, no results over the reduced signature are lost.

The next results give a model-based characterisation.

Proposition 2 *Let$\sigma' \subseteq \sigma$, and let $\Gamma \subseteq \mathcal{L}_\sigma$.*

1. $Mod_{\sigma \setminus \sigma'}(\mathcal{F}(\Gamma, \sigma')) = Mod_\sigma(\Gamma)_{|(\sigma \setminus \sigma')}$
2. $Mod_\sigma(\mathcal{F}(\Gamma, \sigma')) = (Mod_\sigma(\Gamma)_{|(\sigma \setminus \sigma')})_{\uparrow\sigma}$

Thus the models of $\mathcal{F}(\Gamma, \sigma')$ are exactly the models of Γ restricted to the signature $\sigma \setminus \sigma'$. The second part expresses forgetting with respect to the original signature. The form in Proposition 2.2 has frequently appeared in the literature for characterising forgetting.

[Zhang and Zhou, 2009] give four postulates characterising their approach to forgetting in S5. An analogous result follows here.

Definition 2 *Signature σ is* irrelevant *to Γ, $IR(\Gamma, \sigma)$, iff there is Γ' such that $\Gamma \leftrightarrow \Gamma'$ and $Sig(\Gamma') \cap \sigma = \emptyset$.*

Proposition 3 $\Gamma' = \mathcal{F}_O(\Gamma, \sigma')$ *iff*

1. $\Gamma \vdash \Gamma'$
2. *If* $IR(\phi, \sigma')$ *and* $\Gamma \vdash \phi$ *then* $\Gamma' \vdash \phi$
3. *If* $IR(\phi, \sigma')$ *and* $\Gamma \nvdash \phi$ *then* $\Gamma' \nvdash \phi$
4. $IR(\Gamma', \sigma')$

Applying the Framework

We next instantiate the approach to specific logical systems, including PC, FOL, a relevance logic, and disjunctive logic programs. Given space constraints, some familiarity with these approaches is assumed.

Propositional Logic For this part, $\mathcal{L}$ is the language of PC, and $\mathcal{P}$, a signature, is a set of propositional atoms. For set of formulas Γ, Γ_{CNF} is the clause form of Γ, expressed as a set of sets of literals. $Res(S, p)$ is the set of resolvants of clauses in S with respect to p; e.g. for $S = \{\{p, q\}, \{\neg p, r\}, \{\neg p, s\}, \{t\}\}$, $Res(S, p) = \{\{q, r\}, \{q, s\}\}$. We obtain:

[2]This term is standard in model theory. It should not be confused with the term *reduct* in answer set programming.

Theorem 1 *Let $\Gamma \subseteq \mathcal{L}$ and $p \in \mathcal{P}$.*

1. $\mathcal{F}(\Gamma, p) \leftrightarrow \Gamma[p/\top] \vee \Gamma[p/\bot]$ *for finite* Γ.
2. $\mathcal{F}(\Gamma, p) \leftrightarrow \Gamma_{CNF|(\mathcal{P}\setminus\{p\})} \cup Res(\Gamma_{CNF}, p)$.

The first part shows that, restricted to PC, the result of forgetting according to Definition 1 coincides with the Boole definition. The second part gives an alternative means of computing forgetting that offers several advantages over the Boole definition. The Boole definition, while easily computable, only works for a finite set of formulas and results in a disjunction as the top-level connective in a KB (which in general may be awkward to work with). It does not readily extend to other logics including, as we will argue, FOL.

For the second means of computing forgetting, there is no need to require a finite KB, and the resulting KB is in a more useful form. However, it may result in a quadratic (in the number of clauses mentioning p) blowup of the KB. Other properties are inherited from Proposition 1 and 3. The model-theoretic characterisation of forgetting in PC is a corollary to Proposition 2.

First-Order Logic In FOL, nonlogical symbols are of two sorts: predicate and function symbols. [Lin and Reiter, 1994] define the forgetting of a predicate symbol via interpretations, using a definition analogous to Proposition 2. Forgetting a constant symbol doesn't appear to have been considered in the literature (nor for that matter, arbitrary function symbols). For forgetting a constant symbol we have the following.

Proposition 4 *Let $\Gamma \subset \mathcal{L}_\sigma$ be finite, c a constant, and $\Gamma_c = \{\phi \in \Gamma \mid c \in \sigma(\phi)\}$. For x not appearing in Γ we have:*

$$\mathcal{F}(\Gamma, c) \leftrightarrow (\Gamma \setminus \Gamma_c) \cup \{\exists x \wedge_{\mu \in \Gamma_c} \mu[c/x]\}.$$

Lin and Reiter also discuss forgetting a ground atomic formula. Our definition of forgetting is inapplicable in this case. Moreover, if Definition 1 were modified to admit ground atomic formulas, it would produce unacceptable results. E.g. consider forgetting that John is a student, $st(J)$, in Γ by taking the set of consequences $\Gamma' \subseteq \Gamma$ that don't mention $st(J)$. Then Γ' would still contain $\exists x(x = J \wedge st(x))$, since this is a consequence of Γ that does not mention $st(J)$. So not only does Γ' still entail $st(J)$, but Γ' is also not a (deductively-closed) theory.

We suggest that $st(J)$, or any well-formed formula for that matter, is not an appropriate object for forgetting: $st(J)$ is an *assertion* and the removal of this fact is in fact the removal of a *proposition*. In such a case the appropriate belief change operation is *contraction*. See the full paper for details.

Relevance Logic We next consider the relevance logic of *first degree entailment* $\mathbf{E}_{fde}$ [Anderson and Belnap Jr., 1975], a logic weaker than classical PC. $\mathbf{E}_{fde}$ provides an example where the standard Boole definition of forgetting is unsuitable. A formula of $\mathbf{E}_{fde}$ is of the form $\phi \rightarrow \psi$ where ϕ, ψ are propositional formulas formed using the standard connectives $\wedge$, $\vee$, $\neg$ but not mentioning $\rightarrow$. In $\mathbf{E}_{fde}$, the so-called "paradoxes of implication" such as $p \wedge \neg p \rightarrow \phi$ and $\phi \rightarrow p \vee \neg p$ are missing. See [Anderson and Belnap Jr., 1975] for an axiomatisation and semantic theory.

If we write $\phi \rightleftarrows \psi$ for $\phi \rightarrow \psi$ and $\psi \rightarrow \phi$, then it can be shown that $\phi \rightleftarrows \phi'$ holds where ϕ' is ϕ in CNF. Moreover, for ϕ, ψ in CNF, determining whether $\phi \rightarrow \psi$ holds can be easily determined: the entailment holds just when for each clause (disjunction) C in ψ there is a clause in ϕ whose literals are are among those in C.

With these facts, it is straightforward to define forgetting in $\mathbf{E}_{fde}$. For Γ a set of propositional formulas formed using connectives $\wedge$, $\vee$, $\neg$, define $\mathcal{C}n(\Gamma)$ by:

$$\mathcal{C}n(\Gamma) = \{\phi \mid \Gamma \rightarrow \phi\}.$$

Then Definition 1 can be applied. We obtain:

Theorem 2 *Let Γ be a set of propositional formulas formed using $\wedge$, $\vee$, $\neg$. Then*

$$\mathcal{F}(\Gamma, p) \rightleftarrows \{C \in \Gamma_{CNF} \mid \textit{ not: } p \rightarrow C\}.$$

Thus forgetting p from Γ is characterised by just those clauses in Γ, in CNF, that don't mention p. So forgetting in this case is a trivial operation once a KB is in CNF.

Logic Programming We turn next to answer set programming (ASP) [Gelfond and Lifschitz, 1988]. A (*disjunctive*) *logic program* (DLP) over a set of atoms $\mathcal{A}$ is a finite set of rules of the form

$$a_1; \ldots; a_m \leftarrow b_1, \ldots, b_n, {\sim}c_1, \ldots, {\sim}c_p$$

where $a_i, b_j, c_k \in \mathcal{A}$ and $0 \leq m, n, p$.

Operators ';' and ',' express disjunction and conjunction respectively. For atom a, ${\sim}a$ is (default) negation. A program induces 0, 1, or more *answer sets*, roughly, minimal sets of atoms that satisfy the rules. The difficulty is that a program is nonmonotonic with respect to its answer sets. E.g., the program $\{q \leftarrow {\sim}p\}$ has answer set $\{q\}$ while $\{q \leftarrow {\sim}p,\ p\}$ has answer set $\{p\}$.

More recently ASP has been put on a monotonic footing with a notion called *strong equivalence* [Lifschitz *et al.*, 2001] that provides an account of logical equivalence for programs. Subsequently, [Wong, 2008] has provided an inferential system for disjunctive rules that preserves strong equivalence. The set of inference rules may be found in the aforecited reference; the key point is that these rules allow a consequence relation $\mathcal{C}n(\cdot)$ to be defined, and consequently Definition 1 can be used.

Moreover, we can define a notion analogous to Res in Theorem 1 for computing *forget* in a DLP. That is, for a DLP P and atom a, we can define $ResLP(P, a)$ to be a set of rules such that

1. each rule in $ResLP(P, a)$ is obtained by a single application of an inference rule to rules in P; and
2. no rule in $ResLP(P, a)$ mentions a.

Then we obtain the result:

Theorem 3 *Let P be a DLP and $a \in \mathcal{A}$. Then:*

$$\mathcal{F}(P, a) \leftrightarrow P_{|(\mathcal{A}\setminus\{a\})} \cup ResLP(P, a).$$

The relation $\leftrightarrow$ is that of strong equivalence. Since we inherit the results of Proposition 1 and 3, the results of forgetting are independent of syntax, even though the expression

on the right hand side of Theorem 3 is a set of rules obtained by transforming and selecting rules in P. As in PC, forgetting an atom results in at worst a quadratic blowup in the size of the program. This then results in an interesting operator: As indicated in the Background section, there has been substantial effort in addressing forgetting in ASP. Such previous work is *syntactic*, in that the result of forgetting depends on the form of the underlying logic program, whereas the current approach is independent of how a program is expressed.

Other Logics Clearly the approach can be extended to other classes of logics. Thus, there is no obstacle in principle to providing a syntactic definition (analogous to Theorem 3) for forgetting in S5, for example by exploiting the inferential system RS5 of [Enjalbert and Fariñas del Cerro, 1989]. Further, the central definitions are applicable to description logics. The current approach then provides a definition of forgetting regardless of whether a particular system has the uniform interpolation property or not.

Conclusion

We have presented an account of forgetting as an abstract, knowledge level, belief change operator, in which forgetting constitutes a reduction in the signature of a KB. By taking a general view, various issues are clarified, unified, and simplified. To this end, the central definition is applicable to any logic with a well-defined consequence relation, and the results obtained are applicable to all subsumed logics. For a given logic and signature, we obtain a unique, syntax-independent operation of forgetting, independent also of a domain of application. Thus, to this last point, forgetting doesn't involve losing information about a domain *per se* but rather involves losing the ability to express, or represent, information about the domain.

We argue that this view leads to insights with respect to specific logics. In the literature, there is no universally-agreed on use of the term *forgetting*, and it has been used for conceptually different phenomena. Some approaches describe forgetting as "becoming ignorant" or base intuitions on forgetting knowledge. Such approaches would appear to address some form of *contraction* in an agent's knowledge.

We have considered "instantiations" of forgetting in PC, FOL, a relevance logic, and in disjunctive logic programs. Forgetting in FOL is somewhat different from the accepted approach; forgetting is readily definable in at least some relevance logics; and the definition applied to disjunctive logic programs yields a new syntax-independent definition of forgetting. Forgetting has most often been identified with uniform interpolation. Indeed there is a close relation: a uniform interpolant is a syntactic object, which may or may not exist for a given logic. Forgetting, in contrast, is well-defined for any "reasonable" logic. If a uniform interpolant exists, it corresponds to a (finite) representation of the result of forgetting. In fact, the key computational question for forgetting is whether a uniform interpolant exists and, if so, how it is determined.

References

A.R. Anderson and N.D. Belnap Jr. *Entailment: The Logic of Relevance and Necessity, Vol. I.* Princeton University Press, 1975.

Thomas Eiter and Kewen Wang. Forgetting in answer set programming. *Artificial Intelligence*, 172(14):1644–1672, 2008.

Patrice Enjalbert and Luis Fariñas del Cerro. Modal resolution in clausal form. *Theoretical Computer Science*, 65(1):1–33, 1989.

Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*, pages 1070–1080. The MIT Press, 1988.

Silvio Ghilardi. An algebraic theory of normal forms. In *Annals of Pure and Applied Logic*, volume 71, pages 189–245, 1995.

Boris Konev, Dirk Walther, and Frank Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 830–835, 2009.

V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.

F. Lin and R. Reiter. Forget it! In *AAAI Fall Symposium on Relevance*, New Orleans, November 1994.

Hans P. van Ditmarsch, Andreas Herzig, Jérôme Lang, and Pierre Marquis. Introspective forgetting. *Synthese*, 169(2):405–423, 2009.

Zhe Wang, Kewen Wang, Rodney W. Topor, and Xiaowang Zhang. Tableau-based forgetting in $\mathcal{ALC}$ ontologies. In *ECAI'10*, pages 47–52, 2010.

Ka-Shu Wong. Sound and complete inference rules for SE-consequence. *Journal of Artificial Intelligence Research*, 31(1):205–216, January 2008.

Yan Zhang and Yi Zhou. Knowledge forgetting: Properties and applications. *Artificial Intelligence*, 173(16-17):1525–1537, November 2009.

A Psychology-Inspired Approach to Automated Narrative Text Comprehension

Irene-Anna Diakidoy
University of Cyprus
eddiak@ucy.ac.cy

Antonis Kakas
University of Cyprus
antonis@cs.ucy.ac.cy

Loizos Michael
Open University of Cyprus
loizos@ouc.ac.cy

Rob Miller
University College London
rsm@ucl.ac.uk

Abstract

We report on an ongoing research program to develop a formal framework for automated narrative text comprehension, bringing together know-how from research in Artificial Intelligence and the Psychology of Reading and Comprehension. It uses argumentation to capture appropriate solutions to the frame, ramification, and qualification problems, and their generalizations as required for text comprehension. In this first part of the study we concentrate on the central problem of integration of the explicit information from the text narrative with the reader's implicit commonsense world knowledge, and the associated tasks of elaboration and revision.

Introduction

Text comprehension has long been identified as a key test for Artificial Intelligence (AI), with a central position in many forms of the Turing Test, and enormous benefits in human-computer interaction. The rise of computing over the Internet, where so much data is in the form of textual information, has given even greater importance to this topic. We report on a research program aiming to learn from the extensive text comprehension work in Psychology, and draw guidelines for developing frameworks for automated narrative text comprehension and, in particular, *story comprehension* (SC).

Our research program brings together know-how from research in Psychology and AI — in particular, our understanding of Reasoning about Actions and Change and Argumentation — to provide a formal representational and computational framework for SC that can be empirically evaluated and iteratively developed. This empirical evaluation, which forms an important part of the program, is based on the following methodology: *(i)* set up a corpus of stories with questions to test different aspects of story comprehension; *(ii)* harness the world knowledge used by human readers for comprehension; *(iii)* use this world knowledge in our framework and automated system, and compare its comprehension behaviour with that of the human readers.

The Psychology of Story Comprehension

Comprehending text entails the construction of a mental representation of the information contained in the text. However, no text specifies clearly and completely all the implications of text ideas or the relations between them. Therefore, comprehension depends on the ability to mentally represent the text-given information and to generate ***bridging and elaborative inferences*** that connect and elaborate text ideas resulting in a mental or ***comprehension model*** of the story. Inference generation is necessary in order to comprehend any text as a whole, i.e., as a single network of interconnected propositions instead of as a series of isolated sentences, and to appreciate the suspense and surprise that characterize narrative texts or stories, in particular (Brewer and Lichtenstein 1982; McNamara and Magliano 2009).

Although inference generation is based on the activation of background ***world knowledge***, the process is constrained by text information. Concepts encountered in the text activate related conceptual knowledge in the readers' long-term memory (Kintsch 1988). Nevertheless, at any given point in the process, only a small subset of all the possible knowledge-based inferences remain activated and become part of the mental representation: those that connect and elaborate text information in a way that contributes to the ***coherence*** of the mental model (Rapp and den Broek 2005; McNamara and Magliano 2009). Inference generation is a task-oriented process that follows the principle of ***cognitive economy*** enforced by a limited-resource cognitive system.

However, the results of this coherence-driven selection mechanism can easily exceed the limited working memory capacity of the human cognitive system. Therefore, coherence on a more global level is achieved through ***higher-level integration*** processes that operate to create macro-propositions that generalize or subsume a number of text-encountered concepts and the inferences that connected them. In the process, previously selected information that maintains few connections to other information is dropped from the mental model. This results in a more consolidated network of propositions that serves as the new anchor for processing subsequent text information (Kintsch 1998).

Comprehension also requires an iterative general ***revision mechanism*** of the readers' mental model. The feelings of suspense and surprise that stories aim to create are achieved through discontinuities or changes (in settings, motivations, actions, or consequences) that are not predictable or are wrongly predictable solely on the basis of the mental model created so far. Knowledge about the structure and the func-

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

tion of stories leads readers to expect discontinuities and to use them as triggers to revise their mental model (Zwaan 1994). A change in time or setting in the text may serve as a clue for revising parts of the mental model while other parts remain and are integrated with subsequent text information.

The interaction of bottom-up and top-down processes for coherence carries the possibility of ***different*** but equally ***legitimate or successful comprehension*** outcomes, due to the qualitative and quantitative differences in the conceptual and mental state knowledge of different readers. Comprehension is successful if these differences are primarily in elaboration.

Scope of the Present Paper

We concentrate on the development of an appropriate Reasoning about Actions and Change and Default Reasoning framework for representing narratives and the *world knowledge* needed for the central comprehension process of *synthesizing and elaborating* the explicit text information with new inferences, and *revising* them in the presence of new narrative information. Our working hypothesis is that higher level features of comprehension, such as coherence and cognitive economy, can be tackled on top of the framework we develop. We are also assuming as solved the issue of correctly parsing the natural language of the text into some information-equivalent structured (e.g., logical) form of the story narrative, without discounting the importance of this latter problem, nor the possibility of the need to be tackled in conjunction with the problems on which we are focusing.

An excerpt from one of the stories used for the initial evaluation of our approach, is used as our running example.

Story excerpt: *[...] He aimed at the first turkey, and pulled the trigger. After a moment's thought, he opened his shotgun and saw there were no bullets in the shotgun's chamber. He loaded his shotgun, aimed at the turkey and pulled the trigger again. Undisturbed, the bird nearby continued to chirp and build its nest. Papa Joe was very confused. Would this be the first time that his shotgun had let him down?*

KRR for Story Comprehension

The close link between human commonsense reasoning, such as that used for SC, and argumentation has been recently re-enforced by new psychological evidence (Mercier and Sperber 2011) suggesting that human reasoning is in its general form inherently argumentative. In our proposed approach of KRR for SC, the reasoning to construct a comprehension model and its qualification and revision at all levels as the story unfolds, will be captured through a uniform acceptability requirement on the arguments that support the conclusions in the model. We use methods and results from Argumentation Theory in AI, e.g., (Dung 1995; Modgil and Prakken 2012), and its links to the area of Reasoning about Action and Change (RAC) with Default Reasoning on the static properties of domains (see (van Harmelen, Lifschitz, and Porter 2008) for an overview).

In particular, we use a typical RAC language of Fluents, Actions, and Times, to represent the narrative $\mathcal{N}$ of a story, and also the world knowledge $\mathcal{W}$ used for the story's comprehension. The exact time-points are mostly inconsequential, and are meant to stand for the abstract scenes in a story.

The narrative of our story excerpt is represented as a sequence of timed-observations (pj is short for "Papa Joe"):

$\text{OBS}(alive(turkey), 1)$, $\text{OBS}(aim(pj, turkey), 1)$,
$\text{OBS}(pull\_trigger(pj), 1)$, $\text{OBS}(\neg gun\_loaded, 4)$,
$\text{OBS}(load\_gun, 5)$, $\text{OBS}(pull\_trigger(pj), 6)$,
$\text{OBS}(chirp(bird), 10)$, $\text{OBS}(nearby(bird), 10)$.

World knowledge is represented as a collection of unit-arguments, in the form of simple associations between concepts in the language. This stems from a key observation in Psychology that typically all world knowledge, and irrespective of type, is inherently default. It is not in the form of an elaborate formal theory of detailed definitions of concepts, but rather is better regarded as a collection of relatively loose semantic associations between concepts, reflecting typical rather than absolute information. Thus, knowledge need not be fully qualified at the representation level, since it can be qualified via the reasoning process by the relative strength of other (conflicting) pieces of knowledge. This addresses the ***endogenous qualification problem***, while the implicit priority of information in the narrative over any unit-argument also addresses the ***exogenous qualification problem***.

The world knowledge needed to make sense of our story excerpt is given below, with the priorities $p1 \succ c1$, $p2 \succ c1$:

$c1:\ \texttt{cau}(fired\_at(pj, X), \{aim(pj, X), pull\_trigger(pj)\})$
$c2:\ \texttt{cau}(\neg alive(X), \{fired\_at(pj, X), alive(X)\})$
$c3:\ \texttt{cau}(noise, \{fired\_at(pj, X)\})$
$c4:\ \texttt{cau}(\neg chirp(bird), \{noise, nearby(bird)\})$
$c5:\ \texttt{cau}(gun\_loaded, \{load\_gun\})$
$p1:\ \texttt{pro}(\neg fired\_at(pj, X), \{\neg gun\_loaded\})$
$p2:\ \texttt{pro}(\neg fired\_at(pj, X), \{\neg noise\})$ (story-specific)

Each unit-argument is of the general form $\texttt{arg}(H, B)$, with a literal (a fluent / action or its negation) H for its head, and a set of literals B for its body. Unit-arguments are able to capture information pertaining to how properties are caused to come about ($\texttt{cau}(H, B)$), how they relate to each other ($\texttt{pro}(H, B)$), and how they persist ($\texttt{per}(H, \{H\})$); unit-arguments of the latter type are implicitly assumed to be present for each literal H. Conflicts between unit-arguments are resolved based on the priority relation $\succ$ that holds between them. In general, the priority relation $\succ$ includes *(i)* $\texttt{cau}(H, B_1) \succ \texttt{per}(\neg H, B_2)$; *(ii)* $\texttt{per}(H, B_1) \succ \texttt{pro}(\neg H, B_2)$; and *(iii)* any given story-specific priorities.

The priority amongst these basic units of knowledge gives a form of non-monotonic reasoning (NMR) for deriving new properties that hold in the story. In particular, priorities in *(i)* address the ***frame problem***, ensuring that properties cease to persist when there is causal evidence to the contrary. When we need to reason with defeasible property information, such as default rules about the normal state of the world in which a story takes place, we are also faced with a ***generalized frame problem***, where "a state of the world persists irrespective of the existence of general state laws". Hence, if we are told that the world is in fact in some exceptional state that violates a general (default) property, this will continue to be the case in the future, until we learn of (or derive) some causal information that returns the world into its normal state. The solution to this generalized frame problem is

captured succinctly by inclusion of the priorities in *(ii)*.

Story-specific priorities account for the proper qualification of information in the context of a specific story. In our example story, for instance, we are told (outside the excerpt) that the gun had always made a loud noise when it fired. This is captured by including the story-specific priority $p2 \succ c1$.

In accordance to psychological evidence, the use of unit-arguments as loose associations allows for the *"fast thinking"* and easy elaboration with new inferences critical in narrative comprehension. On the other hand, easy elaboration and the extreme form of qualification that it needs are mitigated by the requirement that elaborative inferences need to be *grounded* on the narrative and *sceptical* in nature.

Argumentation Semantics

Given a narrative $\mathcal{N}$ and a world knowledge $\mathcal{W}$, we formulate an argumentation framework $\langle \mathcal{A}, \mathcal{T}, \mathcal{D} \rangle$ by defining the set of arguments $\mathcal{A}$, the attacking relation $\mathcal{T} \subseteq \mathcal{A} \times \mathcal{A}$, and the defending relation $\mathcal{D} \subseteq \mathcal{A} \times \mathcal{A}$, as in the sequel.

Arguments in $\mathcal{A}$ are collections of tuples, each comprising a unit-argument, the time-point at which it is applied, and the inference that follows from its application. The introduction of time-points is necessitated as each given unit-argument can be possibly applied multiple times, with each application qualified by other conflicting knowledge in different ways.

The inference that follows is of the form (X, T), where X is a fluent / action literal, and T the time-point at which it is inferred to hold. We allow a unit-argument $\texttt{arg}(H, B)$ to be used in the usual forward direction to draw (H, T) as inference (with T the time-point at which the unit-argument's head is applied), given that the premises B hold (at the appropriate time-point). In addition, we allow a unit-argument to be used in a backward direction (capturing, in this manner, reasoning by reductio ad absurdum) to draw $(\neg X, T)$ as inference (with T the time-point at which the unit-argument's body is applied), given that $X \in B$ and that the premises $B \setminus \{X\}$ and $\neg H$ hold (at the appropriate time-points).

Despite the use of defeasible world knowledge, the framework includes ***reasoning by contradiction***. The psychological debate on whether humans reason by contradiction, e.g., contraposition (Rips 1994; Johnson-Laird and Yang 2008) notwithstanding, it is still natural for a formal argumentation framework to capture such indirect reasoning; e.g., (Kakas and Mancarella 2013; Kakas, Toni, and Mancarella 2013). A main consequence is that it gives a form of ***backward persistence***, e.g., from an observation to support (but not necessarily conclude, due to a possible qualification) that the observed property holds also at previous time-points. Note also that the separation of the inference type (e.g., forward and backward) is known to be significant in preference-based argumentation (Modgil and Prakken 2012), and necessitates a careful treatment of the definition of attacks and defences.

To reflect the suggestion by Psychology that inferences drawn by readers are strongly tied to the story, we require that the activation conditions of argument tuples must be eventually traced back to the explicit information in the narrative of the story representation. Thus, each argument is restricted so that the premises of each tuple it contains are supported either directly by the narrative itself, or by inferences of other tuples in the argument in a non-cyclic manner.

The second key suggestion by Psychology that inferences are sceptical is reflected by allowing attacks to be generated "easily", and insisting that defences are "harder" to generate. Therefore, in case of "equally strong" arguments (leading to a non-deterministic split), there is an attack, but no defense.

Roughly, an argument A_1 attacks an argument A_2, and thus $(A_1, A_2) \in \mathcal{T}$, if they draw conflicting conclusions. The use of contrapositive reasoning for backward inference also means that it is possible to have arguments that support conclusions that are not contrary to each other, but whose unit-arguments have conflicting heads. For instance, in our running example we can use the causal unit-argument $c1$ in A_1 to forward derive $fired\_at(pj, X)$ and the property unit-argument $p1$ in A_2 to backward derive gun_loaded from $fired\_at(pj, X)$; despite that the derived conclusions of A_1 and A_2 are not in conflict, the unit-arguments used have conflicting heads. Although not all such *indirect* conflicts are important, a certain subset does need to be accounted for.

Assuming that $(A_1, A_2) \in \mathcal{T}$, an argument A_2 defends an argument A_1, and thus $(A_2, A_1) \in \mathcal{D}$, if the former undercuts the latter, but the latter does not undercut the former. Undercutting means that a unit-argument in A_2 has a higher priority than a unit-argument in A_1, capturing, in effect, a weakness in the chain of reasoning that A_1 used to attack A_2; cf. (Modgil and Prakken 2012). Again, care needs to be taken in dealing with indirect conflicts, as a result of the *defeasible nature* of the world knowledge and the allowance of reasoning by contradiction on such defeasible information.

Finally, an argument Δ is acceptable if it does not have a conflict with the narrative or itself, and for every argument A that attacks Δ, Δ defends A. An acceptable argument corresponds to a comprehension model of the story $\mathcal{N}$ under the world knowledge $\mathcal{W}$. A comprehension model can be tested, as is often done in Psychology, through a series of multiple-choice questions with answers of the form "X holds at T". Such an answer is accepted if (X, T) is entailed by the comprehension model, it is rejected if $(\neg X, T)$ is entailed by the comprehension model, and it is possible otherwise.

Evaluation through Empirical Studies

We have implemented the KRR framework as a Prolog system able to read a story narrative as a sequence of segments, compute a comprehension model, answer questions, and repeat the process, revising its conclusions as more parts of the story narrative become available. The process runs in time polynomial in the size of the world knowledge and the size of the story narrative, and is sound w.r.t. the framework.

In the first part of the evaluation of our approach we carried a *psychological study* to ascertain the world knowledge that is activated to successfully comprehend stories such as our example story on the basis of data obtained from human readers. We developed a set of inferential questions to follow the reading of pre-specified story segments. These assessed the extent to which readers connected, explained, and elaborated key story elements. Readers were instructed to answer each question and to justify their answers using a "think-aloud" method of answering questions while reading, in order to reveal the world knowledge that they had used.

The qualitative data from the readers was pooled together and analysed as to the frequencies of the types of responses in conjunction with the information given in justifications and think-aloud protocols. Considering those readers that demonstrated successful comprehension according to psychological criteria, our system was able to identify the most popular answers to questions, and also to recognize questions for which no single answer was (sceptically) accepted, following the variability demonstrated by human readers.

Related Work and Conclusions

Automated story understanding has been an ongoing endeavor of AI for more than forty years (Mueller 2002; 2013). Logic-related approaches have largely proceeded under the assumption that standard logical reasoning techniques can subsequently be applied; e.g., satisfiability (Mueller 2003) or planning (Niehaus and Young 2009). To our knowledge there has been very little work relating story comprehension with computational argumentation, an exception being (Bex and Verheij 2010), in which a case is made for combining narrative and argumentation techniques in the context of legal reasoning. Argumentation for reasoning about actions and change, on which our formal framework builds, has been studied in (Vo and Foo 2005; Michael and Kakas 2009).

To complete a fully automated approach to SC we continue drawing lessons from Psychology to address further the computational aspects of cognitive economy and coherence, by applying "computational heuristics" on top of the semantic framework we have developed. We are also investigating the systematic extraction or acquisition of commonsense world knowledge using lexical databases (Miller 1995; Baker, Fillmore, and Lowe 1998; Palmer, Gildea, and Kingsbury 2005), knowledge base archives (Lenat 1995), or automated extraction of commonsense knowledge from text (Michael and Valiant 2008; Michael 2009; 2011; 2013).

We envisage that the strong inter-disciplinary nature of our work can provide a concrete and important test-bed for evaluating the development of frameworks in AI while at the same time offering valuable feedback for Psychology.

References

Baker, C. F.; Fillmore, C. J.; and Lowe, J. B. 1998. The Berkeley FrameNet Project. In *ACL'98*.

Bex, F. J., and Verheij, B. 2010. Story Schemes for Argumentation about the Facts of a Crime. In *CMN'10*.

Brewer, W., and Lichtenstein, E. 1982. Stories are to Entertain: A Structural-Affect Theory of Stories. *Journal of Pragmatics* 6:473–486.

Dung, P. M. 1995. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *AIJ* 77(2):321–358.

Johnson-Laird, P. N., and Yang, Y. 2008. *Mental Logic, Mental Models, and Simulations of Human Deductive Reasoning*. The Cambridge Handbook of Computational Psychology. 339–358.

Kakas, A., and Mancarella, P. 2013. On the Semantics of Abstract Argumentation. *Logic Computation* 23:991–1015.

Kakas, A.; Toni, F.; and Mancarella, P. 2013. Argumentation for Propositional Logic and Nonmonotonic Reasoning. In *Commonsense'13*.

Kintsch, W. 1988. The Role of Knowledge in Discourse Comprehension: A Construction-Integration Model. *Psychological Review* 95:163–182.

Kintsch, W. 1998. *Comprehension: A Paradigm of Cognition*. NY: Cambridge University Press.

Lenat, D. B. 1995. CYC: A Large-Scale Investment in Knowledge Infrastructure. *CACM* 38(11):32–38.

McNamara, D. S., and Magliano, J. 2009. Toward a Comprehensive Model of Comprehension. *The Psychology of Learning and Motivation* 51:297–384.

Mercier, H., and Sperber, D. 2011. Why Do Humans Reason? Arguments for an Argumentative Theory. *Behavioral and Brain Sciences* 34(2):57–74.

Michael, L., and Kakas, A. C. 2009. Knowledge Qualification through Argumentation. In *LPNMR'09*.

Michael, L., and Valiant, L. G. 2008. A First Experimental Demonstration of Massive Knowledge Infusion. In *KR'08*.

Michael, L. 2009. Reading Between the Lines. In *IJCAI'09*.

Michael, L. 2011. Causal Learnability. In *IJCAI'11*.

Michael, L. 2013. Machines with Websense. In *Commonsense'13*.

Miller, G. A. 1995. WordNet: A Lexical Database for English. *CACM* 38(11):39–41.

Modgil, S., and Prakken, H. 2012. A General Account of Argumentation with Preferences. *AIJ* 195:361–397.

Mueller, E. T. 2002. Story Understanding. In Nadel, L., ed., *Encyclopedia of Cognitive Science*, volume 4, 238–246. London: Macmillan Reference.

Mueller, E. T. 2003. Story Understanding through Multi-Representation Model Construction. In *HLT-NAACL 2003 Workshop on Text Meaning*, 46–53.

Mueller, E. T. 2013. Story Understanding Resources. http://xenia.media.mit.edu/∼mueller/storyund/ storyres.html. Accessed February 28, 2013.

Niehaus, J., and Young, R. M. 2009. A Computational Model of Inferencing in Narrative. In *INT'09*.

Palmer, M.; Gildea, D.; and Kingsbury, P. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics* 31(1):71–106.

Rapp, D., and den Broek, P. V. 2005. Dynamic Text Comprehension: An Integrative View of Reading. *Current Directions in Psychological Science* 14:297–384.

Rips, L. 1994. *The Psychology of Proof*. MIT Press.

van Harmelen, F.; Lifschitz, V.; and Porter, B. 2008. *Handbook of Knowledge Representation*. Elsevier Science.

Vo, Q. B., and Foo, N. Y. 2005. Reasoning about Action: An Argumentation-Theoretic Approach. *JAIR* 24:465–518.

Zwaan, R. A. 1994. Effect of Genre Expectations on Text Comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 20:920–933.

Concept Dissimilarity with Triangle Inequality

Felix Distel
Institute of Theoretical Comp. Sci.
Faculty of Computer Science
TU Dresden, Germany
felix@tcs.inf.tu-dresden.de

Jamal Atif
Université Paris Sud, LRI
TAO, Orsay, France
jamal.atif@lri.fr

Isabelle Bloch
Institut Mines Télécom
Télécom ParisTech
CNRS LTCI, Paris, France
isabelle.bloch@telecom-paristech.fr

Abstract

Several researchers have developed properties that ensure compatibility of a concept similarity or dissimilarity measure with the formal semantics of Description Logics. While these authors have highlighted the relevance of the triangle inequality, none of their proposed dissimilarity measures satisfy it. In this work we present a theoretical framework for dissimilarity measures with this property. Our approach is based on concept relaxations, operators that perform stepwise generalizations on concepts. We prove that from any relaxation we can derive a dissimilarity measure that satisfies a number or properties that are important when comparing concepts.

1 Introduction

By nature description logics are well equipped for representing precise knowledge in a formal manner. As ontologies and description logics (DL) reach out to a broader audience some limitations become evident. In practice, it often occurs that two concepts have similar meanings, but no precise logical relationship can be established. Similarity measures, or dually dissimilarity measures, are attempts to quantify the differences between concepts. They are crucial in areas such as information retrieval in ontologies, ontology alignment, inductive logic programming and for some tasks in non-monotonic reasoning such as model-based revision or aggregation.

In a DL setting similarity can be defined between individuals, concepts, or even ontologies. In this work we focus exclusively on concept similarity. A large number of concept similarity measures has been developed, most of which are tailored to the specific needs of a particular field, such as biomedicine (Pesquita et al. 2009), or geospatial reasoning (Janowicz and Wilkes 2009). These approaches can be classified according to various criteria, such as the ones given in (Borgida, Walsh, and Hirsh 2005). Initially, the quality of similarity measures has only been measured in terms of empirical evaluations. Increasingly, researchers are starting to look at theoretical properties that ensure compatibility of a similarity measure with the formal semantics of description logics. Works such as (d'Amato, Staab, and Fanizzi 2008) and (Lehmann and Turhan 2012) list amongst others the properties of a metric, in particular the triangle inequality, as well as soundness with respect to equivalence and subsumption.

The triangle inequality has been somewhat controversial and in some applications such as (Janowicz and Wilkes 2009) it is not needed. In other applications such as metric-based conceptual clustering and distance-based optimization methods it is crucial (Fayyad et al. 1996). Unfortunately, even the measures presented in (Lehmann and Turhan 2012) and (d'Amato, Staab, and Fanizzi 2008) with their otherwise good theoretical properties do not satisfy the triangle inequality. Our results aim to provide knowledge engineers from these fields with an adequate measure.

In this work, we give a general framework that can be used to construct concept dissimilarity measures with good theoretical properties, including the triangle inequality. The framework is based on concept relaxations, operators that can be used to successively make concepts more general. A directed distance between two concepts C and D can then be defined as the number of times D needs to be relaxed before it subsumes C. We show that the maximum of the two directed distances yields a good dissimilarity measure. Finally, we demonstrate ways to instantiate the framework.

2 Preliminaries

2.1 Description Logics

Description logics are a family of knowledge representation formalisms (Baader 2003). Every description logic $\mathcal{L}$ provides a set of *concepts* $\mathsf{C}(\mathcal{L})$. Concept descriptions are recursively obtained from a set of *concept names* $\mathcal{N}_C$ and a set of *role names* $\mathcal{N}_R$ using concept constructors such as conjunction $\sqcap$, existential restrictions $\exists$ or the top concept $\top$, among others. The description logic that only allows for these three constructors is called $\mathcal{EL}$. In $\mathcal{EL}$, concepts can be visualized as *$\mathcal{EL}$-Description Trees* where node labels represent concept names and edges represent roles. For example the tree in Figure 1 represents the concept

$$\mathsf{Person} \sqcap \exists \mathsf{c}.\mathsf{Male} \sqcap \exists \mathsf{c}.\exists \mathsf{c}.\mathsf{Female}. \qquad (1)$$

Using a model based semantics one can define a generality relation on concepts. If D is more general than C, in other words D subsumes C, we write $C \sqsubseteq D$. We say that C and D are *equivalent* (denoted by $C \equiv D$) if both $C \sqsubseteq D$ and $D \sqsubseteq C$ hold.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

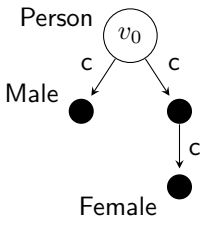


Figure 1: $\mathcal{EL}$-Description Tree for (1)

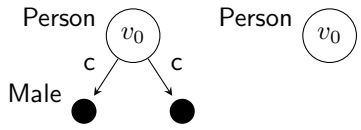


Figure 2: Consecutive applications of ρ_{depth} to (1)

In description logics, axioms are typically stored in ontologies, which can be divided into TBoxes and ABoxes. We define our framework in the absence of background ontologies.

2.2 Similarity and Dissimilarity on Concepts

When similarity measures were first investigated within the DL community, researchers mainly focused on adaptations of existing measures from other fields (cf. (Borgida, Walsh, and Hirsh 2005) for a survey). The quality of these measures was mainly examined in an empirical way, showing that they perform well in a given setting, but providing little transferable insight. It was only in (d'Amato, Staab, and Fanizzi 2008) that qualitative criteria were developed, based on the ones given by (Bock and Diday 2000). The following definition is slightly adapted to dissimilarity between concepts.

Definition 1 (Dissimilarity (Bock and Diday 2000)). *Let $\mathcal{L}$ be a DL language. A function $d\colon \mathsf{C}(\mathcal{L}) \times \mathsf{C}(\mathcal{L}) \to \mathbb{R}$ is called a* dissimilarity measure *if it satisfies the following properties for all $C, D \in \mathsf{C}(\mathcal{L})$.*

- positiveness: $d(C, D) \geq 0$
- reflexivity: $d(C, C) = 0$, *and*
- symmetry: $d(C, D) = d(D, C)$.

These properties can be expected to hold for any dissimilarity measure. In a description logics context it should also be compatible with the semantics of the logic. To ensure this, (d'Amato, Staab, and Fanizzi 2008) and more recently (Lehmann and Turhan 2012) have introduced an extended set of properties. These properties are originally stated for similarity measures, here we present their equivalents for dissimilarity measures.

Definition 2. *A dissimilarity measure $d\colon \mathsf{C}(\mathcal{L}) \times \mathsf{C}(\mathcal{L}) \to \mathbb{R}$ is called*

- equivalence closed *if $d(C, D) = 0 \implies C \equiv D$,*
- equivalence sound *if $D \equiv E \implies d(C, D) = d(C, E)$,*
- subsumption preserving *if $C \sqsubseteq D \sqsubseteq E \implies d(C, D) \leq d(C, E)$*
- reverse subsumption preserving *if $C \sqsubseteq D \sqsubseteq E \implies d(D, E) \leq d(C, E)$*
- structurally dependent *if for all sequences $(C_n)_n$ of atoms with $C_i \not\sqsubseteq C_j$ for all $i, j \in \mathbb{N}$, $i \neq j$ the concepts*

$$D_n = \prod_{i \leq n} C_i \sqcap D, E_n = \prod_{i \leq n} C_i \sqcap E$$

satisfy $\lim_{n\to\infty} d(D_n, E_n) = 0$ for all $C, D, E \in \mathsf{C}(\mathcal{L})$.

- *We say that d fulfills the* triangle inequality *if $d(C, E) \leq d(C, D) + d(D, E)$ for all $C, D, E \in \mathsf{C}(\mathcal{L})$.*

A dissimilarity d is a *metric* if it satisfies the triangle inequality and is additionally strict, i.e. $d(x, y) = 0$ implies $x = y$.

A desirable feature of a good dissimilarity measure is that concepts with more common features should be less dissimilar than concepts with few common features. Structural dependence is a formalization of this idea. Another attempt has been introduced in (d'Amato, Staab, and Fanizzi 2008), where it is formalized in terms of common subsumers.

Definition 3 ((Strict) Monotonicity). *A dissimilarity measure $d\colon \mathsf{C}(\mathcal{L}) \times \mathsf{C}(\mathcal{L}) \to \mathbb{R}$ is called* (strictly) monotone *if for all $C, D, E \in \mathsf{C}(\mathcal{L})$ that satisfy*

- *every common subsumer of C and E also subsumes D,*
- *there is a common subsumer of C and D that does not subsume E,*

it holds that $d(C, D) \leq d(C, E)$, respectively $d(C, D) < d(C, E)$.

3 General Framework

We provide a general framework for defining dissimilarity measures. All dissimilarity measures obtained within this framework have all properties from Section 2.2, except monotonicity and structural dependence. The framework is based on *concept relaxation operators*, operators that allow a stepwise generalization of concepts.

Definition 4 (Relaxation). *A* (concept) relaxation *is an operator $\rho\colon \mathsf{C}(\mathcal{L}) \to \mathsf{C}(\mathcal{L})$ that satisfies the following three properties for all $C, D \in \mathcal{L}$.*

1. *ρ is* non-decreasing*, i.e. $C \sqsubseteq D$ implies $\rho(C) \sqsubseteq \rho(D)$,*
2. *ρ is* extensive*, i.e. $C \sqsubseteq \rho(C)$, and*
3. *ρ is* exhaustive*, i.e. $\exists k \in \mathbb{N}_0\colon \top \sqsubseteq \rho^k(C)$, where ρ^k denotes ρ applied k times, and ρ^0 is the identity.*

Examples for relaxation operators that can be used to instantiate the framework are presented in Section 4.

A dissimilarity measure that is equivalence sound and closed should have the value $d(C, D) = 0$ if and only if $C \equiv D$, i.e. iff $C \sqsubseteq D$ and $D \sqsubseteq C$. Like (Lehmann and Turhan 2012) and (Suntisrivaraporn 2013) we first introduce directed measures d^d_ρ that capture how "far" D is from being a subsumer of C, and vice versa. If both $C \sqsubseteq D$ and $D \sqsubseteq C$ hold, then both directed measures will be 0. The directed measure $d^d_\rho(C, D)$ counts how often we need to successively relax D to reach a subsumer of C. If we think of concepts in terms of sets of individuals, then the intuition behind successive relaxations can be visualized as in Figure 3.

Definition 5 (Directed measure). *Let ρ be a relaxation on $\mathsf{C}(\mathcal{L})$. For $C, D \in \mathsf{C}(\mathcal{L})$ the* directed measure *$d^d_\rho(C, D)$ is defined as*

$$d^d_\rho(C, D) = \min\{k \in \mathbb{N}_0 \mid C \sqsubseteq \rho^k(D)\},$$

where ρ^k denotes ρ applied k times, and ρ^0 is the identity.

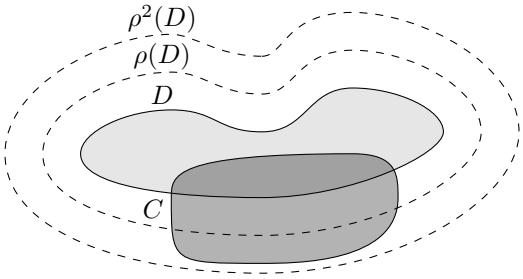


Figure 3: D needs to be relaxed twice before it subsumes C, i.e. $d^d_\rho(C, D) = 2$

The directed measure is always finite because ρ is exhaustive. We can then define the *relaxation dissimilarity* based on a relaxation operator simply as the maximum of the two directed measures.

Definition 6 (Relaxation Dissimilarity). *Let $\rho\colon \mathcal{L} \to \mathcal{L}$ be a relaxation on $\mathsf{C}(\mathcal{L})$. For two concepts C and D the* relaxation dissimilarity $d_\rho(C, D)$ *is defined as*

$$d_\rho(C, D) = \max\{d^d_\rho(C, D), d^d_\rho(D, C)\}.$$

Theorem 1. *For every relaxation ρ the operator d_ρ is a dissimilarity measure, that is equivalence sound, equivalence closed, subsumption preserving and reverse subsumption preserving, and satisfies the triangle inequality.*

Proof. Positiveness, reflexivity and symmetry follow trivially from Definitions 5 and 6, and therefore d_ρ is a dissimilarity measure.

We have the following chain of equivalences: $C \equiv D$, iff $C \sqsubseteq D$ and $D \sqsubseteq C$, iff $C \sqsubseteq \rho^0(D)$ and $D \sqsubseteq \rho^0(C)$, iff $d^d_\rho(C, D) = d^d_\rho(D, C) = 0$, iff $d_\rho(C, D) = 0$. Thus d_ρ is both equivalence sound and equivalence closed.

To prove the triangle inequality, let C, D, E be concept descriptions and let $d_\rho(C, D) = d_1$, $d_\rho(D, E) = d_2$. Then in particular, $d^d_\rho(C, D) \leq d_1$ and thus $C \sqsubseteq \rho^{d_1}(D)$ by extensivity. Similarly, we obtain $D \sqsubseteq \rho^{d_2}(E)$. Since relaxations are non-decreasing we obtain from the latter

$$\rho^{d_1}(D) \sqsubseteq \rho^{d_1+d_2}(E)$$

and therefore $C \sqsubseteq \rho^{d_1+d_2}(E)$, i.e. $d^d_\rho(C, E) \leq d_1 + d_2$. Analogously, it can be shown that $d^d_\rho(E, C) \leq d_1 + d_2$ and thus $d_\rho(C, E) \leq d_1 + d_2 = d_\rho(C, D) + d_\rho(D, E)$.

To show subsumption preservation let $C \sqsubseteq D \sqsubseteq E$ with $d_\rho(C, E) = d$. Then in particular, $E \sqsubseteq \rho^d(C)$ and thus also $D \sqsubseteq \rho^d(C)$. On the other hand, $C \sqsubseteq \rho^0(D) \sqsubseteq \rho^d(D)$ by extensivity, which yields $d_\rho(C, D) \leq k = d_\rho(C, E)$, which proves subsumption preservation. □

A comparison of these properties with those of some existing measures is provided in Table 1.

4 Instantiations

Theorem 1 shows that our framework produces dissimilarity measures with good theoretical properties. The framework can be instantiated with any relaxation operator. The behaviour of the resulting measures can vary greatly depending on the relaxation. This is demonstrated by the following examples. A trivial relaxation in any description logic is the operator $\rho_\top$ that maps every concept to $\top$. It results in a very coarse dissimilarity measure $\rho_\top$ that is 0 iff the concepts are equivalent and 1 otherwise.

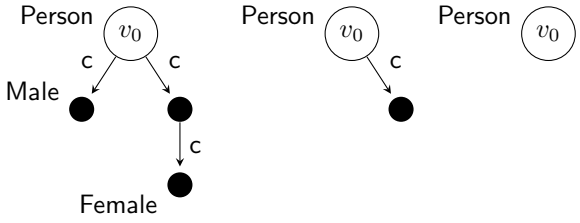


Figure 4: Consecutive application of ρ_{leaves} to (1)

Relaxations from tree operations For the lightweight logic $\mathcal{EL}$ (Baader, Küsters, and Molitor 1999) have proven a close connection between $\mathcal{EL}$ concepts and description trees. Due to this connection any operator that maps description trees to strict subtrees gives rise to a relaxation. One possibility is the operator ρ_{depth} that reduces the role depth of each concept by 1, simply by pruning the description tree (cf. Figure 2). The corresponding dissimilarity ρ_{depth} measures the first depth-level where the concepts differ in the description tree. It gives higher weight to features at a smaller depth. For example, if we compare the concepts $\mathsf{F} := \mathsf{Male} \sqcap \exists\mathsf{hasChild}.\top$ and $\mathsf{HoJ} := \mathsf{Male} \sqcap \exists\mathsf{marriedTo}.(\mathsf{Female} \sqcap \mathsf{Judge})$ to the concept $\exists\mathsf{hasChild}.\top$ the value will be 2 in both cases, since the change occurs at the lowest level, in the concept name Male. This is counterintuitive, since F and $\exists\mathsf{hasChild}.\top$ share more common features than HoJ and $\exists\mathsf{hasChild}.\top$. A slightly better behaviour can be achieved by the relaxation ρ_{leaves} that removes all leaves from a description tree (Figure 4).

Relaxations from distances between models As logics become more expressive, it becomes harder to directly define a relaxation on the concepts. Since the models remain simple labeled graphs, even for complex descriptions, one solution is to identify concepts with the set of their models. Similar to related work from Section 5 one might start with a simple distance between models, e.g. an edit distance, and generalize it to a Hausdorff distance between sets of models. However, since the model space is infinite the Hausdorff distance can often not be computed directly.

A workaround is to use the distance on the model space to define dilations, as used in mathematical morphology (Serra 1982), on sets of models. For some distances, such as a simple tree edit distance, the dilated sets themselves correspond to DL concepts. The operator that maps a concept to the concept corresponding to its dilated set of models can be shown to be a relaxation. All these instantiations will be further studied in our future work.

Table 1: Properties of some (dis-)similarity measures

Measure	Equivalence Sound	Monotone	Equivalence Closed	Subs. Preserving	Rev. Subs. Preserving	Structurally Dependent	Triangle Inequality
(Lehmann and Turhan 2012)	✓	–	✓	✓	✓	✓	–
(d'Amato, Staab, and Fanizzi 2008)	✓	✓	–	✓	✓	–	–
relaxation dissimilarity	✓	–	✓	✓	✓	–	✓

5 Existing Metrics for Other Logics

Outside of description logics several works have proposed metrics between logical objects. Works such as (Nienhuys-Cheng 1998; Ramon and Bruynooghe 1998) exploit the fact that is relatively easy to define a metric on ground expressions in first order logic. They extend these ground distances to sets of atoms, or Herbrand interpretations using constructions such as Hausdorff distances or Manhattan distances.

In some cases it is straightforward to define a distance between two terms if one is a generalization of the other. To obtain a distance between two arbitrary terms one can simply use the sum of the distances to their least general common generalization. In a general form (Birkhoff 1993) has presented this idea as the classical distance in graded lattices. It is used to define a distance between first order literals by (Hutchinson 1997), who then generalizes it to a distance between clauses using the Hausdorff metric. This idea can also be extended to cases where there is no unique minimally general generalization (De Raedt and Ramon 2009).

6 Discussion

In this work, we have presented a framework for dissimilarity measures whith good theoretical properties (cf. Table 1). Our measures satisfy at the same time the properties of a metric, in particular the triangle inequality, and they are compatible with the semantics of description logics, in particular they are equivalence sound. Some hints for instantiations of the proposed framework have been provided, and will be the focus of future work.

The similarity measures that we have presented here are defined for concepts without TBoxes. If the background ontology is an acyclic TBox, they can trivially be adapted by comparing only unfolded concepts. In principle, it is possible to generalize relaxations with respect to general TBoxes, but it is left for future work how to instantiate them.

Acknowledgments. This work was initiated during the stay of Felix Distel at Telecom ParisTech in summer 2012, supported by a grant from this institution. Felix Distel has also been supported by the Collaborative Research Center 912 'Highly Adaptive Energy-Efficient Computing'. The authors would like to thank Anni-Yasmin Turhan for fruitful discussions.

References

Baader, F.; Küsters, R.; and Molitor, R. 1999. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 96–101. Morgan-Kaufmann.

Baader, F. 2003. Description Logic terminology. In Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press. 485–495.

Birkhoff, G. 1993. *Lattice theory*, volume 25 of *Colloquium publications*. Providence, Rhode Island: American Mathematical Society, 3rd edition.

Bock, H. H., and Diday, E. 2000. *Analysis of symbolic data: exploratory methods for extracting statistical information from complex data*. Berlin: Springer.

Borgida, A.; Walsh, T. J.; and Hirsh, H. 2005. Towards measuring similarity in description logics. In *Proc. of the 2005 Int. Workshop on Description Logics (DL)*.

d'Amato, C.; Staab, S.; and Fanizzi, N. 2008. On the influence of description logics ontologies on conceptual similarity. In *Knowledge Engineering: Practice and Patterns*. Springer. 48–63.

De Raedt, L., and Ramon, J. 2009. Deriving distance metrics from generality relations. *Pattern Recognition Letters* 30(3):187–191.

Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R. 1996. Advances in knowledge discovery and data mining.

Hutchinson, A. 1997. Metrics on terms and clauses. In *Proc. of the European Conf. on Machine Learning (ECML)*, 138–145. Springer.

Janowicz, K., and Wilkes, M. 2009. SIM-DLA: A novel semantic similarity measure for description logics reducing inter-concept to inter-instance similarity. In *The Semantic Web: Research and Applications*. Springer. 353–367.

Lehmann, K., and Turhan, A.-Y. 2012. A framework for semantic-based similarity measures for $\mathcal{ELH}$-concepts. In *Proc. of the 13th European Conf. on Logics in Artificial Intelligence (ECAI)*, LNAI, 307–319. Springer Verlag.

Nienhuys-Cheng, S.-H. 1998. Distances and limits on herbrand interpretations. In *Inductive Logic Programming*. Springer. 250–260.

Pesquita, C.; Faria, D.; Falcao, A. O.; Lord, P.; and Couto, F. M. 2009. Semantic similarity in biomedical ontologies. *PLoS computational biology* 5(7):e1000443.

Ramon, J., and Bruynooghe, M. 1998. A framework for defining distances between first-order logic objects. In *Inductive Logic Programming*. Springer. 271–280.

Serra, J. 1982. *Image analysis and mathematical morphology*. London.: Academic Press.

Suntisrivaraporn, B. 2013. A similarity measure for the description logic $\mathcal{EL}$ with unfoldable terminologies. In *Proc. of the 5th Int. Conf. on Intelligent Networking and Collaborative Systems (INCoS)*, 408–413. IEEE.

On Redundant Topological Constraints*
(Extended Abstract)

Matt Duckham
Department of Infrastructure Engineering
University of Melbourne

Sanjiang Li, Weiming Liu, Zhiguo Long
Centre for Quantum Computation & Intelligent Systems
Faculty of Engineering and Information Technology
University of Technology Sydney

Abstract

The Region Connection Calculus (RCC) is a well-known calculus for representing part-whole and topological relations. It plays an important role in qualitative spatial reasoning, geographical information science, and ontology. The computational complexity of reasoning with RCC has been investigated in depth in the literature. Most of these works focus on the consistency of RCC constraint networks. In this paper, we consider the important problem of redundant RCC constraints. For a set Γ of RCC constraints, we say a constraint (xRy) in Γ is *redundant* if it can be entailed by the rest of Γ. A *prime subnetwork* of Γ is a subset of Γ which contains no redundant constraints but has the same solution set as Γ. It is natural to ask how to compute a prime subnetwork, and when it is unique. In this paper, we show that this problem is in general intractable, but becomes tractable if Γ is over a tractable subclass of RCC. If $\mathcal{S}$ is a tractable subclass in which weak composition distributes over non-empty intersections, then we can show that Γ has a unique prime network, which is obtained by removing all redundant constraints from Γ. As a byproduct, we identify a sufficient condition for a path-consistent network being minimal.

1 Introduction

Qualitative spatial reasoning (QSR) is a common subfield of artificial intelligence and geographical information science, and has applications in GIS, cognitive robotics, high-level understanding of video data etc. The Region Connection Calculus (RCC) (Randell, Cui, and Cohn 1992) is perhaps the most well-known calculus for representing qualitative spatial information. Based on a binary connectedness relation, it defines a class of binary topological relations between regions in a connected topological space (e.g., the real plane). The RCC is an expressive formalism for representing topological information, and the computational complexity of reasoning with RCC has been investigated in depth in the literature. Most of these works focus on the consistency or satisfiablility of RCC constraint networks.

In this paper, we consider the important problem of redundant RCC constraints. Given a set Γ of RCC constraints, we say a constraint (xRy) in Γ is *redundant* if it can be entailed by the rest of Γ, i.e., removing (xRy) from Γ will not change the solution set of Γ. It is natural to ask when a network is redundant and how to get an irredundant subset without changing the solution set. We call a subset of Γ a *prime subnetwork* of Γ if it contains no redundant constraints but has the same solution set as Γ.

We show that it is in general co-NP hard to determine if a constraint is redundant in a network of RCC constraints, but Γ is over a tractable subclass, then a prime subnetwork can be found in $O(n^5)$ time. If in addition weak composition distributes over non-empty intersections of relations in $\mathcal{S}$, then Γ has a unique prime subnetwork, which is obtained by removing all redundant constraints from Γ.

As in the case of propositional logic formulas (Liberatore 2005), redundancy of RCC constraints "often leads to unnecessary computation, wasted storage, and may obscure the structure of the problem" (Belov et al. 2012). Finding a prime subnetwork can be useful in at least the following aspects: a) computing and storing the relationships between spatial objects and hence saving space for storage and communication; b) facilitating comparison between different constraint networks; c) handling the inconsistency by modifying critical constraints; d) unveiling the essential graphical structure of a network; and e) adjusting geometrical objects to meet topological constraints (Wallgrün 2012).

Motivational Example: Placename Footprints

To motivate our discussion, we focus briefly on one specific application to illustrate just one of our five uses of prime subnetworks: saving space for storage. Figure 1 gives a small example of a set of spatial regions formed by the geographic "footprints" associated with placenames in the Southampton area of the UK. The footprints are derived from crowd-sourced data, formed from the convex hull of the sets of coordinate locations at which individuals used the placenames on social media. Using such data sets in natural language placename searches frequently requires queries over the topological relationships between footprints (e.g., "is Clarence Pier in Southampton?"). Computing such relationships on-the-fly requires computationally intensive and slow geometric operations; by contrast Web-search queries demand rapid responses.

One potential solution is to cache the topological rela-

*Future improvements and extended versions of this paper will be published in CORR at http://arxiv.org/abs/1403.0613

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

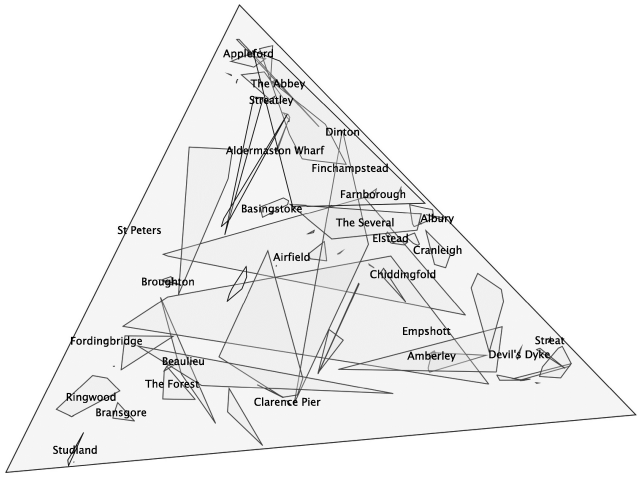


Figure 1: Examples of crowd-sourced geographic placename "footprints" around Southampton, UK

tions between all footprints of interest. However, even the small example in Figure 1, the 84 footprints then require $84 * 83/2 = 3486$ stored relations. The moderate-sized footprint data set from which Figure 1 contains a total of 3474 footprints leads to a constraint network with $6,032,601$ relations. It is easy to see that as crowd-sourced data sources continue to grow, the volumes of such data is set to explode. In the case of footprints, many of the relationships can be inferred, and computing the prime subnetwork can reduce the number of stored relationships to be approximately linear in the number of footprints. In the case of the Southampton constraint network, 1324 redundant relations lead to a prime subnetwork with only 2150 relations needing to be stored. For the full data set, $5,713,563$ redundant relations lead to a prime subnetwork of just $319,038$ relations (in contrast to the full constraint network of more than 6 million relations).

In Section 2 we recall the RCC constraint language and then define the key notions of redundant constraint and prime subnetwork in Section 3. We present our major results in Section 4 and conclude the paper in Section 5.

2 RCC Constraint Language

The RCC was introduced in (Randell, Cui, and Cohn 1992). Let U be the set of nonempty regular closed sets of $\mathbb{R}^2$. We call each element in U a region. For two regions a, b, we say a is a *part* of b, written $a\mathbf{P}b$, if $a \subseteq b$; say a is *connected* to b, written $a\,\mathbf{C}\,b$, if $a \cap b \neq \varnothing$. Using $\mathbf{C}$ and $\mathbf{P}$, we define

$$
\begin{aligned}
x\,\mathbf{PP}\,y &\equiv x\,\mathbf{P}\,y \wedge \neg(y\,\mathbf{P}\,x)\\
x\mathbf{O}y &\equiv (\exists z)(z\,\mathbf{P}\,x \wedge z\,\mathbf{P}\,y)\\
x\,\mathbf{DR}\,y &\equiv \neg(x\mathbf{O}y)\\
x\,\mathbf{PO}\,y &\equiv x\mathbf{O}y \wedge \neg(x\,\mathbf{P}\,y) \wedge \neg(y\,\mathbf{P}\,x)\\
x\,\mathbf{EQ}\,y &\equiv x\,\mathbf{P}\,y \wedge y\,\mathbf{P}\,x\\
x\,\mathbf{DC}\,y &\equiv \neg(x\mathbf{C}y)\\
x\,\mathbf{EC}\,y &\equiv x\,\mathbf{C}\,y \wedge \neg(x\mathbf{O}y)\\
x\,\mathbf{TPP}\,y &\equiv x\,\mathbf{PP}\,y \wedge (\exists z)(z\,\mathbf{EC}\,x \wedge z\,\mathbf{EC}\,y)\\
x\,\mathbf{NTPP}\,y &\equiv x\,\mathbf{PP}\,y \wedge \neg(x\,\mathbf{TPP}\,y)
\end{aligned}
$$

Write $\mathbf{PP}^{-1}$, $\mathbf{TPP}^{-1}$ and $\mathbf{NTPP}^{-1}$ for the converses of $\mathbf{PP}$, $\mathbf{TPP}$ and $\mathbf{NTPP}$, respectively. Then

(1) $\mathcal{B}_5 = \{\mathbf{DR}, \mathbf{PO}, \mathbf{EQ}, \mathbf{PP}, \mathbf{PP}^{-1}\}$

(2) $\mathcal{B}_8 = \{\mathbf{DC}, \mathbf{EC}, \mathbf{PO}, \mathbf{EQ}, \mathbf{TPP}, \mathbf{NTPP}, \mathbf{TPP}^{-1}, \mathbf{NTPP}^{-1}\}$

are two jointly exhaustive and pairwise disjoint (JEPD) sets of relations, i.e., for any two regions $a, b \in U$, a, b is related by exactly one relation in $\mathcal{B}_l$ ($l = 5, 8$). We call the Boolean algebra generated by relations in $\mathcal{B}_l$ the RCCl algebra, which consists all relations that are unions of the basic relations in $\mathcal{B}_l$. For convenience, we denote a non-basic RCCl relation R as the subset of $\mathcal{B}_l$ it contains. For example, we write $\{\mathbf{DR}, \mathbf{PO}, \mathbf{PP}\}$ for the relation $\mathbf{DR} \cup \mathbf{PO} \cup \mathbf{PP}$, and write $\star_5$ and $\star_8$ for the universal relation.

The composition of two basic RCC5/8 relations is not necessarily a relation in RCC5/8. For two RCC5/8 relations R and S, we call the smallest relation in RCC5/8 that contains $R \circ S$ the *weak composition* of R and S, written $R \circ_w S$ (Düntsch, Wang, and McCloskey 2001; Li and Ying 2003).

RCC5/8 Constraint Network

An RCC5/8 constraint has the form (xRy), where x, y are variables taking values from U, the set of regions, R is an RCC5/8 relation (not necessarily basic). Given a set Γ of RCC5/8 constraints over variables $V = \{v_1, v_2, ..., v_n\}$, we say Γ is *consistent* or *satisfiable* if there is an assignment $\sigma : V \to U$ such that $(\sigma(v_i), \sigma(v_j))$ satisfies the constraint in Γ that relates v_i to v_j.

Without loss of generality, we assume Γ has the form $\{x_i R_{ij} x_j\}_{i,j=1}^n$, where, for any $1 \leq i, j \leq n$, there is a unique constraint R_{ij}, and $R_{ji} = R_{ij}^{\sim}$ and $R_{ii} = \mathbf{EQ}$. In this sense, we call Γ a *constraint network*. Let $\Gamma = \{x_i R_{ij} x_j\}_{i,j=1}^n$ and $\Gamma' = \{x_i R'_{ij} x_j\}_{i,j=1}^n$ be two RCC5/8 constraint networks. We say Γ and Γ' are *equivalent* if they have the same set of solutions; and say Γ *refines* Γ' if $R_{ij} \subseteq R'_{ij}$ for all (i, j). We say an RCC5/8 network Γ is a *basic* network if each constraint is either the universe relation or a basic relation; and say a basic network *complete* if there are no universal relations.

Suppose $\mathcal{S}$ is a subset of RCC5/8. We say an RCC5/8 network $\Gamma = \{v_i R_{ij} v_j\}$ is over $\mathcal{S}$ if $R_{ij} \in \mathcal{S}$ for every pair of variables v_i, v_j. The consistency problem over $\mathcal{S}$, written as $\mathsf{CSP}(\mathcal{S})$, is the decision problem of the consistency of an arbitrary constraint network over $\mathcal{S}$. It is well known that the consistency problem over RCC5/8, i.e., $\mathsf{CSP}(RCC5/8)$, is NP-complete and RCC8 has three maximal tractable subclasses that contain all basic relations (Renz 1999) and RCC5 has only one (Jonsson and Drakengren 1997).

We say a network $\Gamma = \{v_i R_{ij} v_j\}$ *path-consistent* if for every $1 \leq i, j, k \leq n$, we have

$$R_{ij} \subseteq R_{ik} \circ_w R_{kj}. \tag{3}$$

A cubic algorithm, henceforth called the *path-consistency algorithm* or PCA, has been devised to enforce path-consistency. For any RCC5/8 network Γ, the PCA either detects inconsistency of Γ or returns a path-consistent network,

written Γ_p, which is equivalent to Γ and known as the *algebraic closure* or *a-closure* of Γ (Ligozat and Renz 2004). It is easy to see that in this case Γ_p also refines Γ, i.e., we have $S_{ij} \subseteq R_{ij}$ for each constraint $(x_i S_{ij} x_j)$ in Γ_p.

Proposition 1. *Let $\mathcal{S}$ be a tractable subclass of RCC5/8 which contains all basic relations. An RCC5/8 network Γ over $\mathcal{S}$ is consistent if applying PCA to Γ does not result inconsistency.*

In particular, we have

Proposition 2. *A basic RCC5/8 network Γ is consistent if it is path-consistent.*

Distributive Subalgebra

Write $\widehat{\mathcal{B}}_5$ for the closure of $\mathcal{B}_5$ under converse, intersection, and weak composition in RCC5. Then $\widehat{\mathcal{B}}_5$ contains the basic relations as well as

$$\{\mathbf{PO}, \mathbf{PP}\}, \{\mathbf{PO}, \mathbf{PP}^{-1}\}, \{\mathbf{PO}, \mathbf{PP}, \mathbf{PP}^{-1}, \mathbf{EQ}\},$$
$$\{\mathbf{DR}, \mathbf{PO}, \mathbf{PP}\}, \{\mathbf{DR}, \mathbf{PO}, \mathbf{PP}^{-1}\}, \{\mathbf{DR}, \mathbf{PO}\}, \star_5,$$

where $\star_5 = \{\mathbf{DR}, \mathbf{PO}, \mathbf{PP}, \mathbf{PP}^{-1}, \mathbf{EQ}\}$. It is interesting to note that in $\widehat{\mathcal{B}}_5$ the weak composition operation is *distributive* over nonempty intersections in the following sense.

Lemma 3. *Let R, S, T be three relations in $\widehat{\mathcal{B}}_5$. Suppose $S \cap T$ is nonempty. Then we have*

$$R \circ_w (S \cap T) = R \circ_w S \cap R \circ_w T \tag{4}$$
$$(S \cap T) \circ_w R = S \circ_w R \cap T \circ_w R. \tag{5}$$

In what follows, we call such a subclass a distributive subalgebra. Formally, we have

Definition 4. Let $\mathcal{S}$ be a subclass of RCC5/8. We say $\mathcal{S}$ is a *distributive subalgebra* if

- $\mathcal{S}$ contains all basic relations, and is closed under converse, weak composition, and intersection;
- weak composition distributes over nonempty intersections of relations in $\mathcal{S}$.

3 Redundant Constraint and Prime Subnetwork

We give a formal definition of redundant constraints.

Definition 5. Suppose Γ is an RCC5/8 network over variables $V = \{v_1, ..., v_n\}$. We say Γ *entails* a constraint $(v_i R v_j)$, written $\Gamma \models (v_i R v_j)$, if for every solution $\{a_1, ..., a_n\}$ of Γ we have $(a_i, a_j) \in R$. A constraint $(v_i R v_j)$ in Γ is *redundant* if $\Gamma \setminus \{(v_i R v_j)\}$ entails $(v_i R v_j)$. We say Γ is *reducible* if it has a redundant constraint, and *irreducible* otherwise. We say a subset Γ' of Γ is a *prime* subnetwork of Γ if Γ' is irreducible and equivalent to Γ.

Each universal constraint $(v_i \star v_j)$ in Γ is, by definition, a redundant constraint in Γ.

Given an RCC5/8 network Γ, a very interesting question is, *how to find a prime subnetwork of* Γ? This problem is clearly at least as hard as determining if Γ is reducible. Similar to the case of propositional logic formulars (Liberatore 2005), we have the following result for RCC5/8.

Proposition 6. *Let Γ be an RCC5/8 network and suppose (xRy) is a constraint in Γ. It is co-NP-complete to decide if (xRy) is redundant in Γ.*

A naive method to obtain a prime subnetwork is to remove redundant constraints one by one from Γ until we get an irreducible network. Suppose we have an oracle which can tell if a constraint is redundant. Then in an additional $O(n^2)$ time we can find an irreducible network that is equivalent to Γ by removing several constraints from Γ.

Despite that it is in general intractable to determine if a constraint is redundant, we have a polynomial time procedure if the constraints are all taken from a tractable subclass.

Proposition 7. *Let $\mathcal{S}$ be a tractable subclass of RCC5/8 that contains all basic relations. Suppose Γ is a network over $\mathcal{S}$. Then in $O(n^3)$ time we can determine whether a constraint is redundant in Γ and in $O(n^5)$ time find all redundant constraints of Γ. In addition, a prime subnetwork for Γ can be found in $O(n^5)$ time.*

Definition 8. The *core* of an RCC5/8 network Γ, written Γ_c, is defined to be the set of non-redundant constraints in Γ.

It is easy to see that the core of Γ is contained in every prime subnetwork of Γ. *Are prime subnetworks unique*? In general this is not the case.

In the following we assume without loss of generality that Γ has the following property:

$$(\forall i, j)[(i \neq j) \rightarrow (\Gamma \not\models (v_i \, \mathbf{EQ} \, v_j))]. \tag{6}$$

In the next section we show that, if Γ is a constraint network over a distributive subalgebra of RCC5/8, then Γ_c is the unique prime network of Γ. This is quite surprising, as, in general, knowing that (xRy) and (uSv) are both redundant in Γ does not imply that (uSv) is redundant in $\Gamma \setminus \{(xRy)\}$.

4 Networks over a Distributive Subalgebra

In this section, we assume $\mathcal{S}$ is a distributive subalgebra of RCC5/8. Let Γ be a consistent network over $\mathcal{S}$ which satisfies (6). We show that Γ_c is equivalent to Γ and hence the unique prime network of Γ.

Definition 9 (cf. (Chandra and Pujari 2005; Liu and Li 2012)). Suppose $\Theta = \{v_i T_{ij} v_j\}_{1 \leq i,j \leq n}$ is an RCC5/8 network. We say Θ is *minimal* if for every pair of variables v_i, v_j $(i \neq j)$ and every basic relation α in T_{ij}, there exists a solution $\{a_1, a_2, \cdots, a_n\}$ of Θ such that (a_i, a_j) is an instance of α.

Each consistent RCC5/8 constraint network has a unique minimal network, but it is in general NP-hard to compute it.

Notation: We write Γ_m for the minimal network of Γ, Γ_p for the a-closure of Γ, and Γ_c for the core of Γ.

To prove that Γ_c is equivalent to Γ, we need two important results. The first result, stated in Theorem 10, shows that Γ_m is exactly Γ_p. The second result, stated in Proposition 11, shows that a particular constraint (xRy) is redundant in Γ iff its corresponding constraint in Γ_p is redundant. Our main result, stated in Theorem 12, then follows immediately.

Theorem 10. *Let $\mathcal{S}$ be a distributive subalgebra of RCC5/8. Suppose Γ is a consistent network over $\mathcal{S}$ and Γ_p its a-closure. Then Γ_p is Γ_m, the minimal network of Γ.*

Proposition 11. *Let $\mathcal{S}$ be a distributive subalgebra of RCC5/8. Suppose Γ is a consistent network over $\mathcal{S}$ which satisfies* (6)*. Assume that (xRy) and (xSy) are the constraints from x to y in Γ and Γ_p respectively. Then (xRy) is redundant in Γ iff (xSy) is redundant in Γ_p.*

Recall that Theorem 10 asserts that Γ_p is minimal. As a consequence, we have our main result.

Theorem 12. *Let $\mathcal{S}$ be a distributive subalgebra of RCC5/8. Suppose Γ is a consistent network over $\mathcal{S}$ which satisfies* (6) *and Γ_c the core of Γ. Then Γ_c is equivalent to Γ and hence the unique prime network of Γ.*

Recall that Proposition 7 shows that the core of an RCC5/8 network over a tractable subclass can be found in $O(n^5)$ time. In the next subsection we show this can be improved if the network is over a distributive subalgebra.

A Cubic Algorithm for Computing the Core of Γ

We first consider the special case when Γ is path-consistent.

Lemma 13. *Let $\mathcal{S}$ be a distributive subalgebra of RCC5/8. Suppose Γ is a path-consistent network over $\mathcal{S}$. Then a constraint $(v_iR_{ij}v_j)$ is redundant in Γ iff $R_{ij} = \bigcap\{R_{ik} \circ_w R_{kj} : k \neq i, j\}$, i.e., R_{ij} is the intersection of the weak compositions of all paths from v_i to v_j which have length 2.*

Suppose Γ is a consistent network over a distributive subalgebra of RCC5/8 and satisfies (6). Proposition 11 and Lemma 13 suggest a simple way for computing Γ_c, the unique prime network of Γ. By Proposition 11, a constraint $(v_iR_{ij}v_j)$ in Γ is redundant iff the corresponding constraint $(v_iS_{ij}v_j)$ in Γ_p is redundant. Furthermore, Lemma 13 shows that $(v_iS_{ij}v_j)$ is redundant in Γ_p iff S_{ij} is the intersection of all $S_{ik} \circ_w S_{kj}$ ($k \neq i, j$). We hereby have the following cubic algorithm for finding all redundant constraints in Γ. For each constraint $(v_iS_{ij}v_j)$, to verify if $S_{ij} = \bigcap\{S_{ik} \circ_w S_{kj} : k \neq i, j\}$, we introduce a relation P_{ij} which consists of all basic relations α that are not in $S_{ik} \circ_w S_{kj}$ for some $k \neq i, j$ and then check if $P_{ij} \cup S_{ij}$ is the universal relation.

Algorithm 1: Algorithm for finding redundant constraints in constraint networks over a distributive subalgebra $\mathcal{S}$ of RCC5/8, where $\star_l$ is the universal relation in RCCl.

Input: a consistent constraint network $\Gamma = \{v_iR_{ij}v_j : 1 \leq i, j \leq n\}$ over $\mathcal{S}$.
Output: $Redun$: the set of redundant constraints of Γ.

```
Redun ← ∅;
Γ_p = the a-closure of Γ;
for each constraint (v_i S_ij v_j) ∈ Γ_p do
    P_ij ← ∅;
    for each variable v_k ∈ V \ {v_i, v_j} do
        temp_ij = S_ik ∘_w S_kj;
        for each basic RCC5/8 relation α do
            if α ∉ temp_ij then
                P_ij ← P_ij ∪ {α};
            end
        end
    end
    if P_ij ∪ S_ij = ⋆_l then
        Redun ← Redun ∪ {(v_i R_ij v_j)};
    end
end
```

5 Conclusion

In this paper, we have systematically investigated the computational complexity of redundancy checking for RCC5/8 constraints. Although it is in general co-NP-complete, we have shown that a prime network can be found in $O(n^5)$ for any consistent network over a tractable subclass of RCC5/8. If the constraints are taken from a distributive subalgebra, we proved that the core of the constraint network is the unique prime network and can be found in cubic time.

Some of these results (e.g., all results before Theorem 10) can be applied to many other qualitative calculi e.g. Interval Algebra (Allen 1983) immediately, but Proposition 11 and Theorem 12 do use the particular algebraic properties of RCC5/8. Future work will consider how to extend our results to these qualitative calculi.

Acknowledgments

The authors are grateful to Ross Purves, University of Zürich, for helping to develop the footprint data set illustrated in Fig. 1. This work was partially supported by ARC (DP120103758, DP120104159) and NSFC (61228305).

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11):832–843.

Belov, A.; Janota, M.; Lynce, I.; and Marques-Silva, J. 2012. On computing minimal equivalent subformulas. In *CP*, 158–174.

Chandra, P., and Pujari, A. K. 2005. Minimality and convexity properties in spatial CSPs. In *ICTAI*, 589–593.

Düntsch, I.; Wang, H.; and McCloskey, S. 2001. A relation-algebraic approach to the Region Connection Calculus. *Theor. Comput. Sci.* 255(1-2):63–83.

Jonsson, P., and Drakengren, T. 1997. A complete classification of tractability in RCC-5. *Journal of Artificial Intelligence Research* 6.

Li, S., and Ying, M. 2003. Region Connection Calculus: Its models and composition table. *Artif. Intell.* 145(1-2):121–146.

Liberatore, P. 2005. Redundancy in logic I: CNF propositional formulae. *Artif. Intell.* 163(2):203–232.

Ligozat, G., and Renz, J. 2004. What is a qualitative calculus? A general framework. In *PRICAI*, 53–64.

Liu, W., and Li, S. 2012. Solving minimal constraint networks in qualitative spatial and temporal reasoning. In *CP*, 464–479.

Randell, D. A.; Cui, Z.; and Cohn, A. G. 1992. A spatial logic based on regions and connection. In *KR*, 165–176.

Renz, J. 1999. Maximal tractable fragments of the Region Connection Calculus: A complete analysis. In *IJCAI*, 448–455.

Wallgrün, J. O. 2012. Exploiting qualitative spatial reasoning for topological adjustment of spatial data. In *SIGSPATIAL/GIS*, 229–238.

Belief Revision in the Propositional Closure of a Qualitative Algebra*

Valmi Dufour-Lussier,[1,2,3] **Alice Hermann,**[1,2,3] **Florence Le Ber,**[4] **Jean Lieber**[1,2,3]
[1] Université de Lorraine, LORIA, UMR 7503 — 54506 Vandœuvre-lès-Nancy, France
FirstName.LastName@loria.fr
[2] CNRS — 54506 Vandœuvre-lès-Nancy, France
[3] Inria — 54602 Villers-lès-Nancy, France
[4] ICube – Université de Strasbourg/ENGEES, CNRS — 67412 Illkirch, France
florence.leber@engees.unistra.fr

Abstract

Belief revision is an operation that aims at modifying old beliefs so that they become consistent with new ones. The issue of belief revision has been studied in various formalisms, in particular, in qualitative algebras (QAs) in which the result is a disjunction of belief bases that is not necessarily representable in a QA. This motivates the study of belief revision in formalisms extending QAs, namely, their propositional closures: in such a closure, the result of belief revision belongs to the formalism. Moreover, this makes it possible to define a contraction operator thanks to the Harper identity. Belief revision in the propositional closure of QAs is studied, an algorithm for a family of revision operators is designed, and an open-source implementation is made freely available on the web.

Keywords: qualitative algebras, belief revision, belief contraction, propositional closure

Introduction

Belief revision is an operation of belief change that consists in modifying minimally old beliefs so that they become consistent with new beliefs (Alchourrón, Gärdenfors, and Makinson 1985). One way to study this issue following a knowledge representation angle is to consider a formalism and to study some belief revision operators defined on it: how they are defined and how they can be implemented.

In particular, it is rather simple to define a revision operator on a qualitative algebra (QA, such as the Allen algebra) by reusing the work of (Condotta et al. 2010) about the related issue of belief merging. The result of such a belief revision is a set of belief bases to be interpreted disjunctively, and which is not necessarily representable as a single belief base: QAs are not closed under disjunction.

This gives a first motivation for the study of belief revision in the propositional closure of a QA: the revision operator in such a closure gives a result that is necessarily representable in the formalism. Another motivation lies in the possibility of defining a contraction operator in this formalism, thanks to the Harper identity.

*This research was partially funded by the project Kolflow (`http://kolflow.univ-nantes.fr`) of the French National Agency for Research (ANR), program ANR CONTINT.
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The first section of the paper contains some preliminaries about various notions used throughout the paper. The next section briefly describes some properties of such a formalism. Finally, an algorithm and an implementation of this algorithm for a revision operator in the propositional closure of a QA are presented with an example.

The research report (Dufour-Lussier et al. 2014) is a long version of this paper including more detailed preliminaries, the proofs and some additional examples.

Preliminaries

Qualitative algebras

Qualitative algebras (QAs) are formalisms that are widely used for representation depending on time and/or on space (Stock 1997). Formulas built upon QAs are closed under conjunction, though the symbol $\wedge$ is not systematically used. Some of the usual notations and conventions of QAs are changed to better fit the scope of this paper. In particular, the representation of knowledge by graphs (namely, qualitative constraint networks) is not well-suited here, because of the propositional closure introduced afterwards.

First, the Allen algebra is introduced: it is one of the most famous QAs and it will be used in our examples throughout the paper. Then, a general definition of QAs is given.

The Allen algebra is used for representing relations between time intervals (Allen 1983). A formula of the Allen Algebra can be seen as a conjunction of constraints, where a constraint is an expression of the form $x\ r\ y$ stating that the interval x is related to the interval y by the relation r. 13 base relations are introduced (cf. figure 1); a relation r is either one of these base relations or the union of base relations $r_1, \ldots, r_m$ denoted by $r_1 \mid \ldots \mid r_m$.

For example, if one wants to express that the maths course is immediately before the physics course which is before the English course (either with a time lapse, or immediately before it), one can write the formula:

$$\texttt{maths}\ \mathsf{m}\ \texttt{physics} \quad \wedge \quad \texttt{physics}\ \mathsf{b} \mid \mathsf{m}\ \texttt{english}$$

$\mathcal{L}_{\text{Allen}}$ is the set of the formulas of the Allen algebra.

Qualitative algebras in general are defined below, first by their syntax and then by their semantics.



bi, mi, oi, si, di and fi represent respectively the inverse relations of the relations represented by b, m, o, s, d and f.

Figure 1: The base relations of $\mathcal{L}_{\text{Allen}}$.

Syntax. A finite set of symbols $\mathfrak{B}$ is given (with $|\mathfrak{B}| \geq 2$). A *base relation* is an element of $\mathfrak{B}$. A *relation* is an expression of the form $r_1 \mid \ldots \mid r_m$ ($m \geq 0$), such that a base relation occurs at most once in a relation and the order is irrelevant (e.g. $r_1 \mid r_2$ and $r_2 \mid r_1$ are equivalent expressions). The set of relations is denoted by $\mathfrak{R}$.

A finite set of symbols $\mathcal{V}$, disjoint from $\mathfrak{B}$, is given. A *(qualitative) variable* is an element of $\mathcal{V}$.

A *constraint* is an expression of the form $x\ r\ y$ where $x, y \in \mathcal{V}$ and $r \in \mathfrak{R}$.

A *formula* φ is a conjunction of n constraints ($n \geq 1$): $x_1\ r_1\ y_1\ \wedge\ \ldots\ \wedge\ x_n\ r_n\ y_n$. A constraint of φ is one of the constraints of this conjunction. Let $\mathcal{L}_{\text{QA}}$ be the set of the formulas of the considered QA. The atoms of $\mathcal{L}_{\text{QA}}$ are the constraints.

A formula $\varphi \in \mathcal{L}_{\text{QA}}$ is under normal form if for every $x, y \in \mathcal{V}$ with $x \neq y$, there is exactly one $r \in \mathfrak{R}$ such that $x\ r\ y$ is a constraint of φ. Then, this relation r is denoted by $r_\varphi(x, y)$.

A *scenario* σ is a formula under normal form such that, for every variables x and y, $x \neq y$, $r_\sigma(x, y) \in \mathfrak{B}$. Therefore, there are $|\mathfrak{B}|^{|\mathcal{V}|\times(|\mathcal{V}|-1)}$ scenarios.

Semantics. The semantics of a QA can be defined classically, thanks to a domain, a variable being mapped into a subset of this domain and a relation being mapped on a relation between such subsets. For Allen algebra, the domain is the set $\mathbb{Q}$ of rational numbers and, given an interpretation $\mathcal{I}$, a variable x is mapped to an interval $\mathcal{I}(x) = [a, b]$ of $\mathbb{Q}$ ($a < b$). The semantics of each of the basic relations is defined. For example, $\mathcal{I}$ satisfies $x_1\ \mathsf{m}\ x_2$ if $b_1 = a_2$ where $\mathcal{I}(x_i) = [a_i, b_i]$. $\mathcal{I}$ satisfies $x_1\ (r_1 \mid \ldots \mid r_m)\ x_2$ if it satisfies one of the constraints $x_1\ r_k\ x_2$ for $k \in \{1, \ldots, m\}$. A formula is consistent (or satisfiable) if there exists an interpretation satisfying each of its constraints. Finally, for $\varphi_1, \varphi_2 \in \mathcal{L}_{\text{QA}}$, $\varphi_1 \models \varphi_2$ if, for every interpretation $\mathcal{I}$ satisfying φ_1, it satisfies also φ_2. The research report give more details on this first definition of the semantics.

The semantics can be characterized a posteriori thanks to consistent scenarios.

Let Ω be the set of consistent scenarios on the variables of $\mathcal{V}$. It can be proven that $|\Omega| \leq |\mathfrak{B}|^{|\mathcal{V}|\times(|\mathcal{V}|-1)/2}$: if $x\ r\ y$ is a constraint of a consistent scenario σ then $y\ r^-\ x$ is also a constraint of σ.

Let $\mathcal{M} : \mathcal{L} \to 2^\Omega$ be defined by

$$\mathcal{M}(\varphi) = \{\sigma \in \Omega \mid \sigma \models \varphi\}$$

for $\varphi \in \mathcal{L}$, where $\models$ is the entailment relation defined below, thanks to the semantics based on a domain.

Ω and $\mathcal{M}$ make it possible to define a semantics on $\mathcal{L}$ which coincides with the semantics based on a domain (hence the same entailment relation $\models$): $\varphi_1 \models \varphi_2$ iff $\mathcal{M}(\varphi_1) \subseteq \mathcal{M}(\varphi_2)$. However, this second semantics is more practical to use for defining revision operators on QAs.

Belief change

Belief revision is an operation of belief change. Intuitively, given the set of beliefs ψ an agent has about a static world, it consists in considering the change of their beliefs when faced with a new set of beliefs μ, assuming that μ is considered to be unquestionable by the agent. The resulting set of beliefs is noted $\psi \dot{+} \mu$, and depends on the choice of a belief revision operator $\dot{+}$. In (Alchourrón, Gärdenfors, and Makinson 1985), the principle of minimal change has been stated and could be formulated as follows: ψ is minimally changed into ψ' such that the conjunction of ψ' and μ is consistent, and the result of the revision is this conjunction. Hence, there is more than one possible $\dot{+}$ operator, since the definition of $\dot{+}$ depends on how belief change is "measured". More precisely, the minimal change principle has been formalized by a set of postulates, known as the AGM postulates (after the names of the authors of (Alchourrón, Gärdenfors, and Makinson 1985)).

In (Katsuno and Mendelzon 1991b), revision has been studied in the framework of propositional logic (with a finite set of variables). The AGM postulates are translated into this formalism and a family of revision operators is defined based on distance functions d on Ω, where Ω is the set of interpretations: the revision of ψ by μ according to $\dot{+}^d$ ($\psi\ \dot{+}^d\ \mu$) is such that

$$\mathcal{M}(\psi\ \dot{+}^d\ \mu) = \{\omega \in \mathcal{M}(\mu) \mid d(\mathcal{M}(\psi), \omega) = d^*\}$$
$$\text{with } d^* = d(\mathcal{M}(\psi), \mathcal{M}(\mu)) \quad (1)$$

Intuitively, d^* measures, using d, the minimal modification of ψ into ψ' needed to make $\psi' \wedge \mu$ consistent.

This approach can be extended to other formalisms for which a model-theoretic semantics can be defined and such that a distance function can be specified on the set of interpretations Ω. However, in some of these formalisms, a representability issue can be raised: it may occur that a subset Σ of Ω is not representable, i.e. there is no formula φ such that $\mathcal{M}(\varphi) = \Sigma$. This representability issue is addressed below, for the case of QAs.

Belief contraction is the operation of belief change that associates to a set of beliefs ψ and a set of beliefs μ, a set of beliefs $\psi \dot{-} \mu$ such that $\psi \dot{-} \mu \not\models \mu$. In propositionally closed formalisms, the Harper identity makes it possible to define a contraction operator $\dot{-}$ thanks to a revision operator $\dot{+}$ with

$$\psi \dot{-} \mu = \psi \vee (\psi \dot{+} \neg\mu) \quad (2)$$

Belief merging is another operation of belief change. Given some sets of beliefs $\psi_1, \ldots, \psi_n$, their merging is a set of beliefs Ψ that contains "as much as possible" of the beliefs in the ψ_i's. Intuitively, Ψ is the conjunction of $\psi'_1, \ldots, \psi'_n$ such that each ψ_i has been minimally modified into ψ'_i in order to make this conjunction consistent. Some postulates of belief merging have been proposed and discussed (Konieczny and Pérez 2002), in a similar way as the AGM postulates.

Belief revision in qualitative algebras

In (Condotta et al. 2010) a belief merging operator is defined that can be easily adapted for defining a revision operator. It is based on a distance between scenarios. Let δ be a distance function on $\mathfrak{B}$. Let $\sigma, \tau \in \Omega$, be two scenarios based on the same set of variables $\mathcal{V}$. Then, d is defined by

$$d(\sigma, \tau) = \sum_{x,y \in \mathcal{V}, x \neq y} \delta(r_\sigma(x,y), r_\tau(x,y))$$

One of the possibilities for δ is the use of a neighborhood graph, i.e. a connected, undirected graph whose vertices are the base relations: $\delta(r,s)$ is the length of the shortest path between r and s. Then, the scenarios of the revision of ψ by μ according to $\dot{+}^d$ are the scenarios of μ that are the closest ones to scenarios of ψ according to d. The set of optimal scenarios is not necessarily representable in $(\mathcal{L}_{\text{QA}}, \models)$. One solution to address this issue is to consider that the result of revision is a set of scenarios.

Algorithms for implementing this kind of belief revision in QAs are presented in (Dufour-Lussier et al. 2012) and (Hué and Westphal 2012).

Propositional closure of a qualitative algebra

The propositional closure of a QA $(\mathcal{L}_{\text{QA}}, \models)$ is a formalism $(\widehat{\mathcal{L}}_{\text{QA}}, \models)$ defined as follows. $\widehat{\mathcal{L}}_{\text{QA}}$ is the smallest superset of $\mathcal{L}_{\text{QA}}$ that is closed for $\neg$ and $\wedge$. Then, $\widehat{\mathcal{L}}_{\text{QA}}$ is closed for $\vee$: $\varphi_1 \vee \varphi_2$ is an abbreviation for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$. The entailment relation is based on the consistent scenarios: $\varphi_1 \models \varphi_2$ if $\mathcal{M}(\varphi_1) \subseteq \mathcal{M}(\varphi_2)$, $\mathcal{M}$ being extended on $\widehat{\mathcal{L}}_{\text{QA}}$ by $\mathcal{M}(\varphi_1 \wedge \varphi_2) = \mathcal{M}(\varphi_1) \cap \mathcal{M}(\varphi_2)$ and $\mathcal{M}(\neg\varphi) = \Omega \setminus \mathcal{M}(\varphi)$.

Proposition 1 (representability). *Every set of scenarios $\Sigma \subseteq \Omega$ is representable in $\widehat{\mathcal{L}}_{QA}$. More precisely, with $\varphi = \bigvee_{\sigma \in \Sigma} \sigma$, $\mathcal{M}(\varphi) = \Sigma$.*

Every formula of $\widehat{\mathcal{L}}_{\text{QA}}$ can be written in DNF (disjunctive normal form, i.e. disjunction of conjunctions constraints), since it is a propositionally closed formalism, but the following proposition goes beyond that.

Proposition 2 (DNF-w/oN form). *Every $\varphi \in \widehat{\mathcal{L}}_{QA}$ is equivalent to a formula in DNF using no negation symbol.*

Belief revision in $(\widehat{\mathcal{L}}_{\text{QA}}, \models)$

Given a distance function d on Ω, a revision operator on $(\widehat{\mathcal{L}}_{\text{QA}}, \models)$ can be defined according to equation (1). Indeed, proposition 1 implies that $\{\omega \in \mathcal{M}(\mu) \mid d(\mathcal{M}(\psi), \omega) = d^*\}$ is representable.

An algorithm for computing $\dot{+}^d$ in $\widehat{\mathcal{L}}_{\text{QA}}$

The principle of the algorithm is based on the following proposition.

Proposition 3 (revision of disjunctions). *Let ψ and μ be two formulas of $\widehat{\mathcal{L}}_{QA}$ and $\{\psi_i\}_i$ and $\{\mu_j\}_j$ be two finite families of $\widehat{\mathcal{L}}_{QA}$ such that $\psi = \bigvee_i \psi_i$ and $\mu = \bigvee_j \mu_j$.*

Let $d^*_{ij} = d(\mathcal{M}(\psi_i), \mathcal{M}(\mu_j))$ for any i and j. Then:

$$\psi \dot{+}^d \mu \equiv \bigvee_{i,j,d^*_{ij}=d^*} \psi_i \dot{+}^d \mu_j$$
$$\text{with } d^* = d(\mathcal{M}(\psi), \mathcal{M}(\mu))$$
$$\text{Moreover, } \quad d^* = \min_{ij} d^*_{ij} \tag{3}$$

The algorithm for $\dot{+}^d$ in $\widehat{\mathcal{L}}_{\text{QA}}$ consists roughly in putting ψ and μ in DNF-w/oN form then applying proposition 3 on them, using the $\dot{+}^d$ algorithm on $\mathcal{L}_{\text{QA}}$ for computing the $\psi_i \dot{+}^d \mu_j$'s. More details are given in the research report.

Implementation: the REVISOR/PCQA engine

REVISOR is a collection of several revision engines that are open-source and freely available.[1]

In particular, REVISOR/QA implements $\dot{+}^d$ in three QAs: the Allen algebra, INDU—an extension of the Allen algebra taking into account relations between intervals according to their lengths (Pujari, Kumari, and Sattar 1999)—and RCC8—a QA for representing topological relations between regions of space (Randell, Cui, and Cohn 1992). Moreover, it is easy to use a different qualitative algebra, by giving in the code some tables (composition table, inverse relation table, and table for the values $\delta(r,s)$ for $r, s \in \mathfrak{B}$). The engine is written in Perl, but can be used through a Java library. The worst-case complexity of this implementation is of the order of $O\left(|\mathfrak{B}|^{\frac{|\mathcal{V}|\cdot(|\mathcal{V}|-1)}{2}}\right)$.

REVISOR/PCQA implements $\dot{+}^d$ on the propositional closures of the QAs $\mathcal{L}_{\text{Allen}}$, INDU and RCC8: it actually uses REVISOR/QA and is one of the engines of REVISOR. The worst-case complexity of this implementation is of the order of $O\left(|\mathcal{V}|^4 |\mathfrak{B}|^{\frac{|\mathcal{V}|\cdot(|\mathcal{V}|-1)}{2}}\right)$, according to a coarse analysis.

The following example has been executed using REVISOR/PCQA, and is included with the source code. The `README` file associated with REVISOR/QA on the REVISOR website explains how it can be executed.

This example uses a belief contraction operator. According to (2), a contraction operator $\dot{-}^d$ can be defined that is based on $\dot{+}^d$. Now let us consider the set of beliefs ψ of an agent called Maurice about the dates of birth and death of famous mathematicians. Maurice thought that Boole was born after de Morgan and died before him and that de Morgan and Weierstraß were born the same year (say, at the same time) but the former died before the latter:

$\psi =$ `Boole d De Morgan` $\wedge$ `De Morgan s Weierstraß`

[1] http://revisor.loria.fr

where, Boole is the interval of time between the birth and the death of Boole, and so on. Now, Germaine, a friend of Maurice, tells him that she is not sure whether Boole was born strictly after Weierstraß. Since Maurice trusts Germaine (and her doubts), he wants to make the contraction of its original beliefs ψ by μ with

$$\mu = \texttt{Boole bi} \mid \texttt{mi} \mid \texttt{oi} \mid \texttt{f} \mid \texttt{d Weierstraß}$$

The result, computed by REVISOR/PCQA in less than one second, is $\psi \dot{-}^d \mu$, equivalent to the following formula:

$$\begin{array}{l} (\texttt{Boole d De Morgan} \wedge \texttt{De Morgan s Weierstraß}) \\ \vee (\texttt{Boole s Weierstraß} \wedge \texttt{De Morgan di Weierstraß}) \\ \vee (\texttt{Boole s De Morgan} \wedge \texttt{De Morgan s Weierstraß}) \\ \vee \begin{pmatrix} \texttt{Boole d De Morgan} \wedge \texttt{Boole s Weierstraß} \\ \wedge \texttt{De Morgan o Weierstraß} \end{pmatrix} \end{array}$$

Actually, the last term of this disjunction corresponds to the reality, provided that the intervals of time correspond to a year granularity: George Boole (1815-1864), Augustus De Morgan (1806-1871), Karl Weierstraß (1815-1897).

In (Dufour-Lussier et al. 2014), other examples of use of REVISOR/PCQA, including an analysis of the computing time, are presented. In particular, it is shown that it may be the case that a family of revision problems, albeit formalizable in both $\mathcal{L}_{\text{Allen}}$ and in $\widehat{\mathcal{L}}_{\text{Allen}}$, are solved in much less time in the more expressive formalism $\widehat{\mathcal{L}}_{\text{Allen}}$.

Conclusion

This paper has presented an algorithm for distance-based belief revision in the propositional closure $\widehat{\mathcal{L}}_{\text{QA}}$ of a qualitative algebra $\mathcal{L}_{\text{QA}}$, using the revision operation on $\mathcal{L}_{\text{QA}}$. This work is motivated by the fact that it gives a revision operation whose result is representable in the formalism, by the fact that some practical examples are easily represented in $\widehat{\mathcal{L}}_{\text{QA}}$ whereas they are quite difficult to represent in $\mathcal{L}_{\text{QA}}$, and by the fact that it makes it possible to define a contraction operator thanks to the Harper identity (which requires disjunction and negation). The preprocessing of the algorithm consists in putting the formulas into a disjunctive normal form without negation. Then, proposition 3, which reduces a revision of disjunctions to a disjunction of the least costly revisions, is applied. REVISOR/PCQA is an implementation of this revision operator for the Allen algebra, INDU and RCC8.

A first direction of research following this work is the improvement of the computation time of the REVISOR/PCQA system. One way to do it is to parallelize it. A sequential optimization would consist in finding a heuristic for ranking the pairs (i, j), with the aim of starting from the best candidates, in order to obtain a low upper bound of d^* sooner (hence a pruning of a part of the search trees developed for the computation of $\dot{+}^d$ in $(\mathcal{L}_{\text{QA}}, \models)$).

Another direction of research is to study how other belief change operations can be implemented in this formalism, in particular belief merging (Konieczny and Pérez 2002) and knowledge update (Katsuno and Mendelzon 1991a).

References

Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the Logic of Theory Change: partial meet functions for contraction and revision. *Journal of Symbolic Logic* 50:510–530.

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11):832–843.

Condotta, J.-F.; Kaci, S.; Marquis, P.; and Schwind, N. 2010. A Syntactical Approach to Qualitative Constraint Networks Merging. In *Proc. of the 17th LPAR (Logic for Programming, Artificial Intelligence and Reasoning)*, 233–247.

Dufour-Lussier, V.; Le Ber, F.; Lieber, J.; and Martin, L. 2012. Adapting Spatial and Temporal Cases. In Watson, I., and Díaz Agudo, B., eds., *International Conference for Case-Based Reasoning*, volume 7466 of *Lecture Notes in Artificial Intelligence*, 77–91. Lyon, France: Amélie Cordier, Marie Lefevre.

Dufour-Lussier, V.; Hermann, A.; Le Ber, F.; and Lieber, J. 2014. Belief revision in the propositional closure of a qualitative algebra (extended version). Technical report, LORIA, Université de Lorraine, CNRS. http://hal.inria.fr/hal-00954512.

Hué, J., and Westphal, M. 2012. Revising qualitative constraint networks: Definition and implementation. In *Tools with Artificial Intelligence (ICTAI)*, 548–555.

Katsuno, H., and Mendelzon, A. 1991a. On the Difference Between Updating a Knowledge Base and Revising It. In Allen, J. F.; Fikes, R.; and Sandewall, E., eds., *KR'91: Principles of Knowledge Representation and Reasoning*. San Mateo, California: Morgan Kaufmann. 387–394.

Katsuno, H., and Mendelzon, A. 1991b. Propositional knowledge base revision and minimal change. *Artificial Intelligence* 52(3):263–294.

Konieczny, S., and Pérez, R. P. 2002. Merging information under constraints: a logical framework. *Journal of Logic and Computation* 12(5):773–808.

Pujari, A. K.; Kumari, G. V.; and Sattar, A. 1999. INDU: An Interval & Duration Network. In Foo, N., ed., *Advanced Topics in Artificial Intelligence*, volume 1747 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 291–303.

Randell, D.; Cui, Z.; and Cohn, A. G. 1992. A spatial logic based on regions and connection. In *Knowledge Representation*, 165–176.

Stock, O., ed. 1997. *Spatial and Temporal Reasoning*. Kluwer Academic Publishers.

How to Argue for Anything: Enforcing Arbitrary Sets of Labellings Using AFs

Sjur K. Dyrkolbotn
Durham Law School, Durham University, UK
s.k.dyrkolbotn@durham.ac.uk

Abstract

We contribute to the investigation of possible outcomes of argumentation under semantics formulated using argumentation frameworks (AFs). In particular, we study this question for the labelling-based formulation of such semantics, generalizing previous work which has focused on extensions. In this paper, we restrict attention to the preferred and semi-stable semantics, showing that as long as we have a sufficient number of fresh arguments available, we can in fact argue for anything. That is, for any set of finite labellings there is an AF that enforces exactly this set as the outcome of argumentation.

1 Introduction

Formal argumentation in the style of (Dung 1995) is becoming an increasingly important formalism in artificial intelligence.[1] Dung observed that by representing argumentation scenarios by directed graphs, referred to as argumentation frameworks (AFs) in this context, various denotational semantics for non-monotonic reasoning could be formulated using graph-theoretic terms. Such semantics work by prescribing to an AF a set of *extensions*, sets of arguments that are regarded as successful when held together. In (Caminada 2006) it was shown that extensions could be defined in terms of three-valued labellings to the argument set, with every argument obtaining the status of being either accepted, defeated or undetermined. Hence an AF can be viewed as a theory in three-valued logic, and it has since been observed that Łukasiewicz logic is particularly well-suited for reasoning about argumentation (Dyrkolbotn and Walicki 2013; Dyrkolbotn 2013).

In recent work, the question of the *signature* of an argumentation semantics has been studied, asking about the structure of the different possible sets of extensions that can arise from an AF under a given semantics (Dunne et al. 2013). This work is interesting in that it promises to provide a bird's-eye view on the behavior of argumentation semantics, allowing us to develop a better understanding of their expressive power and meta-logical properties.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] We point to (Rahwan and Simari 2009) for a recent volume devoted to the use of abstract argumentation theory in AI.

However, developing this theory with respect to sets of extensions is unsatisfactory since the behavior of argumentation semantics is fundamentally three-valued. In particular, it seems to us that the question should also be studied when outcomes of argumentation are represented more realistically as sets of labellings. In this paper we make a contribution in this regard, by characterizing the possible outcomes of finite argumentation under the preferred and semi-stable semantics.[2] In particular, we show that *every* set of labellings can be enforced, as long as we have a sufficient number of additional arguments available. The proof is constructive and also demonstrates that the time needed to enforce a given set of labellings is linear in the number of possible labellings that are not in the outcome.

The structure of the paper is as follows. In Section 2 we present necessary background on AFs and semantics formulated using labellings. Then in Section 3 we prove the main result by constructing canonical AFs that can be used to enforce sets of labellings. In Section 4 we offer a conclusion.

2 Background

We give a terse background on AFs and the labelling-based approach to argumentation semantics, for more details we point to (Rahwan and Simari 2009, Chapter 2). First let us fix a countably infinite set Π of arguments. Then we can define an AF simply as a set of directed edges $E \subseteq \Pi \times \Pi$. If $(p, q) \in E$ we think of it as encoding the fact that p represents an argument that attacks the argument represented by q. We use the notation $E^+(x) = \{y \in \Pi \mid (x, y) \in E\}, E^-(x) = \{y \in \Pi \mid (y, x) \in E\}$, and extend it to sets such that $E^*(A) = \bigcup_{x \in A} E^*(x)$ for $* \in \{+, -\}$. We also define $\Pi(E) = \{x \mid E^+(x) \cup E^-(x) \neq \emptyset\}$, the set of arguments from Π which appear in some attack from E. We will often present AFs by simply depicting the attacks they contain, as in Figure 1.

Given an AF E, an argumentation semantics is used to identify sets of arguments that can be held successfully together. Typically, this involves various formalizations of the

[2] We remark that (Dunne et al. 2013) also notes the possibility of studying signatures with respect to labellings rather than extensions. They have little to say about it, however, apart from listing it as an important direction for future work.



Figure 1: An AF E such that $\Pi(E) = \{p, q, q', p'\}$

intuition that the set should be internally consistent and able to defend itself against attacks from other arguments. Different semantics differ about the details, but they have the same signature: they are defined as an operator $\mathcal{S}$ which takes an AF E and returns a set of sets of $\mathcal{S}(E) \subseteq 2^{\Pi}$, the set of acceptable sets, called extensions. Moreover, to the best of our knowledge, all semantics share the property that arguments in an extension should be internally consistent, free of internal conflict. Formally, for all such semantics $\mathcal{S}$, all AFs E and all $A \in \mathcal{S}(E)$, we have $E^-(A) \subseteq \Pi \setminus A$, such that no two arguments in A attack each other.

For a given argument, it is accepted or it is not, but a boolean perspective fails to do justice to the nature of the structure $(E, \mathcal{S})$ in two important ways. First, it is not clear whether we should say that p is accepted on E under $\mathcal{S}$ when there *exists* some $A \in \mathcal{S}(E)$ such that $p \in A$, or whether we should require $p \in A$ for *all* such A. Both notions of acceptance have been used, and the former is typically dubbed *credulous* while the latter is referred to as *skeptical*.[3]

The second sense in which acceptance is not a binary notion has to do with the structure of E. In particular, given any $A \in \mathcal{S}(E)$ the status of p with respect to A can be any of the following:

$$\begin{array}{ll} 1 : p \in A & 2 : p \in E^+(A) \\ 3 : p \in \Pi \setminus (A \cup E^+(A)) & \end{array} \quad (1)$$

Notice that by conflict-freeness of A, it follows that if $p \in E^+(A)$ then $p \notin A$. Hence when the focus is on the status of individual arguments, we might as well view $\mathcal{S}(E)$ as a set of partitions of Π into three disjoint sets or, equivalently, as a collection of *labellings* (Caminada 2006), functions $\mathsf{c} : \Pi \to \{1, \frac{1}{2}, 0\}$ such that for all $x \in \Pi$:

$$\mathsf{c}(x) = 0 \iff \exists y \in E^-(x) : \mathsf{c}(y) = 1 \quad (2)$$

For any AF E we let $\mathsf{cf}(E)$ be the set of all labellings for E, and we define $\mathsf{c}^1 = \{x \in \Pi \mid \mathsf{c}(x) = 1\}, \mathsf{c}^0 = \{x \mid \mathsf{c}(x) = 0\}$ and $\mathsf{c}^{\frac{1}{2}} = \{x \in \Pi \mid \mathsf{c}(x) = \frac{1}{2}\}$. This, in particular, defines a semantics for argumentation such that for all E, we regard $A \subseteq \Pi$ as acceptable if there is some $\mathsf{c} \in \mathsf{cf}(E)$ such that $\mathsf{c}^1 = A$.[4] In applications of argumentation theory, this is usually considered too permissive, and a range of various restrictions has been considered, each giving rise to a new semantics, the most well-known of which are defined in Figure 2.

Admissible:	$a(E) = \{\mathsf{c} \in \mathsf{cf}(E) \mid E^-(\mathsf{c}^1) \subseteq \mathsf{c}^0\}$
Complete:	$c(E) = \{\mathsf{c} \in \mathsf{cf}(E) \mid \mathsf{c}^1 = \{x \in \Pi \mid E^-(x) \subseteq \mathsf{c}^0\}\}$
Grounded:	$g(E) = \{\bigcap c(E)\}$
Preferred:	$p(E) = \{\mathsf{c}_1 \in a(E) \mid \forall \mathsf{c}_2 \in a(E) : \mathsf{c}_1^1 \not\subset \mathsf{c}_2^1\}$
Semi-stable:	$ss(E) = \{\mathsf{c}_1 \in a(E) \mid \forall \mathsf{c}_2 \in a(E) : \mathsf{c}_1^{\frac{1}{2}} \not\supset \mathsf{c}_2^{\frac{1}{2}}\}$
Stable:	$s(E) = \{\mathsf{c} \in a(E) \mid \mathsf{c}^{\frac{1}{2}} = \emptyset\}$

Figure 2: Various semantics, defined for any $E \subseteq \Pi \times \Pi$

To illustrate the definition, consider the AF in Figure 1. We notice, in particular, that the set of admissible and complete extensions is $a(E) = c(E) = \{\emptyset, \{q\}, \{q'\}\}$, that the grounded extension is empty, that the preferred and semi-stable extensions are $p(E) = ss(E) = \{\{q\}, \{q'\}\}$ and that E does not have any stable extensions (the labellings corresponding to these sets can be easily recovered using Equation (1)).

We can now illustrate why the approach from (Dunne et al. 2013), which focuses on extensions rather than labellings, fails to do full justice to the question of characterizing the possible outcomes of argumentation. For instance, consider enforcing the outcome where one argument, call it w, is accepted, while all other arguments from Π are defeated. That is, there should be only one acceptable labelling and it should assign 1 to w and 0 to all other arguments. Looking at extensions alone there is no way to distinguish this from the case when w is accepted and all other arguments are undecided: in both cases it will hold that $\{w\}$ is the only extension. If we go on by attempting to enforce the outcome that *all* arguments in Π are rejected we come across a *validity* of argumentation that (Dunne et al. 2013) is unable to capture. It is easy to see, in particular, that while many AFs exist for which the empty set is the only admissible extension, it is impossible to construct an AF such that every argument is defeated, since then no arguments labelled 1 would be around to defeat them.

Due to validities of this type, it seems difficult in general to characterize the set of all infinite labellings that can be enforced without the addition of new arguments. In the following we therefore restrict attention to finite AFs, a restriction which is also made in (Dunne et al. 2013). The upshot is that we can always assume availability of additional arguments to help ensure that a given outcome for $V \subset \Pi$ is obtained.[5]

[3] It is natural to view skeptical and credulous acceptance as dual *modalities*, suggesting the study of the set of validities characterizing their interactions. This, in particular, is a different approach to modal reasoning about AFs than that explored in (Grossi 2010b; 2010a), where modalities are used to talk about AFs, to allow semantics to be defined in terms of modal formulas addressing the graph structure. It is closer to what is called an "object level" approach in (Caminada and Gabbay 2009), where arguments are regarded as atoms in a propositional language.

[4] Hence it is not hard to see that values assigned by labellings correspond to the three points of Equation 1 whenever we restrict attention to conflict-free sets of accepted arguments. Notice, in particular, that $p \in \mathsf{c}^0 \iff p \in E^+(\mathsf{c}^1)$ and $p \in \mathsf{c}^{\frac{1}{2}} \iff p \in \Pi \setminus (\mathsf{c}^1 \cup \mathsf{c}^0)$

[5] We remark that in the context of logic-based approaches to argumentation, such as those explored in (Dyrkolbotn and Walicki 2013; Dyrkolbotn 2013), this restriction is suitable as long as we do not consider languages with *infinitary* connectives. Moreover, we note that while the finiteness restriction is not in itself crucial and may be dropped, the presence of a sufficient amount of additional arguments is indeed needed for the results presented in the next section.

3 Enforcing an Arbitrary Set of Labellings

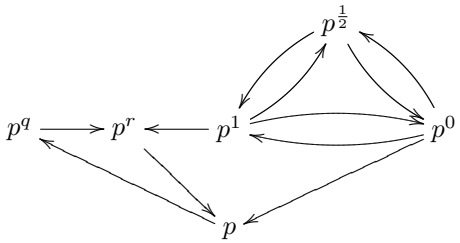
We now proceed to show how canonical AFs can be used to enforce arbitrary sets of labellings to finite $V \subset \Pi$ under preferred and semi-stable semantics. The same AFs will be canonical for both, so our result is relevant also to the investigation of translations between semantics (Dvorák and Woltran 2011; Dvorák and Spanring 2012). In particular, our constructions provide an alternative to the approach of modifying AFs so that their behavior under one semantics can be captured by another. For *any* argumentation semantics which prescribes labellings, we can capture it using preferred and semi-stable semantics, by using the canonical AF.

To construct it formally, assume we are given a finite $V \subseteq \Pi$ and a set of three-valued assignments $F \subseteq \left\{1, \frac{1}{2}, 0\right\}^V$. Then we will construct E_F and show that it enforces F on V in the following sense, for $\mathcal{S} \in \{p, ss\}$:

$$\{\mathsf{c}|_V \mid \mathsf{c} \in \mathcal{S}(E_F)\} = F \tag{3}$$

We rely on two basic constructions to show that there exists E_F which satisfies this equation, for all V, F. First we introduce, for all $p \in \Pi$, the AF Ł_p, which we refer to as a *circuit* for p. It is defined as depicted below.

(4)



The reader may easily verify that Ł_p satisfies the following property, for both preferred and semi-stable $\mathcal{S}$:

$$\{\mathsf{c}|_p \mid \mathsf{c} \in \mathcal{S}(\text{Ł}_p)\} = \{\{p \mapsto 0\}, \{p \mapsto 1\}, \{p \mapsto \frac{1}{2}\}\} \tag{5}$$

That is, in the AF Ł_p, the semantic status of p is completely open: for any of the three values p could have, the preferred and semi-stable semantics admit labellings for Ł_p that give p this value. This implies that for any finite set $V = \{p_1, \ldots, p_n\}$ there is an AF which characterizes the set $X = \left\{1, \frac{1}{2}, 0\right\}^V$, containing *all* functions from V to $\left\{1, \frac{1}{2}, 0\right\}$. In particular, we can take E_X to be the following AF (assuming, of course, that all arguments p_i^x used to construct a circuit for p_i are fresh, i.e. do not occur in V).

$$E_X = \bigcup_{1 \leq i \leq n} \text{Ł}_{p_i} \tag{6}$$

To show the general claim, for arbitrary $F \subseteq X$, we will start from E_X and then inductively define E_F in terms of $E_{F'}$ such that $F = F' \setminus \{f\}$. We will show, in particular, how to obtain E_F from $E_{F'}$ by ensuring that f becomes *forbidden* while all other assignments from F' are still permitted. To explain how to do this, let us first observe the following crucial property, allowing us to characterize any of the three values that p_i can have by an argument that is accepted if, and only if, p_i has this value. In particular, the reader can easily verify that the following holds, for preferred and semi-stable $\mathcal{S}$ and all $\mathsf{c} \in \mathcal{S}(\text{Ł}_{p_i})$:

$$\begin{cases} \mathsf{c}(p_i) = 1 \iff \mathsf{c}(p_i^1) = 1 \\ \mathsf{c}(p_i) = 0 \iff \mathsf{c}(p_i^0) = 1 \\ \mathsf{c}(p_i) = \frac{1}{2} \iff \mathsf{c}(p_i^{\frac{1}{2}}) = 1 \end{cases} \tag{7}$$

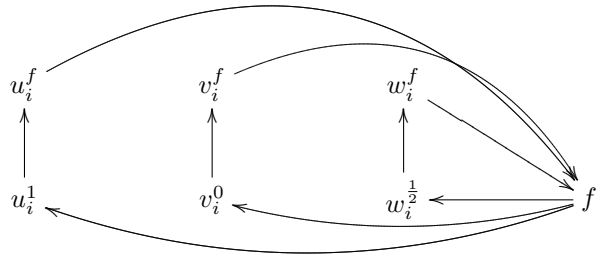
Notice also that $\mathsf{c}(p_i^y) \in \{1, 0\}$ for all y, i and $\mathsf{c} \in \mathcal{S}(\text{Ł}_{p_i})$. That is, the arguments witnessing to the value of p_i are always assigned one of the boolean values. This will be important to keep in mind below. Now, assume as induction hypothesis that we have been able to define $E_{F'}$ for $F' = F \cup \{f\}$ based on E_X such that Equation (3) holds for F', and such that Equation (7) holds for all $\mathsf{c} \in \mathcal{S}(E_{F'})$. To define E_F we first partition V according to the values assigned by f:

$$V = V_1 \cup V_0 \cup V_{\frac{1}{2}}$$

where $V_1 = \{x \mid f(x) = 1\} = \{u_1, \ldots, u_k\}, V_0 = \{x \mid f(x) = 0\} = \{v_1, \ldots, v_l\}, V_{\frac{1}{2}} = \{x \mid f(x) = \frac{1}{2}\} = \{w_1, \ldots, w_m\}$ (notice that this implies that all V_i's are disjoint). Next we let $\{f, x^f \mid x \in V\}$ be a collection of $|V|+1$ fresh arguments, and we define E_F as follows:

$$\begin{aligned} E_F = E_{F'} &\cup \bigcup_{1 \leq i \leq k} \{(u_i^1, u_i^f), (u_i^f, f), (f, u_i^1)\} \\ &\cup \bigcup_{1 \leq i \leq l} \{(v_i^0, v_i^f), (v_i^f, f), (f, v_i^0)\} \\ &\cup \bigcup_{1 \leq i \leq m} \{(w_i^{\frac{1}{2}}, w_i^f), (w_i^f, f), (f, w_i^{\frac{1}{2}}\} \end{aligned}$$

In the figure below we sketch this construction, depicting the pattern made up of the edges added in the induction step.



(8)

First let us note that Equation 7 holds for all $\mathsf{c} \in \mathcal{S}(E_F)$. Indeed, none of the edges that have been added interfere with the fact that for all $x_i \in V$, we have $\mathsf{c}(x_i^y) = 1$ if, and only if, $\mathsf{c}(x_i) = y$. It is then straightforward to establish that there is no $\mathsf{c} \in \mathcal{S}(E_F)$ such that $\mathsf{c}|_V = f$. In particular, if there was such a c it would follow that all arguments of the form $x_i^{f(x_i)}$ would be assigned 1, so all attackers of f would be labelled 0, which in turn would imply that the fresh argument f would be labelled 1. This would be contradiction, however, since f attacks all $x_i^{f(x_i)}$. On the other hand, it is easy to see that any other $f' \in F'$ is still permitted: as long as one or more arguments from V are labelled by something

different from their value under f, it follows from Equation (7) that the argument named f will be rejected, thus rendering all new attacks on existing arguments, which all come from f, irrelevant to the labelling of arguments from V.

Having demonstrated how to construct the desired AF E_F by induction, we conclude as follows.

Theorem 3.1. *For all finite $V \subseteq \Pi$, and all $F \subseteq \{1, \frac{1}{2}, 0\}^V$, there is a finite AF E_F such that for all $\mathcal{S} \in \{p, ss\}$ we have*

$$\{\mathsf{c}|_V \mid \mathsf{c} \in \mathcal{S}(E_F)\} = F$$

The proof of this result essentially consisted in an algorithm for computing E_F. To analyze the complexity of this construction, notice first that E_X has linear size in V. Moreover, remember that we needed $|V| + 1$ new arguments in each step of the procedure, and that the number of total steps required was equal to the number of forbidden labellings. In light of this, we obtain the following corollary.

Corollary 3.2. *For any finite $V \subseteq \Pi$ and any $F \subseteq \{1, \frac{1}{2}, 0\}^V$, we can construct E_F that enforces F under preferred and semi-stable semantics in time $\mathcal{O}(|V| \times |\{1, \frac{1}{2}, 0\}^V \setminus F|)$.*

Before we conclude, we comment briefly on the other semantics from Figure 2. It is easy to see, in particular, that Theorem 3.1 does not hold for any of the other semantics defined there. For the stable semantics this is trivial since it only permits boolean labellings.[6] For the other semantics, notice that all of them always prescribe a labelling such that all other permissible labellings extend it (the empty labelling for the admissible semantics, the grounded labelling for the others). This means, in particular, that no set of labellings can be enforced if it fails to include this labelling.

4 Conclusion

We have studied the finite signatures of the preferred and semi-stable semantics, in terms of the sets of finite labellings they may give rise to. As it turns out, any set of labellings is a possible outcome under these semantics, so in a sense we have indeed shown how to argue for anything. Moreover, while the construction we used takes exponential time to compute in general, it is linear in an interesting parameter: the number of labellings that are *not* supposed to be in the outcome.

This result provides theoretical insight concerning the preferred and semi-stable semantics and it also strengthens the connection between argumentation and three-valued Łukasiewicz logic. As observed in (Dyrkolbotn 2013), any semantics applied to an AF can be seen as a theory in this logic. Hence Theorem 3.1 serves to complete the picture by showing that any finite theory of Łukasiewicz logic can also be represented as an AF. That is, for any such theory there is a corresponding AF such that a three-valued assignment to atoms is a model of the theory if, and only if, it corresponds to a preferred/semi-stable labelling for the corresponding AF.

This is interesting in its own right, but also suggests the possibility of obtaining completeness results for modal logics that allow us to reason about skeptical and credulous acceptance under preferred and semi-stable semantics. It seems, in particular, that we are able to instantiate a canonical class of three-valued doxastic Kripke frames using AFs.[7] This will be explored further in future work.

[6] However, notice that the argument used to prove Theorem 3.1 can be easily adapted to show that the stable semantics is canonical for boolean theories.

[7] In particular, by taking labellings as worlds and letting belief sets be induced by an AF under an argumentation semantics.

References

Caminada, M. W. A., and Gabbay, D. M. 2009. A logical account of formal argumentation. *Studia Logica* 93(2-3):109–145.

Caminada, M. W. A. 2006. On the issue of reinstatement in argumentation. In Fisher, M.; van der Hoek, W.; Konev, B.; and Lisitsa, A., eds., *JELIA*, volume 4160 of *Lecture Notes in Computer Science*, 111–123. Springer.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77:321–357.

Dunne, P. E.; Dvorák, W.; Linsbichler, T.; and Woltran, S. 2013. Characteristics of multiple viewpoints in abstract argumentation. In Beierle, C., and Kern-Isberner, G., eds., *4th Workshop on Dynamics of Knowledge and Belief (DKB-2013)*. FernUniversität in Hagen. 16–31.

Dvorák, W., and Spanring, C. 2012. Comparing the expressiveness of argumentation semantics. In Verheij, B.; Szeider, S.; and Woltran, S., eds., *COMMA*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, 261–272. IOS Press.

Dvorák, W., and Woltran, S. 2011. On the intertranslatability of argumentation semantics. *J. Artif. Intell. Res. (JAIR)* 41:445–475.

Dyrkolbotn, S. K., and Walicki, M. 2013. Propositional discourse logic. *Synthese* 1–37. Available online first at Springer.

Dyrkolbotn, S. K. 2013. The same, similar, or just completely different? Equivalence for argumentation in light of logic. In Libkin, L.; Kohlenbach, U.; and de Queiroz, R. J. G. B., eds., *WoLLIC*, volume 8071 of *Lecture Notes in Computer Science*, 96–110. Springer.

Grossi, D. 2010a. Argumentation in the view of modal logic. In McBurney, P.; Rahwan, I.; and Parsons, S., eds., *ArgMAS*, volume 6614 of *Lecture Notes in Computer Science*, 190–208. Springer.

Grossi, D. 2010b. On the logic of argumentation theory. In van der Hoek, W.; Kaminka, G. A.; Lespérance, Y.; Luck, M.; and Sen, S., eds., *AAMAS*, 409–416. IFAAMAS.

Rahwan, I., and Simari, G. R., eds. 2009. *Argumentation in artificial intelligence*. Springer.

Knowledge Maps of Web Graphs

Valeria Fionda[1], **Claudio Gutierrez**[2], **Giuseppe Pirrò**[3]
[1] Department of Mathematics, University of Calabria
[2] DCC, Universidad de Chile
[3] KRDB, Free University of Bolzano

Abstract

In this short note we give an overview of our research concerning cartography on the Web and its challenges. We present a mathematical formalism to capture the notion of map on the Web, which allows to automatize the construction of maps.

1 Introduction

Cartography is the art of making maps. Its essential aim is that of representing the characteristics of a region of interest. Cartography builds upon two main steps: *selection* and *abstraction* (Robinson et al. 1995). Selection enables to focus only on the particular pieces of information that will serve the map's purpose; in this phase the region of the space to be mapped is chosen. Abstraction is the fundamental property of a map, which states that a map should be smaller than the region it portraits. Thus, a map can be simply defined as an abstract representation of a region of interest.

The Web is a large and interconnected information space commonly accessed and explored via navigation. This space is simply too large and its interrelations too complex for anyone to grasp its content by direct observation. Hence, the possibility of applying cartographic principles to the Web space becomes a relevant matter. Knowledge maps can be useful cues that help to navigate, find routes and discover previously unknown connections between knowledge items on the Web. Effectively, they can play the role of *navigational charts* helping users to cope with the size of the Web (cyber)space (Dodge and Kitchin 2001). Users via knowledge maps can track, record, identify and abstract conceptual regions of information on the Web, for their own use, for sharing/exchanging with other users and/or for further processing (e.g., combination with other maps). Maps are also useful to analyze information. For instance, the availability of a series of chronologically sequential maps enables complex map analysis (e.g., longitudinal analysis) for the detection and forecasting of trends in specific domains (Garfield 1994). This is useful, for instance, in the analysis of the knowledge flows in scientific literature, which helps in understating how the interlinking between disciplines is changing (Rosvall and Bergstrom 2008). Maps of social networks

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

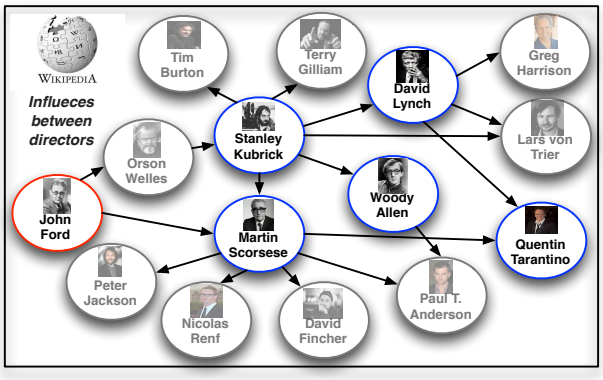


Figure 1: A region of the Web taken from Wikipedia.

can be analyzed to forecast friendship (Ledbetter, Griffin, and Sparks 2007).

The progresses in Web technologies and languages originating from the Semantic Web proposal and, in particular the availability of structured and machine-readable information (RDF data model and SPARQL language) open new perspectives toward the feasibility of knowledge maps of the Web. Now it is possible to leverage logical/algebraic methods to organize and specify in a formal, machine-understandable and reusable way regions and maps of the Web. In this short note we introduce the foundations of a formal notion of map for the Web that will enable the interpretation of maps not only by humans but also by machines, fostering in this way their exchange, combination and reuse at Web level. The following example gives an idea of how a knowledge map of the Web looks like.

Example 1 *(**Mapping favorite directors**). Syd is fond of cinema and wants to discover and keep track of his favorite directors and relations among them.*

The first step toward building the map of *Syd*'s favorite directors is to *select* the region of interest. This could be done by *tracking* the navigational activity of *Syd* on Wikipedia; Syd starting from the page of one of his favorite directors (i.e., J. Ford) has navigated toward directors influenced by J. Ford and manually (book)marked some of them, that is, S. Kubrick, W. Allen, D. Lynch, M. Scorsese and Q. Tarantino.

Note that the region besides these *distinguished* nodes also contains other (lighter) nodes that were visited during the navigation (see Fig. 1). Once the region in Fig. 1 has been obtained, it has to be *abstracted* in a knowledge map. Fig. 2 shows two possible maps of the region in Fig. 1. Which one is a "better" representation of the original region?

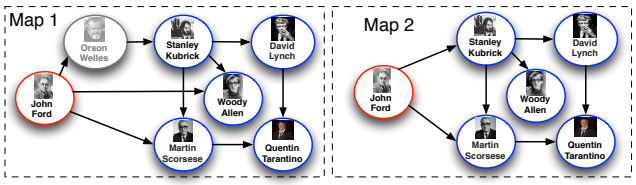


Figure 2: Two possible maps of the region in Fig. 1.

The challenge is to identify a suitable definition of knowledge map that provides an economic and informative representation of the region of interest. Essentially, a map must reflect in a *concise way* information in the region in terms of *connectivity* among some distinguished nodes. What are the formal property of knowledge maps of the Web? Answering these questions enables to have a clear understanding of the relations between regions and maps and also to define means to represent, exchange, reuse and combine different maps (i.e., via an algebra). In this short note we provide a formalization of the notions of region and map of the Web. We discuss several types of maps, present algorithms for constructing such maps and a concrete example of Web map.

2 Formalizing Maps

One of the main challenges toward building maps of the Web is how to move from human-readable maps towards maps that can also be understood and manipulated by machines. This requires the development of a formal notion of map and the investigation of its properties/relations with the region it represents. Let us introduce some notations and basic definitions. Let $\Gamma = (V, E)$ be a graph where V is the set of nodes and E is the set of edges. Then:

- $u \to v$ denotes an edge $(u, v) \in E$. As an example, S. Kubrick $\to$ W. Allen in Fig. 1.
- $u \twoheadrightarrow v$ denotes a path from u to v in Γ. As an example, J. Ford $\twoheadrightarrow$ S. Kubrick in Fig. 1.
- Let $N \subseteq V$. Then, $u \twoheadrightarrow_N v$ if and only if there is a path from u to v in Γ not passing through intermediate nodes in N. As an example, if $N = \{$M. Scorsese, W. Allen$\}$ then J. Ford $\twoheadrightarrow_N$ S. Kubrick but not J. Ford $\twoheadrightarrow_N$ P. T. Anderson in Fig. 1.

Definition 2 (Map) *Let $\Gamma = (V, E)$ be a graph. A* map *$M = (V_M, E_M)$ is a graph s.t. $V_M \subseteq V$ and each edge $(x, y) \in E_M$ implies $x \twoheadrightarrow y$ in Γ.*

By making a parallel with cartography, the graph Γ is the region (or "territory") to be represented/abstracted by the map M. The set of nodes V_M in M represent "points" relevant for the map; these are the *distinctive traits* used as the basis to *identity of the region*. The set of edges E_M in the map M represent the "directions", "routes" and signals connecting different points of interest. A basic (and highly used) example of maps of the Web are bookmarks. In this case, V_M is the set of nodes highlighted or marked, and $E_M = \emptyset$, that is, there is no connectivity recorded among them. More elaborate maps can be built from citation graphs (e.g., Google Scholar); here V_M is a set of URIs of papers of interest and E_M is the citation connectivity graph one can navigate.

Fig. 3 shows some possible maps obtained from the graph in Fig. 1. The map in Fig. 3 (a) is an example of bookmarks. Note that maps (a)-(c) do not capture connectivity among the distinguished nodes. For instance, J. Ford and S. Kubrick that are both distinguished nodes connected by a path in the original region, are not connected in these maps. This points to the notion of completeness of a map.

Definition 3 (Complete Map) *A* map *is complete if $\forall x, y \in V_M$, $x \twoheadrightarrow y$ in Γ implies $x \twoheadrightarrow y$ in M.*

In Fig. 3, maps (d)-(f) are examples of complete maps. However, even completeness is not enough to abstract information. Consider the map in Fig. 3 (d). In the region in Fig. 1 there is an edge connecting J. Ford and M. Scorsese. In the map, although there is a path connecting (via S. Kubrick) the two nodes, the fact that there is an alternative connection (not passing through nodes in V_M) between them is missing. The following notion captures this issue:

Definition 4 (Route-complete map) *A* map *M of Γ is route-complete iff $\forall x, y \in V_M$ $x \twoheadrightarrow_{V_M} y$ in Γ implies $x \to y$ in M.*

By looking at Fig. 3, map (d) is not route-complete while maps (e) and (f) are route-complete. Indeed, for each path between distinguished nodes not passing through any other distinguished node in the region, there is an edge in the map. As an example, maps (e) and (f), both introduce some direct edges like, for instance, J. Ford$\to$S. Kubrick (i.e., e_1) that were not present in the original region; the edge e_1 *abstracts* the path J. Ford$\to$O. Welles$\to$S. Kubrick.

But there are still some issues to address. Consider the edge e_2 between J.Ford and Q.Tarantino in map (f). All paths connecting J.Ford to Q.Tarantino in the region pass through some distinguished nodes (i.e., M.Scorsese, S.Kubrick, D.Lynch) and connectivity information between J.Ford and Q.Tarantino is already encoded in the map via the paths Ford $\to$ Scorsese $\to$ Tarantino, Ford $\to$ Kubrick $\to$ Scorsese $\to$ Tarantino and Ford $\to$ Kubrick $\to$ Lynch $\to$ Tarantino.
Thus, edge e_2 does not add new connectivity information, i.e., it is in some sense redundant.

Definition 5 (Non-redundant map) *A* map *M of Γ is non-redundant iff $\forall x, y \in V_M$ $x \to y$ in M implies $x \twoheadrightarrow_{V_M} y$ in Γ.*

Example of non-redundant maps are reported in Fig. 3 (d) and (e). However, recall that (d) does not satisfy the property of route-completeness. In order to capture completeness, route-completeness and non-redundancy we introduce *good maps*.

Definition 6 (Good map) *A* map *M of Γ is good if and only if it is complete, route-complete and non-redundant.*

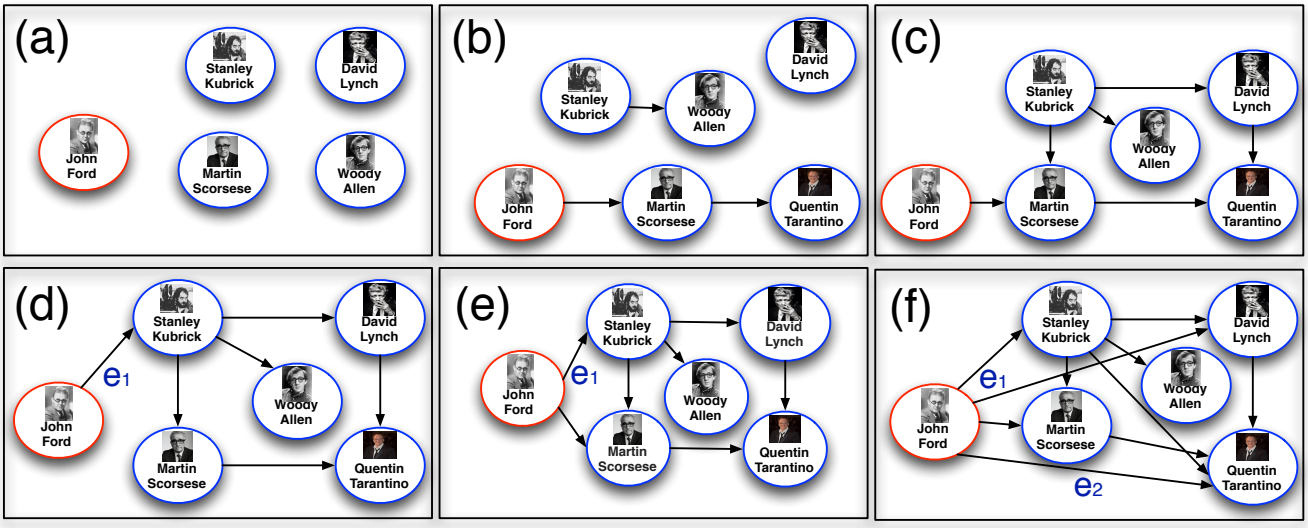


Figure 3: Examples of maps obtained from the region and the distinguished nodes (the set V_M) in Fig. 1.

Thus, good maps comprise the basic features we would like to have in a map in order to faithfully and economically capture connectivity. Fig. 3 (e) shows the good map of the region in Fig. 1. Indeed it is the *unique* good map of such a region. This is not a coincidence. Next theorem give a simple algorithmic characterization of good maps and shows that they are unique:

Theorem 7 *Let $\Gamma = (V_\Gamma, E_\Gamma)$ be a region. Then:*

1. A map M of Γ is good iff $\forall x, y \in V_M$, $x \to y$ in $M \Leftrightarrow x \twoheadrightarrow_{V_M} y$ in Γ.

2. Given $V_M \subseteq V_\Gamma$, there is a unique good map $M = (V_M, E_M)$ over Γ.

Good maps behave very well from algebraic and computationally point of views. In this short note we do not have space to exhibit further formal properties (operations between them, algorithms for computing, etc.), which will be presented in an extended version.

3 Maps for the Web of Data

Our motivation behind the formalism for maps is to develop knowledge maps of the Web graph. The availability of structured interlinked data on the Web (in the form of RDF) is evolving the traditional Web of Documents into a semantic distributed graph known as Web of Linked Data (WoD) (Heath and Bizer 2011), which we will denote as $\mathcal{W} = (V, E)$. Here, nodes are RDF data sources (identified by URIs) and edges RDF predicates. In this section we discuss how to instantiate the conceptual framework introduced in the previous section in the Web of Linked Data.

The problem of building maps in this scenario, given the machinery to build maps already presented, can be reduced to identify regions of interest:

Problem 1 *Given the WoD graph $\mathcal{W} = (V, E)$, construct a subgraph (region) $S = (V', E')$ of $\mathcal{W}$ that contains some distinguished nodes of interest $N \subseteq V'$.*

Graph navigational languages partially address this problem. Current navigational languages (e.g. XPath (W3C 1999), nSPARQL (Pérez, Arenas, and Gutierrez 2010), NautiLOD (Fionda, Gutierrez, and Pirrò 2012), etc.) enable to navigate a labeled graph essentially via matching some pattern (or navigational expression) expressed by means of a regular expression over the alphabet of edge labels. Their output is a set of nodes satisfying the matchings. For our purposes here, their drawback is that the information used during the navigation is forgotten, that is, the semantics uses the navigation only as the means to reach the resulting *set of nodes*. This is not enough for our goal: we need the missing information about connectivity in order to deal with regions and maps.

To cope with this problem, we extend the semantics of navigational languages to incorporate in their results not only the final set of nodes, but some of the paths followed during the evaluation. This allows to build regions instead of getting only sets of nodes. We consider the navigational language $\mathcal{L}$ that abstracts the core of most existing path languages. Its syntax is:

path::=<RDF predicate>| path[test] | (path)* | (path|path) | path/path

The core of $\mathcal{L}$ are regular expressions built upon RDF predicates and boolean tests used to drive the navigation. Tests can be of different types, for instance, ASK SPARQL queries (as in NautiLOD (Fionda, Gutierrez, and Pirrò 2012)) and/or nested regular expressions (Pérez, Arenas, and Gutierrez 2010). We are now ready to provide a concrete example of map.

Example 8 *Specify a region that contain scientists that have been influenced, directly or indirectly up to distance 6, by Tim Berners-Lee (TBL). Build the good map of the obtained region.*

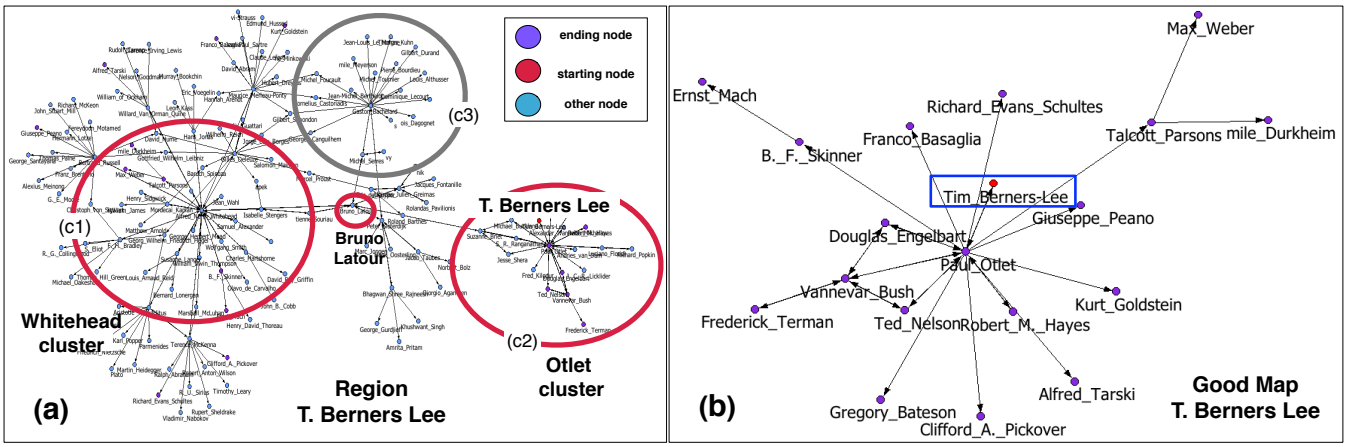


Figure 4: Region (a) and good map (b) for Example 8.

The region can be specified by considering the URI of TBL in DBpedia as starting node and the following $\mathcal{L}$ expression:

```
dbp:influenced<1-6>[test]
```

`test=` `ASK {?person rdf:type dbpedia:Scientist.}`

The notation `<1-6>` is a shorthand for the concatenation up to six steps of the predicate `dbp:influenced`, while the `test` is used to filter the endpoint nodes representing scientists. Fig. 4 (a) reports the region which contains 149 nodes and 236 edges. There are two main influence clusters; the first (c2) around Otlet and the second one (c1) around Whitehead. Note also that Latour is the bridge between the two clusters but (according to `dbpedia.org`) he is not a scientist. In the region there are several nodes that were visited during the evaluation of the expression but did not lead to any result (e.g., the cluster of nodes c3). Fig. 4 (b) shows the good map. It has 18 nodes and 43 edges and represents the connectivity information among scientists in the region.

4 Related Work

The development of our map framework share some characteristics with other approaches that focus on providing some form of "map" of the Web. Dodge (Dodge and Kitchin 2001) in the *Atlas of the Cyberspace*, provides a comprehensive overview of visual representations of *digital landscapes* on the Web. A recent information visualization paradigm used to summarize information is that of *metro maps* (e.g., (Shahaf, Guestrin, and Horvitz 2012)). Other strands of research related to ours are (visual) navigational histories, site maps and bookmarks. An approach enabling to create concept maps is described in Gaines and Shaw (Gaines and Shaw 1995). As for site maps, Li et al. (Li et al. 2001) investigated the problem of building multi-granular maps. Finally, bookmarking consists in marking and sharing (e.g., with Delicious) URIs for the purpose of future reuse.

The crucial difference with the present work is that these approaches are designed for human usage and are mainly oriented to visualization. Moreover, no formal/provable relations of connectivity between the URIs chosen are given.

Acknowledgments Fionda has been supported from the European Commission, European Social Fund and the Calabria region. Gutierrez was supported by the grant FONDECYT No 1110287.

References

DeRose, S. and Clark, J (eds). 1999. XML Path Language (XPath) W3C Recommendation.

Dodge, M., and Kitchin, R. 2001. *Atlas of Cyberspace*. Addison-Wesley Great Britian.

Fionda, V., Gutierrez, C., and Pirrò, G. 2012. Semantic Navigation on the Web of Data: Specification of Routes, Web Fragments and Actions. In *Proceedings of the 21st World Wide Web Conference (WWW)*, 281–290. ACM.

Gaines, B., and Shaw, M. 1995. WebMap: Concept Mapping on the Web. *World Wide Web Journal, 1(1), 171-183* 33.

Garfield, E. 1994. Scientography: Mapping the Tracks of Science. *Current Contents: Social & Behavioural Sciences* 7(45):5–10.

Heath, T., and Bizer, C. 2011. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool.

Ledbetter, A. M.; Griffin, E.; and Sparks, G. G. 2007. Forecasting "friends forever": A longitudinal investigation of sustained closeness between best friends. *Personal Relationships* 14(2):343–350.

Li, W.; Ayan, N.; Kolak, O.; Vu, Q.; Takano, H.; and Shimamura, H. 2001. Constructing Multi-Granular and Topic-Focused Web Site Maps. In *WWW*, 343–354. ACM.

Pérez, J.; Arenas, M.; and Gutierrez, C. 2010. nSPARQL: A Navigational Language for RDF. *Journal of Web Semantics* 8(4):255–270.

Robinson, A. H.; Morrison, J.; Muehrcke, O. C.; Kimerling, A.; and Guptill, S. C. 1995. *Elements of Cartography*. Wiley.

Rosvall, M., and Bergstrom, C. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences* 105(4):1118–1123.

Shahaf, D.; Guestrin, C.; and Horvitz, E. 2012. Trains of Thought: Generating Information Maps. In *Proceedings of the 21st World Wide Web Conference (WWW)*, 899–908. ACM.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

ASP Encodings of Acyclicity Properties

Martin Gebser[*†] and **Tomi Janhunen**[*] and **Jussi Rintanen**[*‡]
Helsinki Institute for Information Technology HIIT
Department of Information and Computer Science
Aalto University, FI-00076 AALTO, FINLAND

Abstract

Many knowledge representation tasks involve trees or similar structures as abstract datatypes. However, devising compact and efficient declarative representations of such properties is non-obvious and can be challenging indeed. In this paper, we take acyclicity properties into consideration and investigate logic-based approaches to encode them. We use answer set programming as the primary representation language but also consider mappings to related formalisms, such as propositional logic, difference logic, and linear programming.

Introduction

Numerous hard computational tasks involve the construction of acyclic or tree structures. Constraint satisfaction and related methods are an important approach for solving many of these problems. Since acyclicity and the property of being a tree are no primitives in common constraint-based representation formalisms, the challenge of formulating such conditions in terms of more basic constraint expressions arises. Hence, in this work, we systematically investigate logic-based approaches to encode respective properties.

Construction of acyclic graphs, trees, or chordal graphs shows up in numerous applications. For instance, Bayesian network structure learning, where directed acyclic graphs provide solution candidates, can be reduced to constraint optimization (Jaakkola et al. 2010; Cussens 2011). Furthermore, chordal Markov network learning amounts to the task of optimizing maximum weight spanning trees induced by chordal graphs (Corander et al. 2013). Chordality is a relaxation of strict acyclicity in which cycles of length three are allowed in an otherwise tree-structured undirected graph. Constraint-based methods can also be used to infer phylogenetic trees (Brooks et al. 2007; Bonet and John 2009), describing the evolution of living organisms, languages, and other evolving systems.

The basic problem to be solved with constraint-based formalisms in the above and other applications is constructing an acyclic or a tree structure subject to further conditions. That the choice among underlying edge candidates actually results in some graph structure with the desired properties must be enforced by corresponding constraints. Straightforward formulations, however, can become impractically large and inefficient, and our work sheds light on ways to encode such constraints succinctly and efficiently.

Although we rely on answer set programming (ASP) (Brewka, Eiter, and Truszczyński 2011) as the primary representation language for encodings, respective formulations in related formalisms, such as propositional satisfiability (SAT), SAT extended with difference logic (DL), and linear programming (LP), can be obtained through automatic polynomial translations from ASP. Linear translations to SAT exist whenever ASP rules are *tight* (Erdem and Lifschitz 2003), i.e., if there are no circular positive dependencies through rules' prerequisites on the ground level. In the non-tight case, level mappings (Janhunen 2004) can be used to bridge the semantic gap between ASP and SAT. Compact linear representations of level mappings can be achieved by means of difference constraints available in DL (Niemelä 2008) or integer variables in LP (Liu, Janhunen, and Niemelä 2012).

In this short paper, we present encoding approaches for directed acyclic graphs. Their extensions to forests and trees, in the directed and undirected case, as well as chordal graphs are addressed in an extended version of this paper.[1]

Directed Graphs

As usual, a *directed graph* G is a pair $\langle V, E\rangle$, where V is a finite set of *vertices* and $E \subseteq V \times V$ is a set of directed *edges*. For some $v \in V$, we denote the number of incoming or outgoing edges, respectively, by $\deg^-(v) = |\{u \mid \langle u, v\rangle \in E\}|$ and $\deg^+(v) = |\{u \mid \langle v, u\rangle \in E\}|$; v is a *root* (or *leaf*) of G if $\deg^-(v) = 0$ (or $\deg^+(v) = 0$). A *path* of length $k-1$ in G is a non-empty sequence $v_1, \ldots, v_k$ of vertices from V such that $\langle v_i, v_{i+1}\rangle \in E$ and $v_i \neq v_j$ for all $1 \leq i < j \leq k$. A sequence $v_0, v_1, \ldots, v_k$ is a *cycle* of length k in G if $\langle v_0, v_1\rangle \in E$, $v_0 = v_k$, and $v_1, \ldots, v_k$ is a path in G.

A *directed acyclic graph* is a directed graph G such that there is no cycle in G. A directed acyclic graph $G = \langle V, E\rangle$ is a *directed forest* if, for every $v_k \in V$, there is exactly one path $v_1, \ldots, v_k$ in G from a root v_1 of G to v_k. Given

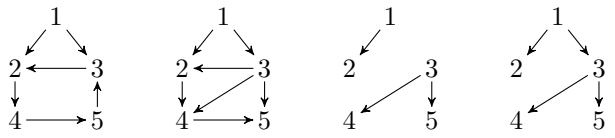
[*]The support from the Finnish Centre of Excellence in Computational Inference Research (COIN) funded by the Academy of Finland (under grant #251170) is gratefully acknowledged.

[†]Also affiliated with the University of Potsdam, Germany.

[‡]Also affiliated with Griffith University, Brisbane, Australia.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] http://research.ics.aalto.fi/software/asp/etc/asp-acyclic.pdf



(a) Cyclic (b) Acyclic (c) Forest (d) Tree

Figure 1: Directed example graphs with five vertices each

```
1 #const n=5.
2 node(1..n).
3 pair(X,Y) :- node(X;Y), X != Y.
4 { edge(X,Y) } :- pair(X,Y).
```

Figure 2: Encoding part for generating directed graphs

that any incoming edge $\langle v_{k-1}, v_k \rangle$ can be extended to a path $v_1, \ldots, v_{k-1}, v_k$ from a root v_1, the former condition is equivalent to requiring $\deg^-(v) \leq 1$ for all vertices $v \in V$. Finally, a *directed tree* is a directed forest $G = \langle V, E \rangle$ with a unique root, i.e., $\deg^-(v) = 0$ holds for exactly one $v \in V$. Some directed example graphs illustrating the introduced acyclicity properties are depicted in Figure 1.

Encodings

We illustrate different encodings by means of first-order specifications in the input language of the ASP grounder GRINGO (Gebser et al. 2012). The underlying principles, however, are of general applicability, and we outline particularities of respective SAT, DL, and LP formulations.

Rules describing the (non-deterministic) generation of directed graphs are shown in Figure 2. The predicate `node`/1 provides the labels $1, \ldots, \mathbf{n}$ for vertices, where **n** is an integer constant standing for the number of vertices, and the symmetric predicate `pair`/2 represents the domain of directed edges given by all pairs of distinct vertices. Any subset of these pairs can be generated via the choice rule in line 4, permitting instances of `edge`/2 to hold without further preconditions. For instance, the atoms characterizing the directed graph in Figure 1(a) are `edge(1,2)`, `edge(1,3)`, `edge(2,4)`, `edge(3,2)`, `edge(4,5)`, and `edge(5,3)`.

Acyclicity Checking

To make sure that a generated directed graph $\langle V, E \rangle$ is acyclic, we may check whether there is a strict partial order $<$ over vertices in V such that $u < v$ if $\langle u, v \rangle \in E$. The rules in Figure 3 encode this approach in an inductive fashion. Given **n**=5, the following (simplified) ground instances are obtained when the label `3` is substituted for `X`:

```
order(3,1) :- not edge(3,1).
order(3,1) :- order(1).
order(3,2) :- not edge(3,2).
order(3,2) :- order(2).
order(3,4) :- not edge(3,4).
order(3,4) :- order(4).
order(3,5) :- not edge(3,5).
order(3,5) :- order(5).
```

```
5 order(X,Y) :- pair(X,Y), not edge(X,Y).
6 order(X,Y) :- pair(X,Y), order(Y).
7 order(X) :- node(X),
              order(X,Y) : pair(X,Y).
9 :- node(X), not order(X).
```

Figure 3: Inductive bottom-up encoding of acyclicity test

```
order(3)   :- order(3,1), order(3,2),
              order(3,4), order(3,5).
:- not order(3).
```

As expressed by `order(3,`v`)`, vertex `3` fulfills the condition of $<$ relative to a potential successor $v \in \{1,2,4,5\}$ if there is no edge from `3` to v or if v has no path to `3`. Then, `order(3)` is derived once the existence of a cycle through vertex `3` can be safely excluded for all potential successors v of `3`. In particular, the positive prerequisites of corresponding ground rules for other vertices along with well-foundedness of derivations, as required in ASP, prohibit `order(3)` to hold if `3` belongs to some cycle. Given this, the integrity constraint denying models such that `order(3)` is false rejects any directed graph with a cycle through `3`, and respective ground rules establish the same for other vertices.

Reconsidering the example graph in Figure 1(a), we have that `order(`u`,`v`)` can be derived for distinct vertices u and v such that $\langle u, v \rangle$ is not an edge. Since each vertex u has some successor v, there still is some atom `order(`u`,`v`)` for u that cannot be concluded in this way, e.g., `order(5,3)` for vertex `5`. Hence, `order(`u`)` remains underivable for all $u \in \{1,2,3,4,5\}$, and the cyclic graph in Figure 1(a) is rejected by means of the rules in Figure 3. Unlike this, vertex `5` has no successor in the acyclic graph shown in Figure 1(b), so that `order(5,`v`)` is derivable for $v \in \{1,2,3,4\}$. This in turn yields `order(5)`, `order(4,5)`, and `order(4)`, given that `5` is the only successor of vertex `4`. Similarly, the derivation of `order(2,4)` leads to `order(2)`, and the establishment of `order(3,2)`, `order(3,4)`, and `order(3,5)` for the successors of vertex `3` allows for deriving `order(3)`. Finally, `order(1)` can be concluded in view of `order(1,2)` and `order(1,3)`. As derivable atoms are compatible with (ground instances of) the integrity constraint in line 9 of Figure 3, the graph in Figure 1(b) passes the acyclicity test.

Encoding Variants

The encoding in Figure 3 is *non-tight*, i.e., it relies on well-foundedness in the presence of circular positive dependencies on the ground level. Level mappings (cf. (Janhunen 2004; Niemelä 2008; Liu, Janhunen, and Niemelä 2012)) furnish an alternative mechanism to express well-founded derivations, where difference constraints or integer variables allow for linear embeddings in DL and LP. A respective approach is taken by the *tight* ASP formulation of the predicate `order`/1 in Figure 4. In order to eliminate circular positive dependencies between rules' prerequisites and conclusions, the auxiliary predicates `order`/3 and `order`/2 include an additional argument for "step counting" that unfolds the derivation of a partial vertex order $<$ witnessing acyclicity. If such an order exists, its construction must be completed in at most **n** steps. For in-

```
5 order(X,Y,1..n) :- pair(X,Y),
                     not edge(X,Y).
6 order(X,Y,N-1)  :- pair(X,Y),
                     order(Y,N).
7 order(X,N) :- node(X), N = 2..n,
                order(X,Y,N) : pair(X,Y).
8 order(X)   :- node(X),
                order(X,Y,1) : pair(X,Y).
```

Figure 4: Unfolded bottom-up derivation of `order`/1

stance, the (total) order $1 < 3 < 2 < 4 < 5$ over vertices of the acyclic graph in Figure 1(b) is represented in terms of the atoms `order(5,2...5)`, `order(4,2...4)`, `order(2,2...3)`, and `order(3,2)` along with `order(`u`)` for $u \in \{1, 2, 3, 4, 5\}$. On the one hand, explicit step counting eliminates the potential of circular derivations, so that straightforward translations like completion (Clark 1978) can be used to map the encoding in Figure 4 to SAT, DL, or LP. In fact, the tight ASP formulation of `order`/1 can be viewed as a directed counterpart of the SAT encoding for the undirected case in (Corander et al. 2013). On the other hand, the step argument introduces an additional dimension increasing the number of atoms as well as constraints. For directed graphs $\langle V, E\rangle$, the size of ground instances grows from $\mathcal{O}(|E|)$ for the encoding in Figure 3 to $\mathcal{O}(|E|\times|V|)$, thus shifting from linear to quadratic space.

Variants of the encodings presented so far can be obtained by replacing the *unconditional* generation of edges in line 4 of Figure 2 by the following (non-deterministic) choice rule:

```
{ edge(X,Y) } :- pair(X,Y), order(Y).
```

The additional prerequisite `order(Y)` necessitates the absence of cycles through a vertex substituted for `Y` before admitting any edge from a predecessor substituted for `X`. Hence, the generation of edges with the above choice rule progresses successively from the leaves of a directed acyclic graph. Whether such *conditional* edge generation is advantageous or not is empirically investigated in the evaluation section below. While the *leaf encodings* in Figure 3 and 4 describe "bottom-up" traversals starting from the leaves of a directed graph, acyclicity tests can likewise be performed "top-down" from roots, and a successive choice rule similar to the one above can optionally be used in *root encodings* instead of line 4 in Figure 2. Without other side constraints, bottom-up and top-down traversals appear fully symmetric, but in extended contexts the orientation of acyclicity tests may interact or interfere.

Notably, the predicate definitions in Figure 3 and 4 are both *stratified* (cf. (Apt, Blair, and Walker 1988)). That is, when instances of the predicate `edge`/2 are fixed, there is at most one (well-founded) model that can be deterministically determined. In the absence of circular positive dependencies, as with ground instances of the rules in Figure 4, the unique model or unsatisfiability is also obtained by evaluating (via unit propagation) the rules' completion. Except for variants introducing prerequisites for conditional edge generation, all encoding parts in the sequel are stratified, so that the choice of edges is the only source of non-determinism.

directed acyclic graphs	**leaf encoding** non-tight	tight	**root encoding** non-tight	tight
CLASP	0.26	—	0.27	—
CLASP/SAT	2.41	1.59	1.55	1.16
LINGELING	5.65	2.53	5.19	2.49
Z3	1.69	—	1.67	—
CPLEX	682.54	623.17	639.38	687.46

Table 1: Average runtimes for directed acyclic graphs

Evaluation

To test our encodings, we ran CLASP (2.1.3) both as an ASP and SAT solver, the SAT solver LINGELING (ats-57807c8-131016), the DL solver Z3 (4.3.1), and the LP solver CPLEX (12.5.0.0) with default settings on a cluster of Linux machines. We used GRINGO (3.0.5) to ground first-order ASP formulations and the translators LP2SAT2 (1.18) (Janhunen and Niemelä 2011), LP2DIFF (1.33) (Janhunen, Niemelä, and Sevalnev 2009), and LP2MIP (1.18) (Liu, Janhunen, and Niemelä 2012) for converting ground instances to SAT, DL, or LP, respectively. In order to perturb the search space for directed acyclic graphs, we added randomly generated XOR-constraints over edges, systematically varying both the length and the number of such constraints.

Average runtimes in seconds for 25 instances with $\mathbf{n}$=25 vertices are given in Table 1. Runs exceeding 1000 seconds were counted as 1000 seconds, and an entry "—" marks that no run completed in time. We compare the performance of the considered solvers on four different encodings of directed acyclic graphs. The non-tight and tight variants of the *leaf encoding* consist of the rules shown in Figure 2 along with the inductive or unfolded bottom-up acyclicity test, respectively, encoded in Figure 3 and 4. Their counterparts denoted by *root encoding* swap the orientation of acyclicity tests by traversing vertices top-down from roots instead of bottom-up from leaves. The runtimes of CLASP on ASP formulations in the first row exhibit no significant differences between the symmetric leaf and root encodings. That is, instances relying on their non-tight variants were solved easily by CLASP (with default settings), whereas all runs on the tight variants timed out. Similar behavior is observed with the Z3 solver, for which non-tight instances are mapped to DL and tight ones to plain SAT. However, LINGELING and CLASP, run as a SAT solver, both perform well on the translations by LP2SAT2, where the tight ASP formulations have some advantage over the level mappings introduced when translating non-tight instances. The performance of the LP solver CPLEX (with default settings) is rather unstable on all of our encoding variants. In particular, CPLEX does not substantially benefit from the use of integer variables to represent level mappings for non-tight instances.

Conclusions

Graphs satisfying acyclicity properties are frequent in knowledge representation tasks. Since these properties do not appear as basic primitives in common constraint-based representation formalisms, formulating them compactly and efficiently is a recurring challenge. We have investigated

logic-based characterizations of directed acyclic graphs and developed encodings in the language of answer set programming. Directed forests and trees, their undirected variants, as well as chordal graphs are addressed in an extended draft.[1]

Experiments indicate the relevance of representation approaches and constraint formulations, with particular emphasis on well-foundedness. While the size of ground instances is linear for the acyclicity test in Figure 3, encodings based on the full transitive closure of edges (Rintanen, Heljanko, and Niemelä 2004; Brooks et al. 2007; Çaylı et al. 2007; Cussens 2008; Çelik et al. 2009; Brewka, Eiter, and Truszczyński 2011) yield quadratic space complexity, even for sparse graphs. As the discovery of a single "universal" encoding is highly unlikely, the alternatives investigated here provide a toolkit for representing graph structures in different applications. In fact, we believe that effective declarative means to specify such fundamental datatypes are important cornerstones for the development of robust constraint-based solving methods.

References

Apt, K.; Blair, H.; and Walker, A. 1988. Towards a theory of declarative knowledge. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming*, 89–148. Morgan Kaufmann Publishers.

Bonet, M., and John, K. 2009. Efficiently calculating evolutionary tree measures using SAT. In Kullmann, O., ed., *Proceedings of the Twelfth International Conference on Theory and Applications of Satisfiability Testing (SAT'09)*, volume 5584 of *Lecture Notes in Computer Science*, 4–17. Springer.

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. Answer set programming at a glance. *Communications of the ACM* 54(12):92–103.

Brooks, D.; Erdem, E.; Erdogan, S.; Minett, J.; and Ringe, D. 2007. Inferring phylogenetic trees using answer set programming. *Journal of Automated Reasoning* 39(4):471–511.

Çaylı, M.; Karatop, A.; Kavlak, E.; Kaynar, H.; Türe, F.; and Erdem, E. 2007. Solving challenging grid puzzles with answer set programming. In Costantini, S., and Watson, R., eds., *Proceedings of the Fourth International Workshop on Answer Set Programming (ASP'07)*, 175–190.

Çelik, M.; Erdoğan, H.; Tahaoğlu, F.; Uras, T.; and Erdem, E. 2009. Comparing ASP and CP on four grid puzzles. In Gavanelli, M., and Mancini, T., eds., *Proceedings of the Sixteenth RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA'09)*. CEUR Workshop Proceedings.

Clark, K. 1978. Negation as failure. In Gallaire, H., and Minker, J., eds., *Logic and Data Bases*, 293–322. Plenum Press.

Corander, J.; Janhunen, T.; Rintanen, J.; Nyman, H.; and Pensar, J. 2013. Learning chordal Markov networks by constraint satisfaction. In Burges, C.; Bottou, L.; Ghahramani, Z.; and Weinberger, K., eds., *Proceedings of the Twenty-seventh Annual Conference on Neural Information Processing Systems (NIPS'13)*, volume 26 of *Advances in Neural Information Processing Systems*, 1349–1357.

Cussens, J. 2008. Bayesian network learning by compiling to weighted MAX-SAT. In McAllester, D., and Myllymäki, P., eds., *Proceedings of the Twenty-fourth International Conference on Uncertainty in Artificial Intelligence (UAI'08)*, 105–112. AUAI Press.

Cussens, J. 2011. Bayesian network learning with cutting planes. In Cozman, F., and Pfeffer, A., eds., *Proceedings of the Twenty-seventh International Conference on Uncertainty in Artificial Intelligence (UAI'11)*, 153–160. AUAI Press.

Erdem, E., and Lifschitz, V. 2003. Tight logic programs. *Theory and Practice of Logic Programming* 3(4-5):499–518.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers.

Jaakkola, T.; Sontag, D.; Globerson, A.; and Meila, M. 2010. Learning Bayesian network structure using LP relaxations. In Teh, Y., and Titterington, D., eds., *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS'10)*, volume 9 of *JMLR Proceedings*, 358–365. JMLR.org.

Janhunen, T., and Niemelä, I. 2011. Compact translations of non-disjunctive answer set programs to propositional clauses. In Balduccini, M., and Son, T., eds., *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday*, volume 6565 of *Lecture Notes in Computer Science*, 111–130. Springer.

Janhunen, T.; Niemelä, I.; and Sevalnev, M. 2009. Computing stable models via reductions to difference logic. In Erdem, E.; Lin, F.; and Schaub, T., eds., *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*, volume 5753 of *Lecture Notes in Artificial Intelligence*, 142–154. Springer.

Janhunen, T. 2004. Representing normal programs with clauses. In López de Mántaras, R., and Saitta, L., eds., *Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECAI'04)*, 358–362. IOS Press.

Liu, G.; Janhunen, T.; and Niemelä, I. 2012. Answer set programming via mixed integer programming. In Brewka, G.; Eiter, T.; and McIlraith, S., eds., *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, 32–42. AAAI Press.

Niemelä, I. 2008. Stable models and difference logic. *Annals of Mathematics and Artificial Intelligence* 53(1-4):313–329.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2004. Parallel encodings of classical planning as satisfiability. In Alferes, J., and Leite, J., eds., *Proceedings of the Ninth European Conference on Logics in Artificial Intelligence (JELIA'04)*, volume 3229 of *Lecture Notes in Computer Science*, 307–319. Springer.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Tracking Beliefs and Intentions in the Werewolf Game

Codruta Girlea
University of Illinois
Urbana, IL 61801, USA
girlea2@illinois.edu

Eyal Amir
University of Illinois
Urbana, IL 61801, USA
eyal@illinois.edu

Roxana Girju
University of Illinois
Urbana, IL 61801, USA
girju@illinois.edu

Abstract

We propose a model of belief and intention change over the course of a dialogue, in the case where the decisions taken during the dialogue affect the possibly conflicting goals of the agents involved. We use Situation Calculus to model the evolution of the world and an observation model to analyze the evolution of intentions and beliefs. In our formalization, utterances, that only change the beliefs and intentions, are observations. We illustrate our formalization with the game of Werewolf.

Introduction

Agents are engaged in dialogues according to goals they can satisfy by talking to other agents (Cohen and Perrault 1979),(Perrault and Allen 1980). A dialogue is a sequence of utterances, produced by agents based on their beliefs, in order to reach intended states of facts by causing other agents to change their beliefs and thus also their intentions. Our goal is to model this interaction between beliefs, intentions, and utterances. The ability to predict decisions resulting from the dialogue is used as a performance measure.

An example of such a domain is the *Werewolf* game. Here, players are assigned the roles of either villagers or werewolves. The game proceeds in alternating day and night stages, overlooked by an impartial judge. By night, unseen by villagers, the werewolves choose and kill a victim. The victim's identity is announced by the judge at the beginning of the next day. Then, the rest of the villagers discuss and vote to execute one person who they agree is a werewolf. The problem is to predict the outcome of each player's vote.

The game can be modeled with utterances as actions in Situation Calculus (McCarthy 1983), based on Austin's theory of performatives (Austin 1975). A problem is accounting for the indefinitely many perlocutionary acts, or possible effects an utterance may have on the hearer. Belief Revision (Alchourron, Gardenfors, and Makinson 1985) enables modeling and reasoning about changes of beliefs, including as a result of knowledge producing actions in Situation Calculus (Shapiro et al. 2000). Applying Belief Revision to our domain is hard because the agents need to do reasoning about beliefs over beliefs.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

We present a new approach to Belief Revision in Situation Calculus. We overcome the need to represent perlocutions by assuming an 'observation model', describing what beliefs, intentions, and unknown properties the utterances expose. The agents' belief states are then filtered (Shirazi and Amir 2011) with the observed utterances, resulting in an updated Kripke structure.

Our new approach allows us to describe dialogues combined with actions. From a Kriple Structure perspective, we add an observation model of utterances, and a revision model, accounting for how the utterances are used to change beliefs. From a Situation Calculus perspective, we model beliefs over beliefs, belief change, and perlocutionary effects. From a Belief Revision perspective, we represent how utterances affect dialogue participants on an individual level, depeding on each agents' internal state.

Our model does not use any linguistic input. We assume the utterances are already parsed to logical formulas that encode both their propositional content and the speech act they function as. Utterances expressed in natural language may provide further insight into the agents' beliefs, encoded in presuppositions and propositional attitudes. We plan to embed those phenomena in our model as future work.

Related Work

Scherl and Levesque (Scherl and Levesque 2003) introduce a situation accessibility relation predicate for the knowledge modality, and a sensing result function for knowledge producing actions, in a single agent scenario. The knowledge of the agent changes only through sensing, and there is no discussion of actions that specifically change the possibly nested beliefs, leaving the world unchanged.

This work has been extended (Shapiro et al. 2000),(Delgrande and Levesque 2012) for Belief Revision and Update. The authors introduce plausabilty as a mapping from possible worlds to natural numbers, defining a total order on possible worlds in which one believes the most plausible possible world. Belief revision changes plausability.

R. Demolombe and P. P. Parra (Demolombe and del Pilar Pozos Parra 2000) define an epistemic logic for the multiagent case, using belief as a simple modality that qualifies sentences but does not allow nesting. Their fragment is tractable, but the fact that it does not allow nested belief makes it inapplicable to domains where agent's actions are

oriented mainly towards changing other agents' beliefs, both about the world and about themselves and other agents.

It is on this (Demolombe and del Pilar Pozos Parra 2000) tractable fragment that most of the work (Demolombe, Mara, and Fern 2005),(Parra, Nayak, and Demolombe 2004) on formalizing intentions or the BDI framework in Situation Calculus is built upon. These formalizations therefore inherit the limited expressing power of the fragment, also not allowing nested beliefs. Here, intentions are about actions, referring to plans, as sequences of actions that are intended, in that specific order, in order to reach the goals. We do not consider planning, our intentions refer to the states of the world currently most desirable to the agent.

None of these approaches can straightforwardly be used in our domain, by enconding utterances as actions, as we need to account for both the change in nested beliefs, and the variability in how an utterance affects the hearer. Early approaches to modeling speech acts as plan operators (Cohen and Perrault 1979) propose a solution to the latter problem by defining mediating acts between the speech acts and the perlocutionary effects. These acts describe the conditions in which the direct effects of the illocutionary act result in the intended effects. This amounts to defining perlocutionary effects as actions where the preconditions are defined on the hearer belief state, and illocutionary effects as actions where the preconditions are defined on the speaker belief state, the latter's effects being carried over to the mediating act's preconditions. A disadvantage is that the set of perlocutionary effects is virtually unbounded for any utterance.

Early work on dialogue modeling (Cohen and Levesque 1985), (Cohen and Levesque 1990) assumes that the dialogue participants are rational, intentional agents that hold beliefs and choose to intend actions according to their goals. This work builds upon earlier theories in pragmatics by Searle (Searle 1976) and Horn (Horn 1984). The authors axiomatize some speech acts as actions in an event calculus, with modal operators for beliefs and goals, and event operators such as *DONE* and *AFTER*. Intention in this framework refers to the intention to execute an action according to a plan in order to reach a goal, so an agent may intend an action, not a set of possible worlds described by a formula.

Formalization of the Observation Model

The observation axioms describe what each observation exposes. The propositional content and function of an utterance is modeled as an observation, rather than an action. The intuition is that utterances do not change the world itself, but change and provide evidence for agent beliefs.

Observation axioms play a similar role to the observation model in statistical models such as the HMM or the POMDP (Kaelbling, Littman, and Cassandra 1998). They also expose a hidden state, using the observed value of a model variable. In our case, the hidden state consists of beliefs, intentions, and unknown properties (roles), and the model variable is a predicate encoding an observable fact.

Preliminary Definitions and Notation

We build upon the framework of Situation Calculus (McCarthy 1983), using successor state axioms as introduced by Reiter (Reiter 1991) as a solution to the frame problem.

We have a set Ag of agents, a set R of roles, and a set S of situations. Some roles $R_u \subseteq R$ are unique. The set of unique roles for a certain game is known by all agents.

The signature specifies sorts α (agent) and σ (situation), a set M of modalities, a set F of fluents, a set C of constant symbols, and a set A of actions, where: $M = \{K, B, P, I\}$ (knowledge, belief, intention, persistent intention), $F = \{r : \alpha\sigma\}_{r\in R} \cup \{dead : \alpha\sigma, alive : \alpha\sigma, voted : \alpha\alpha\sigma\} \cup \{claim_r : \alpha\alpha\sigma\}_{r\in R} \cup \{claim_{\neg r} : \alpha\alpha\sigma\}_{r\in R}, C = \{c : \alpha\}_{c\in Ag} \cup \{s_0 : \sigma\}, A = \{vote(x)\} \cup \{round_n(x)\}_{3\leq n\leq |C|}$.

Modalities Modalities are ternary relations understood as each agent's accessibility relation on situations (Scherl and Levesque 2003). More specifically, for a modality $M \in \{K, B, P, I\}$, $M(x, s', s)$ is used for situation s' being accessible from situation s to agent x according to M.

For modality M, agent x, situation s, and formula ϕ_z with free situation variable z, we use the shorthand notation:

$M(x, \phi_z, s) \equiv \forall s' : \sigma.M(x, s', s) \rightarrow \phi_z[z/s']$

To improve readability, we will denote free situation variables with a single mention by symbol _.

Observation Axioms

Utterances are treated as observations, and are produced in the conditions specified by our axiom system.

An observation axiom is an axiom of the form:

$$\forall \vec{x} : \vec{\alpha}.l(\vec{x}, s) \wedge \wedge_{i\neq j\in I} x_i \neq x_j \rightarrow \vee_k(\phi_k \wedge \phi_k^M)$$

where: l is a literal, $\vec{x}$ is a vector of agent variables, I is a subset of the indices of $\vec{x}$, each ϕ_k is a formula with no modalities, and in each ϕ_k^M, every formula with no modalities appears in the scope of a modal formula.

Literal $l(\vec{x}, s)$ is the *observation* and encodes any fact that becomes available to all agents. Formulas ϕ_k encode facts that are not available to all agents. Formulas ϕ_k^M describe the intentions or beliefs that, in the conditions described by ϕ_k, resulted in the observed value of the predicate encoded by l. Formulas ϕ_k and ϕ_k^M describe the hidden state.

We assume only players can make and be the object of claims, for all roles $r \in R$, claims $c \in \{r, \neg r\}$, situations s:

$\forall x, y : \alpha.claim_c(x, y, s) \rightarrow \neg(judge(x, s) \vee judge(y, s))$

As a simplification, we consider as observations claims about the agents' roles. Other utterances only expose their propositional content. The observation model is known by all agents and consists of the following axioms:

- Accusing another player of being the werewolf:

$$\begin{aligned}&\forall x, y : \alpha.(claim_{wolf}(x, y, s) \wedge x \neq y) \rightarrow (wolf(x, s) \wedge \\ &\neg B(x, wolf(y, \_), s) \wedge B(x, B(y, wolf(x, \_))) \vee \\ &\neg wolf(x, s) \wedge B(x, wolf(y, \_), s)) \end{aligned} \quad (1)$$

- Defending another player or oneself:

$$\begin{aligned}&\forall x, y : \alpha.claim_{\neg wolf}(x, y, s) \rightarrow (wolf(x, s) \\ &\wedge B(x, \exists z : \alpha.z \neq x \wedge z \neq y \wedge B(z, wolf(y, \_), \_), s) \\ &\wedge B(x, \exists z : \alpha.z \neq x \wedge z \neq y \wedge \neg B(z, wolf(y, \_), \_), s) \\ &\wedge (x \neq y \vee \neg B(x, wolf(y, \_), s))) \vee (\neg wolf(x, s) \\ &\wedge B(x, \exists z : \alpha.z \neq x \wedge z \neq y \wedge B(z, wolf(y, \_), s), s) \\ &\wedge \neg B(x, wolf(y, \_), s))) \end{aligned} \quad (2)$$

- Claiming the role of seer:
$\forall x : \alpha.claim_{seer}(x, x, s) \rightarrow (wolf(x, s) \vee seer(x, s))$
$\forall x, y : \alpha.(claim_{seer}(x, y, s) \wedge x \neq y) \rightarrow (\neg wolf(x, s)$
$\wedge B(x, seer(y, \_), s) \wedge \neg B(x, wolf(y, \_), s)$
$\wedge B(x, \exists z : \alpha.I(z, dead(y, \_), \_), s))$ (3)

- Claiming that someone is not a seer:
$$\forall x, y : \alpha.claim_{\neg seer}(x, y, s) \rightarrow seer(x, s) \wedge B(x, \exists z : \alpha.B(z, seer(y, \_), s), s) \quad (4)$$

- Not only utterances can be seen as observations that alter beliefs and intentions. Executing an action such as voting can affect fluents such as $voted$, so those fluents can also be modeled as observations:
$$voted(x, y, do(z, a, s)) \rightarrow I(x, dead(y, \_), s) \quad (5)$$

Belief and Intention Axioms

A first set of axioms describe the end goal of each agent, which in the game of Werewolf is to stay alive.

$P(x, \neg dead(x, \_), s)$

Game rounds end with agents voting to have someone executed, which also reflects in the agents' intentions:

$P(x, \vee_{y \in C, y \neq x} dead(y, \_), s)$

Persistent intentions are intentions and are known by all.

In a reciprocal manner, any agent will intend another agent be dead if she concluded the latter intends her death:

$K(x, I(y, dead(x, \_), s), s) \rightarrow I(x, dead(y, \_), s)$

If an agent knows who the werewolf is, unless she is herself a werewolf, she will intend the werewolf be dead:

$K(x, wolf(y, \_), s) \wedge \neg wolf(y, s) \rightarrow I(x, dead(y, \_), s)$

The werewolf will have a similar approach to the seer.

Conversely, knowing an agent's intention will also give some insight into his belief regarding who the werewolf is:

$\forall x, y : \alpha.I(x, dead(y, \_), s) \rightarrow (\neg wolf(x, s) \wedge B(x, wolf(y, \_), s)) \vee (wolf(x, s) \wedge B(x, \neg wolf(y, \_), s))$

Initial Situation s_0

Initially, the agents are assigned and thus know their roles: $\forall x : \alpha.r(x, s_0) \rightarrow K(x, r(x, \_), s_0)$ for all roles r.

Since each agent has at most one role, for the unique roles $r \in R_u$ we have:
$$\forall x, y : \alpha.r(x, s) \rightarrow (y = x \vee (K(x, \neg r(y, \_), s_0)) \wedge K(x, \neg B(y, r(y, \_), \_), s_0))) \quad (6)$$

Each game will specify a role assignment. The roles will be assigned in the initial situation, but this assignment will not be available to the agents.

Belief Filtering

Observations are used in the process of logical filtering (Shirazi and Amir 2011) to select the situations to be believed or intended. We keep the original logical filtering notation and definition for formulas not containing modalities.

The result of filtering the belief relation with an observation is a new accessibility relation, where situations inconsistent with the observation are no longer accessible.

Definition 1 *Let M be a modality predicate and o an observation. Then, for any situations s, s' and agent $x \in Ag$:*

$Filter[o][M(x, s', s)] \equiv P(x, s', s)$, *if* $M \in \{P, I\}$

$Filter[o][M(x, s', s)] \equiv o_{[s/s']} \wedge M(x, s', s)$, *otherwise*

An agent will believe the consequences of her previous beliefs and the new observation:

Theorem 1 *For any modality $M \in \{K, B\}$, agent Ag, situation s, formulas ϕ, ψ, and observation o with $\{\psi, o\} \models \phi$:*

$$\{M(x, \psi, s)\} \models Filter[o][M(x, \phi, s)]$$

Conversely, only those consequences of previous beliefs that are consistent with the new observation are believed:

Theorem 2 *For any modality $M \in \{K, B\}$, agent x, situation s, formulas ϕ, ψ, and observation o with $\{\psi, \phi\} \models \neg o$:*

$$\{M(x, \psi, s), Filter[o][M(x, \phi, s)]\} \models \Box$$

This does not hold for intentions: as a result of refining her beliefs, an agent can drastically change her intentions.

Formalization of the Dynamic Model

We model actions that affect the world in Situation Calculus. The actions are voting and ending a round of game.

Voting is available to any player and any living player can vote for any living player she intends the death of:
$$Poss(x, vote(y), s) \equiv \neg judge(x, s) \wedge \neg judge(y, s) \wedge I(x, dead(y, \_), s) \quad (7)$$

Ending a game round of n players with x voted out ($round_n(x)$) is available to the judge:

$Poss(u, round_n(x), s) \equiv$
$\exists y_1 ... y_n : \alpha.(\wedge_{1 \leq i \leq n}(alive(y_i, \_) \wedge \neg judge(y_i, \_)) \wedge$
$\wedge_{1 \leq i \neq j \leq n} y_i \neq y_j \wedge (\forall z : \alpha.alive(z, \_) \wedge \neg judge(z, \_) \rightarrow$
$\vee_{1 \leq i \leq n} z = y_i)) \wedge judge(u, s) \wedge \neg dead(x, s) \wedge$
$\exists y_1 ... y_{1+n/2} : \alpha. \wedge_{1 \leq i \leq 1+n/2} voted(y_i, x, s)$

Ending a game round is also an *announcement* action, where the judge announces the identity of the player who was killed by werewolves during the night. Similarly to a sensing action (Scherl and Levesque 2003), it has associated an *announcement result* AR. This is a partial function as, depending on the roles, there may be rounds with no victims.

$AR(round_n(x), s, r) \equiv \vee_{c \in Ag}(r = c \wedge dead(c, s))$
$AR(round_n(x), s, r) \wedge AR(round_n(x), s, r') \rightarrow r = r'$

Successor State Axioms

The successor state axioms describe the effects of executing an action on any of the fluents:

- The roles do not change:
$r(x, do(y, a, s)) \equiv r(x, s) \wedge \neg dead(x, do(y, a, s))$
- Players are dead because they were either already dead, voted out at the end of the current round, or killed by werewolves at the end of the current round:
$dead(x, do(z, a, s)) \equiv dead(x, s) \vee (a = round_n(x) \wedge Poss(z, a, s)) \vee \exists y : \alpha.(a = round_n(y) \wedge Poss(z, a, s) \wedge AR(a, do(z, a, s), x))$

- A player's vote is counted when she performs the action of voting and does not change unless she votes for another player:

$$voted(x, y, do(u, a, s)) \equiv$$
$$(voted(x, y, s) \wedge (u \neq x \vee \forall z : \alpha.a \neq vote(z)))$$
$$\vee (a = vote(y) \wedge u = x \wedge Poss(x, a, s))$$

Note that belief filtering and action execution are orthogonal, in that the first only affects accessibility relations, whereas the latter only affects the current situation. Action execution results in a new situation, and new observations will lead to refining the accessibility of situations believed or intended from the new situation.

Conclusions

We presented a model for beliefs in dialogues where agents have conflicting intentions, and the Werewolf game in particular. Other domains can be formalized in a similar manner, by writing the appropriate observation model and belief and intention interaction axioms.

A possible line of future work is using the formalism for a richer domain, such as trials. Another extension is to probabilistic observation and dynamic models, and probabilistic belief and intention accessibility relations. This would allow better predictions of the voting outcomes, as an agent will no longer choose an action non-deterministically, but will have a preference over intended outcomes.

This formalization could be useful in computational pragmatics, for modeling complex conversations where some reasoning beyond text level is needed to follow and predict what the participants may say. In turn, linguistic cues exposing attitudes and polarity could result in a more accurate model of human interaction. Furthermore, in our example, we assumed the dialogue already parsed in a sequence of observations and relations. In the future, we would need to extract these logical formulas from text.

Acknowledgments

This work was supported by NSF IIS grants 09-17123, 09-68552, grant NSF EAR 09-43627 EA, and a Defense Advanced Research Project Agency (DARPA) grant as part of the Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Alchourron, C. E.; Gardenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic* 50:510–530.

Austin, J. L. 1975. *How to do things with words*. Cambridge, Mass.: Harvard University Press.

Cohen, P. R., and Levesque, H. J. 1985. Speech acts and rationality. In Mann, W. C., ed., *ACL*, 49–60. ACL.

Cohen, P. R., and Levesque, H. J. 1990. Performatives in a rationally based speech act theory. In Berwick, R. C., ed., *ACL*, 79–88. ACL.

Cohen, P. R., and Perrault, C. R. 1979. Elements of a plan-based theory of speech acts. *Cognitive Science* 3(3):177–212.

Delgrande, J. P., and Levesque, H. J. 2012. Belief revision with sensing and fallible actions. In Brewka, G.; Eiter, T.; and McIlraith, S. A., eds., *KR*. AAAI Press.

Demolombe, R., and del Pilar Pozos Parra, M. 2000. A simple and tractable extension of situation calculus to epistemic logic. In Ras, Z. W., and Ohsuga, S., eds., *ISMIS*, volume 1932 of *Lecture Notes in Computer Science*, 515–524. Springer.

Demolombe, R.; Mara, A.; and Fern, O. 2005. Intention recognition in the situation calculus and probability theory frameworks. In *In Computational Logic in Multi Agent Systems*, 358–372.

Horn, L. 1984. Toward a new taxonomy for pragmatic inference: Q-based and r-based implicature. *1984* 1142. Cited by 615.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artif. Intell.* 101(1-2):99–134.

McCarthy, J. 1983. Situations, actions, and causal laws. Technical Report Memo 2, Stanford Artificial Intelligence Project, Stanford University.

Parra, P. P.; Nayak, A.; and Demolombe, R. 2004. Theories of intentions in the framework of situation calculus. In *Declarative Agent Languages and Technologies (DALT 2004), LNCS 3476, Springer-Verlag (2005). In this volume*. Springer Verlag.

Perrault, C. R., and Allen, J. F. 1980. A plan-based analysis of indirect speech acts. *Comput. Linguist.* 6(3-4):167–182.

Reiter, R. 1991. Artificial intelligence and mathematical theory of computation. San Diego, CA, USA: Academic Press Professional, Inc. chapter The Frame Problem in Situation the Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression, 359–380.

Scherl, R., and Levesque, H. J. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence* 144(1–2):1–39.

Searle, J. R. 1976. A classification of illocutionary acts. *Language in Society* 5:1–23.

Shapiro, S.; Pagnucco, M.; Lesprance, Y.; and Levesque, H. J. 2000. Iterated belief change in the situation calculus. In *Principles of Knowledge Rep. and Reasoning: Proc. of the 7th Int. Conf*, 527–538.

Shirazi, A., and Amir, E. 2011. First-order logical filtering. *Artif. Intell.* 175(1):193–219.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Representing and Reasoning about Time Travel Narratives: Foundational Concepts

Leora Morgenstern
Leidos Corporation
Arlington, VA 22203
leora.morgenstern@leidos.com

Abstract

The paper develops a branching-time ontology that maintains the classical restriction of forward movement through a temporal tree structure, but permits the representation of paths in which one can perform inferences about time-travel scenarios. Central to the ontology is the notion of an agent embodiment whose beliefs are equivalent to those of an agent who has time-traveled from the future.

1 Introduction

This paper explores the development of a formal theory of time travel, in which one would be able to represent and reason with time travel narratives. This is part of a long-term research program in developing formal models of narrative (Morgenstern 2008). A longer version of this paper is available at (Morgenstern 2014).

2 Working Example

Our working example is adapted and simplified from the Tapestry episode of *Star Trek: The Next Generation*. When Jean-Luc Picard was young, he was involved in a brawl with the Nausicaans, in which he defended the honor of a friend. Picard's heart was irreparably injured, and he was given an artificial, damage-prone heart instead. When he is a middle-aged captain of the Enterprise, he is injured. The artifical heart malfunctions, and Picard dies. While he is waiting to enter the afterlife, the superbeing Q offers him the chance to change his life. Picard returns to the past and avoids the brawl. However, Picard has now become overly cautious and thus never amounts to anything. When Picard realizes the consequences of avloiding the brawl, he asks Q to revert to his old life: he would rather live a meaningful, even if shortened, life.

3 Temporal Ontology

3.1 Representing Backward Time

AI temporal ontologies are generally founded on linear time, as in the event calculus (Miller and Shanahan 1994) or branching time, as in the situation calculus (Reiter 2001). Because linear time does not facilitate reasoning about alternate possibilities, it seems unsuitable as the underlying ontology for time travel. Branching time seems to have better potential. but we are left with a basic conceptual problem: time goes forward, not back.

To solve this problem, we first move away from the implicit assumption in time-travel accounts that there is a path in the time tree that "really" or "first" occurs; Rather, at any point in the time tree, certain sets of paths are accessible by certain agents. Some of these subtrees intuitively correspond to how the world might be if time travel were allowed.

Second, we introduce the idea of different *embodiments* of agents. Intuitively, an agent changes as he goes through different experiences, most notably by gaining beliefs. Specifically, we talk about the embodiment of an agent $A(Sj, Si)$, where there is a path segment from Si to Sj; this represents someone who has the memories of an agent who has lived from Si to Sj. $A(Sj, Si)$ corresponds to the agent A who has time-traveled back from Sj to Si.

Third, we introduce sets of time-travel (or *departure*) points and re-entry points for particular agents. In general, the path between a re-entry point and a departure point is called a *reversible path segment*. An embodiment of an agent will be characterized by the pair (departurel point, re-entry point), indicating that the agent in this embodiment will have beliefs about what has happened in the reversible path segment.

3.2 Preformal development of the model

Consider the working example (Figure 1).

At S1, Picard can choose to get involved in the brawl (path from S1 to S3), or to withdraw from the brawl (path from S1 to S2). Picard gets involved in the brawl and is stabbed. Picard continues along the path whose initial segment is (S1,S3). Assume that he is injured again between S4 and S7 and is dead at S7. S7 is a time-travel point: Picard has the choice to "time-travel." The re-entry point available to him is S1, the point at which he decided to get involved in the brawl.

We represent time travel as a triple consisting of the departure and the re-entry point, together with the agent who is intuitively doing the time-travel along portions of the subtree rooted at S1, These portions of the subtree are accessible to a more experienced *embodiment* of Picard, Picard(S7,S1), who

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

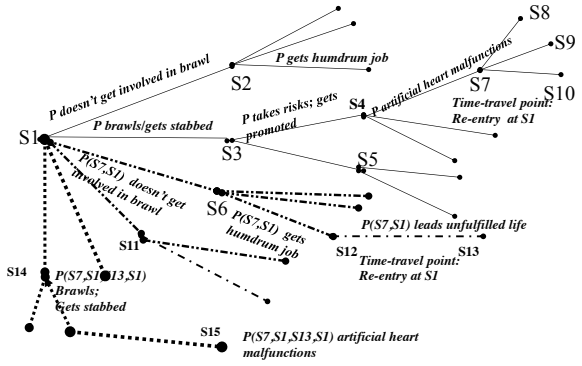


Figure 1: *Star Trek TNG Tapestry Example.*

chooses now to follow a different path. He can no longer follow the path from S1 to S2 but he follows a similar path in which he avoids the brawl (S1,S6).

Nested Time Travel and Embodiments: The time-travel process can be repeated along the path segment of an embodied agent, leading to nested embodiments. E.g., the embodiment of Picard who is aware that brawling will shorten his life realizes, when middle-aged, that he never amounted to anything because he *avoided* the brawl. This embodiment then has a possibility of time travel at S13 and an associated re-entry point at S1. (See Fig. 1.)

Embodiment along a path stays the same or grows deeper, but does not grow shallower. An agent is marked by the path segments that he believes he has lived.

4 Formal Model

4.1 The Time-Travel Tree Structure

Definition 4.1.1 A time-travel tree structure *TTT* is a tuple (S,Act,DP,RP,AG, τ, T), whose elements are described below: [1]

S: S is an infinite set of situations, arranged into a partial order under the precedes relation $<$. The function *time* maps a situation onto its date-clock-time. If $s1 < s2$, then $time(s1) < time(s2)$. There is a path between any two ordered situations. $start$ and end are functions giving the start and end situations of any finite path segment.

Act: A set of actions of the form $do(ag, ac)$ where ag is an agent (see below) and ac is an *actional*, intuitively, an unanchored action type. *Occurs(do(ag,act),s1,s2)* means that the action of ag performing ac occurs between situations $s1$ and $s2$.

DP: A set *DP* $\subset$ *S* of *departure points*, intuitively corresponding to those situations in which an agent can decide to travel to the past, or is involuntarily dispatched to a point inthe past. (Or to the future, in models permitting such time travel.)

RP: A set *RP* $\subset$ *S* of *re-entry points*, intuitively corresponding to those situations to which an agent time travels.

AG: A set of agent embodiments ag. An agent embodiment (AE) ag may be primary, intuitively an agent who has not (yet) time traveled, or secondary, intuitively one who has time traveled.

A primary AE is represented as a, possibly subscripted; a secondary AE is represented as a $2n + 1$-tuple $(a, dp_1, rp_1, \ldots, dp_n, rp_n)$ where a is a primary agent, each $dp_i \in DP$, each $rp_i \in RP$, and (for backward time travel), for each dp_i, rp_i , it is the case that $dp_i > rp_i$.

For $n \geq 1$, we can represent the AE as $(a, dp_1, rp_1, \ldots, dp_{n-1}, rp_{n-1})(dp_n, rp_n)$.The AE $(a, dp_1, rp_1, \ldots, dp_{n-1}, rp_{n-1})$ is the *generating* AE, while $(a, dp_1, rp_1, \ldots, dp_n, rp_n)$ is the *generated* AE. Primary agents can only be generating AEs; secondary agents can be both generating and generated AEs.

Notation 4.1.2: $\hat{a}$ or *a'* is used to range over the primary AE a as well as secondary AEs who are recursively generated by a. This notation is useful when we wish to speak about various embodiments of a specific primary agent. (See the Proof in Example 4.5.2.)

τ: $\tau \subseteq AG \times DP \times RP$. That is, τ is the set of all triples of the form (ag, dp_i, rp_i) which give all the possible ways agent embodiments can travel through the time-travel tree structure. If (ag, dp_i, rp_i) is an element of τ, we say that rp_i is the re-entry point associated with dp_i, from ag's point of view. Note that there may be several re-entry points for a particular departure point of an AE, and several departure points for a re-entry point of a particular AE. Fig. 1 gives an example of the latter scenario.

T: A set of subtrees of S, one for each AE. Assume e is an AE $(a, dp_1, rp_1, \ldots, dp_n, rp_n)$.$T_e$ denotes the subtree rooted at rp_n, the time at which e is first active.

Isomorphisms between subtrees: For each primary AE a, T_a is the subtree of S during which a is active. Let Pa_s denote the subtree of T_a that is rooted at s. If e = $(a, dp_1, rp_1, \ldots, dp_n, rp_n)$, then there is an isomorphism between Pa_s and T_e. Let $\sigma(s)$ denote the image in T_e under this isomorphism. Note that if Pa_s and T_e share the root rp_i, then $\sigma(rp_i) = rp_i$. The existence of this isomorphism is what makes it possible to represent the secondary AE being faced with the same choices that the primary AE faced, and (possibly) making different choices.

4.2 Belief

We use a standard possible worlds semantics of belief as in (Fagin et al. 1995) Thus we have the standard definition of belief in terms of belief-accessible worlds:

Definition 4.2.1:

Holds(s,Bel(ag,p)) $\Leftrightarrow$ $\forall$ *s B(ag,s,s')* $\Rightarrow$ *Holds(s',p)*

We need to be able to say that the more an AE has time traveled, the more an agent believes. Equivalently, fewer possible worlds are belief-accessible to him. Recall that a generating AE and the corresponding generated AE inhabit separate worlds. This is built into the structure of the *TTT*: there are separate, isomorphic structures for generated AEs. We can use the mapping σ between the situations in the isomorphic structures to give us precisely what we need:

Axiom4.2.2:

$B(a(dp_1, rp_1, \ldots, dp_n, rp_n), \sigma(s), \sigma(s'))$
$\Rightarrow B(a(dp_1, rp_1, \ldots, dp_{n-1}, rp_{n-1}), s, s')$

[1] What follows is based on the theory of knowledge and action in (Davis and Morgenstern 2005); however, belief is used here instead of knowledge.

We add the usual KD45 axioms on belief.

E.g., consider the statements P,"If someone gets involved in a barroom brawl, he will have a shortened life span," and Q "If someone avoids a brawl, he becomes a wimp and will not have a meaningful life." Then Picard does not believes either statement in S1; as far as he believes, he can have both a long and meaningful life. At S7, Picard believes that in any branch in which someone brawls, he will not have a long life. That is, any world which is belief accessible to Picard and which is a successor situation to an AE brawling will be on a path in which the AE has a shortened life. Now, consider all such worlds W, and consider the image of such worlds under the isomorphism σ which maps T_{Picard} to $T_{Picard(S7,S1)}$, denoted $\sigma(W)$. Then, by Axiom 4.2.2, the worlds that are belief accessible to Picard(S7,S1) are a subset of $\sigma(W)$. That is, Picard(S7,S1) believes at least as much as Picard. Therefore, at S1, and in all subsequent situations for Picard(S7,S1), he believes P.

Similarly, at S13, having lived the meaningless wimpy life of the non-brawler, Picard(S7,S1) believes Q; we can show via application of Axiom 4.2.2 that in S1, and in all subsequent situations for Picard(S7,S1,S13,S1), Picard(S7,S1,S13,S1) believes Q as well.

4.3 Time Travel Narratives

We define a time travel narrative (TTN) from the perspective of a primary AE A. Intuitively, a TTN describes the intervals of time through which the AE lives. In our approach, this corresponds to a sequence of path segments in the *TTT*, with one path segment ending in a departure point, and the next path segment in the sequence beginning with its associated re-entry point.

Defining the TTN is a bit tricky, since a different AE is associated with each path segment. The following notation is helpful: If PS_i is a path segment, then $A(PS_i)$ is the active agent of PS_i.

Definition 4.3.1: A time travel narrative TTN_A is a sequence of path segments $PS_1 \ldots PS_n$ of the *TTT* that satisfy the following

1. PS_1 is a path segment of T_A.
2. The start and endpoints of the PS_i are characterized recursively as follows:
   (a) The end situation of PS_i is a departure point of A; the starting situation of PS_{i+1} is its associated re-entry point.
   (b) For any PS_i, $i > 2$, if $A(PS_{i-1}) = A(dp_1, rp_1, \ldots, dp_{i-2}, rp_{i-2})$, and dp_{i-1} is the end point of PS_{i-1}, then
      i. the starting point of PS_{i-1} is rp_{i-1}, where rp_{i-1} is dp_{i-1}'s associated re-entry point;
      ii. $A(PS_i) = A(dp_1, rp_1, \ldots dp_{i-1}, rp_{i-1})$.

Example 4.3.2: In Fig. 1, TTN_{Picard} is the sequence of path segments ((S1,S7),(S1,S13), (S1,S15)). (S1,S7) represents Picard's involvement in the brawl, leading to his premature death. S7 is a departure point; the associated re-entry point is S1. (S1,S13) represents the path segment in which Picard(S7,S1) avoids the brawl. S13 is the departure point whose associated re-entry point is S1.

4.4 Goals

For any TTN_A, A may have a goal or set of goals. A goal is represented as a fluent. Let G_j be a goal. G_j is achievable iff it holds in some future situation:

Definition 4.4.1:
Holds(s1,Achievable(G_j)) $\Leftrightarrow \exists s2 > s1(Holds(s2, G_j))$

We are interested in the cases where it is consistent with an agent's beliefs that a goal is achievable. It is straightforward to show that *Holds(s1,¬ Bel(A, ¬ Achievable(G_j)))* $\iff$ *∃ s2 B(A,S1,s2)* $\wedge \exists s3 > s2(Holds(s3, G_j))$.

A set of goals $G = \{G1 \ldots Gn\}$ is said to be achievable if the conjunction of the goals is achievable. An AE has a set of goals only if it is consistent with his beliefs that the conjunction is achievable:

Axiom 4.4.2: *Holds(s, Goalset(A,G))* $\Rightarrow$
Holds(s,¬Bel(A,¬Achievable($\bigwedge_{G_j \in G} G_j$*)))*

Frequently, a goal set is not achievable, leading to the question of which individual goals should be abandoned. We posit an ordering $<_g$ on subsets of G. A preferred subset of G is one that is minimal under this ordering.

4.5 Motivated Time-Travel Narratives

Intuitively, a time-travel narrative is motivated if an AE time travels only when he is in a serious bind and needs to revise history in order to achieve his goals. In this model, this can be expressed by saying that a TTN_A is motivated with respect to the time travel tree if each departure point is taken only after $\hat{A}$ comes to believe that one of his goals can no longer be realized. The associated re-entry point must be chosen so that it is consistent with $\hat{A}$'s beliefs that this life goal, or at least some preferred subset of his goals, can be realized in that re-entry point's future.

Definition 4.5.1
Let *TTT* be a time-travel tree. Let A be be a primary AE and let $TTN_A = PS_1 \ldots PS_n$. Assume that *Holds(start(PS1), Goalset(G))*. Then TTN_A is motivated with respect to *TTT* if the following condition holds:

For all $1 < i \leq n-1$,
if *Holds(end(PS_i), ¬Bel(A(PS_i),Achievable(G)))*, then one of the following is true:
(a) *Holds(start(PS_{i+1}),¬Bel(A(PS_{i+1}),¬Achievable(G)))*
(b) There is some G' that is a preferred subset of G such that *Holds(end(PS_i), ¬Bel(A(PS_i),¬Achievable(G')))*
(c) There is some G' that is a preferred subset of G such that *Holds(start(PS_{i+1}),¬Bel(A(PS_{i+1}),¬Achievable(G')))*

Condition (a) holds when it is consistent with one's beliefs that one can achieve all one's goals by starting over (i.e., re-entering the TTT); condition (b) holds when it is consistent with one's beliefs that one's preferred subset of goals can be achieved in the future (thus negating the need to do time travel at all); condition (c) holds when it is consistent with one's beliefs that a preferred subset of goals is achievable at some re-entry point.

Example: One can show that TTN_{Picard} (Example 4.3.2) is motivated with respect to the *TTT* of the example. Proof in (Morgenstern 2014).

5 Supplementary Examples

In the full paper (Morgenstern 2014), we distinguish between voluntary and involuntary time travel, and give formal characterizations of Second Chance narratives, in which a protagonist has the chance to fix a mistake in his past, and Serendipitous Nnarratives, in which a protagonist unexpectedly has a chance to improve his chances or circumstances. Examples of involuntary time travel are in *Peggy Sue Got Married* and *Groundhog Day*; *Peggy Sue* is a Second Chance narrative, while *Groundhog Day* is a Serendipitous narrative. Only some serendipitous narratives are time-travel stories; examples that are not time-travel narratives include many of the novels of Jane Austen, including *Persuasion*, *Emma*, and *Pride and Prejudice*.

6 Time-Travel Paradoxes

Time travel is subject to a number of well-known paradoxes and puzzles. Some of the best known are the grandfather/autoinfanticide paradox (Barjavel 1943; Horwich 1987), in which an agent travels back to the time when he was an infant, and kills his grandfather or the infant version of himself; the predestination paradox (Novikov 1998), in which no matter what an agent does, he cannot fix the past to prevent some disaster occurring; and closed loop and ontological paradoxes. In the full paper, we show that an advantage of the model proposed here is that these classic paradoxes of time travel do not occur, or occur in a less severe form, within our model.

7 Related Work

Modern theories of narratology, such as (Bal 2009) and (Abbot 2008), discuss representations of non-standard time within narratives. Their focus has generally been on narrative constructs such as flashbacks. None of the work in this field attempts to develop formal theories in which a story can be represented and reasoned about.

The interest of physicists in time travel dates back to the development of Einstein's theory of relativity, and his contention that spacetime is locally curved. Examples of discussions of time travel's feasibility include (Gödel 1949; Malament 1985; Friedman et al. 1990).

8 Evaluation and Future Work

We have classified dozens of time-travel stories (from fiction, film, and TV) with regard to several features and have evaluated whether our theory can handle such features and/or the extent of modification that would be necessary. A sample of this analysis is show in figure 2.

Future work includes extending the theory to handle more cases, especially that of interacting time-traveling agents, and exploring the connection between time-travel and Second Chance narratives.

Acknowledgements

I am grateful to Ernie Davis, Hojjat Ghaderi, Gerhard Lakemeyer, Stephanie Lewis, Sheila McIlraith, Barbara Partee, and Robert Sloan for their helpful comments and suggestions,and to MZ for his indefatigable reading and viewing of hundreds of time-travel stories and episodes, which has helped greatly with this research.

Feature	Example in Fiction	Handles?
Travel to past within lifetime	*Christmas Carol, Peggy Sue, Groundhog*	yes
Travel to any time in past	*Conn. Yankee*	yes
Future time travel	*Time Machine*	minor
Agent travels with object	*Story of the Amulet*	yes
Agent embodiment inhabits self	Star Trek Tapestry	yes
Agent embodiment observes self	*Prisoner of Azkaban*	moderate
Agent emobdiment interacts w. self	*Back to the Future2*	moderate
Multi-agent time travel	*Wizards of Waverly Place*	major
One predetermined future	*Slaughterhouse-Five*	no
Agent cannot change future or past but can bring knowledge back and forth	*Time Traveler's Wife*	no

Figure 2:

References

Abbot, H. P. 2008. *The Cambridge Introduction to Narrative, 2nd Edition*. Cambridge: Cambridge University Press.

Bal, M. 2009. *Narratology: Introduction to the Theory of Narrative, 3rd Edition*. Toronto: University of Toronto Press.

Barjavel, R. 1943. *Le Voyageuer Imprudent*. Gallimard.

Davis, E., and Morgenstern, L. 2005. A first-order theory of communication and multi-agent plans. *Journal of Logic and Computation* 15(5):701–749.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning About Knowledge*. Cambridge, Massachusetts: The MIT Press.

Friedman, J.; Morris, M. S.; Novikov, I. D.; Fernando, E.; Klinkhammer, G.; Thorne, K. S.; and Yurtsever, U. 1990. Cauchy problem in spacetimes with closed timelike curves. *Physical Review D* 42(6):1915–1930.

Gödel, K. 1949. An example of a new type of cosmological solution of Einstein's field equations of gravitation. *Reviews of Modern Physics* 21:447–450.

Horwich, P. 1987. *Asymmetries in Time*. MIT Press.

Malament, D. 1985. Minimal acceleration requirements for "time travel" in Gödel space-time. *Journal of Mathematical Physics* 26(4):774–777.

Miller, R., and Shanahan, M. 1994. Narratives in the situation calculus. *J. Log. Comput.* 4(5):513–530.

Morgenstern, L. 2008. A first-order theory of stanislavskian scene analysis. In *AAAI 2008*. AAAI Press.

Morgenstern, L. 2014. Representing and reasoning about time travel narratives: Extended paper. http://www-formal.stanford.edu/leora/papers or in ResearchGate.

Novikov, I. 1998. *The River of Time*. Cambridge U. Press.

Reiter, R. 2001. *Knowledge in Action*. Cambridge, Masachusetts: MIT Press.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Aggregative Deontic Detachment for Normative Reasoning

Xavier Parent and Leendert van der Torre
University of Luxembourg
6, rue Richard Coudenhove-Kalergi
L-1359 Luxembourg

Abstract

Aggregative deontic detachment is a new form of deontic detachment that keeps track of previously detached obligations. We argue that it handles iteration of successive detachments in a more principled manner than the traditional systems do. To study this new form of deontic detachment, we introduce a 'minimal' logic for aggregative deontic detachment, and we discuss various properties of the logic.

Aggregative Deontic Detachment

The *Handbook of Deontic Logic and Normative Systems* (Gabbay et al. 2013) reports on-going development of new standards for the formal study of normative reasoning. In his chapter "Alternative semantics for deontic logic", Hansson writes:

> "Deontic logic has ... a strong semantic tradition. Using possible worlds and orderings (preference relations) as their major building-blocks, deontic logicians have constructed models that can be used to determine the validity of deontic sentences. ... But, unfortunately, deontic logic also has a long history of getting stuck in semantic principles that support blatantly implausible deontic postulates. This is probably a major reason why the influence of deontic logic ... has been rather weak." (Hansson 2013, p. 447)

In the past, deontic logic seems to have been heavily, perhaps one-sidely, influenced by the modal logic paradigm, to which Hansson alludes here. However, given all the difficulties the traditionel systems face, deontic logic is currently undergoing a paradigm shift from modal logic to what may be called "norm-based semantics" (Hansen 2014). The core idea is to explain the truths of deontic logic not by some set of possible worlds among which some are ideal or at least better than others, but with reference to an explicit set of given norms or existing moral standards. The present short paper embraces such a paradigm shift.

To study aggregative deontic detachment we introduce a new logic for normative reasoning. It builds on so-called input/output (I/O) logic − one of the candidate systems for a new standard described in the *Handbook* (Parent and van der Torre 2013). Detachment (or modus-ponens) is the core mechanism of the semantics being used. The law of detachment (or modus-ponens) is well-known from propositional logic: from a conditional statement and its antecedent, the consequent of the conditional statement is inferred. In the deontic logic literature, this is referred to as "factual" detachment. The rule may be given the following form, where the conditional obligation for x given a is written as $\bigcirc(x \mid a)$:

$$\text{FD}\ \frac{\bigcirc(x \mid a) \qquad a}{\bigcirc x}$$

Van Eck asked, "How can we take seriously a conditional obligation if it cannot, by way of detachment, lead to an unconditional obligation?" (van Eck 1982, p. 263). Thus, it is no wonder that factual detachment is accepted as valid in many traditional systems of deontic logic, either in its plain form or in a qualified one. However, the traditional systems make some extra assumptions that are potentially controversial.

This can be illustrated with the Danielsson-Hansson-Lewis logic for conditional obligation (Danielsson 1968; Hansson 1969; van Fraassen 1972; Lewis 1973), which until recently was mainstream. First, it shares with the so-called classical theory of rational choice the assumption that an individual has (well-defined) preferences, and that a normative judgment is based on a maximization process. Advocates of −as Simon (1957) terms it−"bounded rationality" have argued that an approach to rationality in terms of maximization is not realistic, because human beings lack the cognitive resources to optimize. Second, such a logic takes the so-called trichotomy thesis for granted. It is the assumption that comparable items (worlds) can only be better than, worse than, or equal to each other in overall value. The trichotomy view has been challenged more than once. Some have argued that these three value relations do not exhaust the space of possibilities. The best-known proposal for a fourth *sui generis* relation is Chang's argument for "on a par"−see Chang (2002).

The best way to avoid potential objections is to make as few assumptions as possible. We believe that the assumption that conditionals obey the detachment rule is one that can hardly be challenged. Obligations and permissions are

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

contextual and vary based on the setting. Consequently, a norm always takes the form of a conditional statement. Some philosophers like Boghossian (2000) think (rightly, in our view) that the disposition to reason according to detachment is constitutive of the possession of the concept of conditional, and thus of the concept of norm. The idea is that, if some agent says "if a then x", and if he truly means it, then he commits himself to detaching x given a. If this agent refuses to acknowledge that he is justified in employing detachment, this will be good evidence that he fails to understand what is meant by "if ... then". Accepting detachment and acquiring an implication are two sides of the same coin.

The present paper focuses on the question of how to handle iterations of successive detachments. The standard I/O systems handle them, by injecting the following rule, known as "deontic detachment":

$$\text{DD}\ \frac{\bigcirc(y \mid x) \qquad \bigcirc x}{\bigcirc y}$$

That is fully in line with the tradition in deontic logic. For instance, the Danielsson-Hansson-Lewis semantics for conditional obligation validates such a law. However, potential counterexamples to DD may be found in the literature (McLaughlin 1955; Hansson 1997; Makinson 1999). They all rely on the intuition that the obligation of y ceases to hold when the obligation of x is violated. This can be illustrated with the following example, due to Broome (2013, §7.4):

> You ought to exercise hard everyday
> If you exercise hard everyday, you ought to eat heartily
> ?* You ought to eat heartily

Intuitively, the obligation to eat heartily no longer holds, if you do not take exercise. Like the others, this counterexample suggests an alternative (call it "aggregative") form of detachment, which keeps track of what has been previously detached:

$$\text{ADD}\ \frac{\bigcirc(y \mid x) \qquad \bigcirc x}{\bigcirc(x \wedge y)}$$

This form of detachment has been overlooked in the literature.

To study aggregative deontic detachment we define a system supporting ADD, but not DD. Accepting ADD, but not DD, implies rejecting the rule known as "weakening". This is the rule, where $\vdash$ stands for the deducibility relation in propositional logic:

$$\text{W}\ \frac{\bigcirc(x \mid a) \qquad x \vdash y}{\bigcirc(y \mid a)}$$

ADD and W together imply DD.[1]

We provide not only a suitable semantics, but also a sound and complete axiomatization for the new logic. Due to space limitations we do not give the soundness and completeness proofs here, but we do present some properties of the new logic.

[1](van der Torre and Tan 1998) define a system PDL validating ADD, but not DD. However, PDL has a preference-based semantics.

Logic for Aggregative Deontic Detachment

In input/output logic, a normative code is a set N of conditional obligations. A conditional obligation is a pair (a, x), where a and x are two formulae of classical propositional logic. We use this notation instead of $\bigcirc(x \mid a)$, because the latter has distinct interpretations in the literature. In the notation (a, x), the first element a is called the body of the rule, and is thought of as an input, representing some condition or situation. The second element x is called the head of the rule, and is thought of as an output, representing what the norm tells us to be obligatory in that situation. We use the standard notation $(\top, x)$ for the unconditional obligation of x, where $\top$ is a zero-place connective standing for 'tautology'.

Some further notation. $\mathcal{L}$ is the set of all formulae of classical propositional logic. Given an input $A \subseteq \mathcal{L}$, and a set of generators N, $N(A)$ denotes the image of N under A, i.e., $N(A) = \{x : (a, x) \in N \text{ for some } a \in A\}$. $Cn(A)$ denotes the set $\{x : A \vdash x\}$, where $\vdash$ is the deducibility relation used in classical propositional logic. The notation $x \dashv\vdash y$ is short for $x \vdash y$ and $y \vdash x$. We use PL as an abbreviation for (classical) propositional logic. For the sake of conciseness, we denote by $h(M)$ the set of all the heads of elements of M. Moreover, $\mathcal{B}_A^M$ denotes the set of all B such that $A \subseteq B = Cn(B) \supseteq M(B)$. Intuitively, $\mathcal{B}_A^M$ gathers all the Bs that contain A and are closed under both Cn and M.

Definition 1 (Semantics). *$x \in \mathcal{O}(N, A)$ if and only if there is some finite $M \subseteq N$ such that,*

- *$M(Cn(A)) \neq \emptyset$, and*
- *for all B, if $B \in \mathcal{B}_A^M$, then $x \dashv\vdash \wedge M(B)$.*

We call such a M an A-witness for x. $\mathcal{O}$ is a variation of the operation *out*$_3$ defined by Makinson and van der Torre (2000). We have *out*$_3(N, A) = \cap\{Cn(N(B)) : B \in \mathcal{B}_A^N\}$.

While *out*$_3$ validates DD, $\mathcal{O}$ does not, as illustrated below.

Example 1 (Deontic Detachment). *Consider the two norms $N = \{(\top, a), (a, x)\}$. We have obligation $a \in \mathcal{O}(N, \top)$, since $M = \{(\top, a)\}$ is a $\top$-witness for a. We also have $x \in \mathcal{O}(N, a)$, since $M = \{(a, x)\}$ is an a-witness for x. But we do not have $x \in \mathcal{O}(N, \top)$. To see this, it suffices to go through all the non-empty subsets of N, and check that, for each of them, the smallest relevant B does not deliver heads whose conjunction is equivalent to x:*

M	B	$M(B)$
$\{(\top, a)\}$	$Cn(a)$	$\{a\}$
$\{(a, x)\}$	$Cn(\emptyset)$	$\emptyset$
$\{(\top, a)(a, x)\}$	$Cn(a, x)$	$\{a, x\}$

We define $\mathcal{O}(N) = \{(A, x) : x \in \mathcal{O}(N, A)\}$. This definition leads to an axiomatic characterization that is much alike those used for conditional logic.

Given a set R of rules, (a, x) is said to be derivable from N if (a, x) is in the least superset of N that is closed under the rules in R. This may be written as $(a, x) \in \mathcal{D}(N)$, or equivalently $x \in \mathcal{D}(N, a)$. The specific rules of interest here are described below. They are formulated for a singleton input set A (for such an input set, curly brackets will be omitted). SI, EQ and ACT abbreviate "strengthening of the input," "equivalence" and "aggregative cumulative transitivity."

Definition 2 (Proof system). $(a, x) \in \mathcal{D}(N)$ *if and only if* (a, x) *is derivable from* N *using the rules* {*SI, EQ, ACT*}.

$$SI\ \frac{(a,x) \qquad b \vdash a}{(b,x)} \qquad EQ\ \frac{(a,x) \qquad x \dashv\vdash y}{(a,y)}$$

$$ACT\ \frac{(a,x) \qquad (a \wedge x, y)}{(a, x \wedge y)}$$

ACT yields ADD as a special case (a is $\top$).

Where A is a set of formulae, $(A, x) \in \mathcal{D}(N)$ means that $(a, x) \in \mathcal{D}(N)$, for some conjunction a of elements in A. Moreover, $\mathcal{D}(N, A)$ is $\{x : (A, x) \in \mathcal{D}(N)\}$.

Theorem 1 (Completeness). $\mathcal{O}(N, A) = \mathcal{D}(N, A)$

Below is an illustration.

Example 2 ("Change your mind!"). *Hansson (1997) gives the following example, with credit to Pörn:*

> *"Consider the hoary example of the man who ought to go to a meeting on August 5 and who ought to send, on August 2, a note explaining his absence, if and only if he is in fact going to be absent." (1997, p.425-6)*

The example is structurally identical to the Chisholm example (Chisholm 1963). The norms involved may be rendered as $N = \{(\top, m), (m, s)(\neg m, \neg s)\}$*, where* m *and* s *are for attending the meeting and sending a note, respectively. Given input* $\top$*,* $m \wedge s$ *is outputed, but not* s*. This is as it should be. The obligation of* s *will not be triggered unless the agent is going to fulfil his primary obligation of* m*. In the violation context* $\neg m$*,* $m \wedge s$ *is still outputed. If not, then the following intuitive deontic reasoning pattern would not be supported:*

> *"August 2 arrives, and though he is able to attend the meeting, he has no intention of doing so. He argues: 'I ought to change my mind, forbear note-writing, and attend the meeting. ... My present fulfillment of this obligation will help make up for my sinfully staying at home on the fifth!'." (Hansson 1997, p. 426)*

In the remainder of this paper, we consider some properties of this new logic.

Properties

Deontic properties

Exact factual detachment (EFD) and violation detection (VD) characterise what is special about *deontic* logic.

$$\text{EFD} \quad (x, y) \in N \Rightarrow y \in \mathcal{O}(N, x)$$
$$\text{VD} \quad (A, y) \Rightarrow (A \cup \{\neg y\}, y)$$

Exact factual detachment represents that if there is a norm with precisely the context as body, then the output contains the head. On the one hand, exact factual detachment is relatively weak, as the context has to be *precisely* the body. The much stronger factual detachment principle (FD) discussed in the introduction imposes detachment when the body is *implied* by the context. On the other hand, exact factual detachment principle is already quite strong, as in context a from the norm $(a, \bot)$ the contradiction $\bot$ is detached, and in case of a dilemma of (a, x) and $(a, \neg x)$, in context a both x and $\neg x$ are detached.

Violation detection says that violations are explicit. The distinctive feature of norms and obligations with respect to other types of rules and modalities is that they can be violated. For example, dilemma examples arise because some obligation has to be violated, and contrary-to-duty examples arise because some obligation has been violated. Modal logic offers a useful representation for violations. An obligation for x has been violated if and only if we have $\neg x \wedge \bigcirc x$. In our notation with explicit norms, this is $x \in \mathcal{O}(N, A)$ with $\neg x \in Cn(A)$.

Theorem 2. $\mathcal{O}$ *satisfies* EFD, FD *and* VD.

Logical properties

Substitution (SUB), replacements of logical equivalents (RLE), implication (IMP) and paraconsistency (PC) say something about *logic*.

$$\begin{array}{ll} \text{SUB} & x \in \mathcal{O}(N, A) \Rightarrow x[\sigma] \in \mathcal{O}(N[\sigma], A[\sigma]) \\ \text{RLE} & N \approx M \Rightarrow \mathcal{O}(N) \subseteq \mathcal{O}(M) \\ \text{IMP} & \mathcal{O}(N, A) \subseteq Cn(m(N) \cup A) \\ \text{PC} & x \in \overline{V}(N, A) \Rightarrow \exists M \subseteq N : x \in \mathcal{O}(M, A) \\ & \text{and } \mathcal{O}(M, A) \cup A \text{ consistent} \end{array}$$

Substitution is well known from classical propositional logic. We use the notation $x[\sigma]$ to denote a substitution instance of x. Thus, $x[\sigma]$ is obtained from x by replacing uniformly, in x, all occurrences of a propositional letter by the same propositional formula. $A[\sigma]$ and $N[\sigma]$ extend the notion of substitution instance to sets of formulae, and sets of norms in the straightforward way.

Replacement for logically equivalent expressions expresses a principle of irrelevance of syntax. For $N \approx M$, read "N is (logically) equivalent to M". The simplest way to define $\approx$ is as follows: $N \approx M$ whenever M is obtained from N, by replacing each $(b, y) \in N$ with some (c, z) such that b is equivalent with c, and y is equivalent with z.

Implication makes use of the so-called materialisation of a normative system, which means that each norm (a, x) is interpreted as a material conditional $a \rightarrow x$, i.e. as the propositional sentence $\neg a \vee x$. The implication property says that if the materializations of N, written as $m(N)$, do not imply $a \rightarrow x$, then $(a, x) \notin \mathcal{O}(N)$. This represents the idea that we cannot derive more than we can derive in propositional logic. In general, implication in the base logic is the upper bound.

To prevent explosion we do not want to derive the whole language, unless maybe in pathological cases in which the normative system contains a norm for each propositional formula. A consequence relation may be said to be paraconsistent if it is not explosive, though there are various ways to make this formal.

To define our paraconsistency property, we distinguish obligations representing violations from other obligations. That is, we decompose an operator $\mathcal{O}(N, A)$ into two operators $V(N, A)$ and $\overline{V}(N, A)$, such that

- $V(N, A) = \{x \in \mathcal{O}(N, A) \mid \neg x \in Cn(A)\}$
- $\overline{V}(N, A) = \mathcal{O}(N, A) \setminus V(N, A)$

Trivially, we have

$$\mathcal{O}(N, A) = V(N, A) \cup \overline{V}(N, A)$$

The basic idea of our paraconsistency property is that obligations in $\overline{V}$ can be derived from a set of norms M in N, such that this set of norms M does not explode.

The underlying intuition to restrict to a set of norms is raised in the following example. If we can derive $\bigcirc(y \wedge z)$ from $\bigcirc(x \wedge y)$ and $\bigcirc(\neg x \wedge z)$, and we have substitution and replacements of logical equivalents, then we also derive $\bigcirc y$ from $\bigcirc x$ and $\bigcirc \neg x$, in other words, we have deontic explosion. This can be verified by replacing y by $x \vee y$ and z by $\neg x \vee y$. Therefore, we restrict the set of norms we use to a set of norms which is in some sense "consistent" with the input A.

Theorem 3. *$\mathcal{O}$ satisfies* SUB*,* RLE*,* IMP *and* PC*.*

Norm change properties

The next two properties are: norm monotony (NM) and norm induction (NI). We call them "norm change properties", because the N is no longer held constant.

NM $\quad \mathcal{O}(N) \subseteq \mathcal{O}(N \cup M)$

NI $\quad M \subseteq \mathcal{O}(N) \Rightarrow \mathcal{O}(N) = \mathcal{O}(N \cup M)$

The most interesting property is norm induction. It says that if there is an output x for an input a, and we add the norm (a, x) to the normative system, then for all inputs, the output of the normative system stays the same. This is called "norm induction", because the norm is induced from the relation between facts and obligations.

Theorem 4. *$\mathcal{O}$ satisfies* NM *and* NI*.*

Together, exact factual detachment, norm monotony and norm induction are equivalent to saying that $\mathcal{O}$ is a closure operator.

Reusability properties

The reusability properties relate the system to standard I/O logic. Their formulation appeals to some key notions of so-called constrained input/output logic, developed by Makinson and van der Torre (2001) in order to reason about norm violation:

$conf(N, A) = \{N' \subseteq N \mid out(N', A) \cup A \text{ consistent } \}$

$maxf(N, A) = \{N' \in conf(N, A) \mid N' \subseteq \text{-maximal } \}$

$outf(N, A) = \{out(N.A) \mid N' \in maxf(N, A)\}$

The first property listed below represents inclusion in reusable output (IO). The following two properties define redundancy R and strong redundancy SR.

IO $\quad \mathcal{O}(N, A) \subseteq out_3(N, A)$

R $\quad outf_3(N, A) = outf_3(\mathcal{O}(N), A)$

SR $\quad outf_3(N \cup M, A) = outf_3(\mathcal{O}(N) \cup M, A)$

Redundancy is worth a mention, because it makes a bridge to constrained input/output logic.

Summary

This paper has introduced a 'minimal' logic for aggregative deontic detachment, with a sound and complete axiomatization. Various properties of the logic have been discussed. Summarizing, the operator $\mathcal{O}(N)$ is a closure operator, which means that it satisfies factual detachment, norm monotony and norm induction. In addition, it satisfies substitution and replacement of logical equivalents. Finally, the operator satisfies violation detection, implication, and paraconsistency.

References

Boghossian, P. 2000. Knowledge of logic. In Boghossian, P., and Peacocke, C., eds., *New Essays on the A Priori*. Oxford: Clarendon Press. 229–254.

Broome, J. 2013. *Rationality Through Reasoning*. West Sussex, UK: Wiley-Blackwell.

Chang, R. 2002. The possibility of parity. *Ethics* 112:659–688.

Chisholm, R. 1963. Contrary-to-duty imperatives and deontic logic. *Analysis* 24:33–36.

Danielsson, S. 1968. *Preference and Obligation: Studies in the Logic of Ethics*. Uppsala: Filosofiska freningen.

Gabbay, D.; Horty, J.; Parent, X.; van der Meyden, R.; and van der Torre, L., eds. 2013. *Handbook of Deontic Logic and Normative Systems*. London, UK: College Publications.

Hansen, J. 2014. Reasoning about permission and obligation. In Hansson, S. O., ed., *David Makinson on Classical Methods for Non-Classical Problems*. Springer. 287–333.

Hansson, B. 1969. An analysis of some deontic logics. *Noûs* 3:373–398.

Hansson, S. O. 1997. Situationist deontic logic. *Journal of Philosophical Logic* 26(4):423–448.

Hansson, S. 2013. Alternative semantics for deontic logic. In Gabbay et al. (2013). 445–498.

Lewis, D. 1973. *Counterfactuals*. Oxford: Blackwell.

Makinson, D., and van der Torre, L. 2000. Input/output logics. *Journal of Philosophical Logic* 29(4):383–408.

Makinson, D., and van der Torre, L. 2001. Constraints for input/output logics. *Journal of Philosophical Logic* 30(2):155–185.

Makinson, D. 1999. On a fundamental problem in deontic logic. In Namara, P. M., and Prakken, H., eds., *Norms, Logics and Information Systems*, Frontiers in Artificial Intelligence and Applications. Amsterdam: IOS Press. 29–54.

McLaughlin, R. N. 1955. Further problems of derived obligation. *Mind* 64(255):400–402.

Parent, X., and van der Torre, L. 2013. Input/output logic. In Gabbay et al. (2013). 499–544.

Simon, H. 1957. *Models of Man, Social and Rational*. New York: John Wiley and Sons.

van der Torre, L., and Tan, Y.-H. 1998. Prohairetic deontic logic (PDL). In Dix, J.; Fariñas del Cerro, L.; and Furbach, U., eds., *Logics in Artificial Intelligence*. Berlin Heidelberg: Springer. 77–91.

van Eck, J. 1982. A system of temporally relative modal and deontic predicate logic, and its philosophical applications. *Logique & Analyse* 25:249–290 and 339–381.

van Fraassen, B. C. 1972. The logic of conditional obligation. *Journal of Philosophical Logic* 1:417–438.

Action Theories over Generalized Databases with Equality Constraints (Extended Abstract)

Fabio Patrizi and Stavros Vassos
Department of Computer, Control, and Management Engineering (DIAG)
Sapienza University of Rome
Rome, Italy

Abstract

In this work we study action theories of the situation calculus such that the initial KB is a generalized database with equality constraints (GFDBs). We show that GFDBs characterize the class of definitional KBs and that they are closed under progression. We also show that, under conditions, generalized projection queries can be decided based on an induced transition system and evaluation of local conditions over states.

Introduction

Situation calculus basic action theories (BATs) (Reiter 2001) consist of a first-order knowledge base (KB) which describes the initial state of a given domain, and a set of first-order axioms that specify how the properties of the domain change under the effects of actions. Two important reasoning problems are *projection*, i.e., *predicting* whether a condition would hold if a series of actions were to be performed, and *progression*, i.e., *updating* the KB by a new description that reflects the current state after actions have been performed.

For the general case it has been shown that second-order logic may be required to capture the updated KB (Lin and Reiter 1997; Vassos and Levesque 2013) and an overview of some special cases where it becomes first-order is studied in (Vassos and Patrizi 2013). Similarly, answering projection queries is in the general case undecidable and a few decidable cases have been studied, e.g., when the KB is a regular database (Reiter 2001) or the case of a modified version of the situation calculus language built using a two-variable fragment of first-order logic (Gu and Soutchanski 2007).

Notably, the case when the KB has the form of a *generalized database with constraints* (Kanellakis, Kuper, and Revesz 1995), which allows to specify relations with possibly *infinitely many tuples*, has not been investigated. In this work we show that for a special type of BATs whose KB is a generalized database with equality constraints, projection is decidable and a first-order progression can always be computed. We then look into richer forms of projection that may refer to more than one possible evolution of the initial KB, e.g., capturing properties of the form "after execution of $\alpha_1, \ldots, \alpha_n$ condition ϕ always holds in the future" and specify a condition that ensures decidability also for them.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

BATs with generalized fluent databases

The situation calculus as presented by Reiter (2001) is a three-sorted first-order language with equality (and some limited second-order features). The sorts are used to distinguish between actions, situations, and objects. A *situation* represents a world history as a sequence of actions. S_0 is used to denote the initial situation where no actions have occurred. Sequences of actions are built using the function symbol *do*, such that $do(a, s)$ denotes the successor situation resulting from performing action a in situation s. A (relational) *fluent* is a predicate whose last argument is a situation, whose value can change from situation to situation. We also assume a finite number of fluent and action symbols, $\mathcal{F}$ and $\mathcal{A}$, and an infinite number of constants $\mathcal{C}$.

Reiter (2001) investigates the case where the initial knowledge base (KB) is a *definitional theory* characterized as follows: $\bigwedge_{F_i \in \mathcal{F}} \forall \vec{x}_i . F(\vec{x}_i, S_0) \equiv \phi_i(\vec{x}_i)$, where $\phi_i(\vec{x}_i)$, called the *definition* for F_i, is an unrestricted first-order formula mentioning no situations. Definitional KBs express *complete* information for fluents under the assumption of the unique-name axioms for constants captured in a set $\mathcal{E}$. For example the following axiom states that there are exactly two atoms true for $In(x_1, x_2, S_0)$, namely $In(box, it_1, S_0)$ and $In(box, it_2, S_0)$:

$$\forall x \forall y (In(x, y, S_0) \equiv (x{=}box \wedge (y{=}it_1 \vee y{=}it_2))).$$

Nonetheless, the definition for a fluent can be any unrestricted first-order formula built over the constants in $\mathcal{C}$ and equality. For instance, it could have the following form implying an infinity of ground atoms that are true in S_0:

$$\forall x \forall y (In(x, y, S_0) \equiv (x{\neq}box \wedge (y{=}it_1 \vee y{=}it_2))).$$

More complicated definitions can be formed also using quantification which is actually not adding to the expressiveness as first-order theories of equality admit quantifier elimination (Enderton and Enderton 2001).

We now present definitions from (Kanellakis, Kuper, and Revesz 1995) in the context of the situation calculus.

Definition 1. An *equality constraint* is a literal formula $x\theta y$ or $x\theta c$, where $c \in \mathcal{C}$ and θ is $=$ or $\neq$. A *generalized k-tuple* over variables $x_1, \ldots, x_k$ is a finite conjunction ψ of equality constraints whose all variables are free and among $x_1, \ldots, x_k$. A *generalized relation of arity k* is a finite set

$R = \{\psi_1, \ldots, \psi_q\}$, of generalized k-tuples over $x_1, \ldots, x_k$. The *formula corresponding to a generalized relation* R is the disjunction $\psi_1 \vee \cdots \vee \psi_q$. We will use ϕ_R to denote the quantifier-free formula corresponding to relation R.

We will be dealing with the so-called *basic action theories* of the form $\mathcal{D} = \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_0 \cup \Sigma$, where: $\mathcal{D}_{ap}$ is a set of action precondition axioms (APs) and $\mathcal{D}_{ss}$ is a set of successor state axioms (SSAs), one per fluent symbol $F_i \in \mathcal{F}$, $\mathcal{D}_{una}$ is the set of unique-names axioms for actions, $\mathcal{D}_0$ describes the initial situation S_0, and Σ is a set of foundational axioms which formally define legal situations. We identify BATs over generalized databases as follows.

Definition 2. A set $\mathcal{D}_0$ of first-order sentences uniform in S_0 is a *generalized fluent database (GFDB)* iff it has the form $\bigwedge_{F_i \in \mathcal{F}} \forall \vec{x}_i . F(\vec{x}_i, S_0) \equiv \phi_i(\vec{x}_i)$, where $\phi_i(\vec{x}_i)$ corresponds to a generalized relation over $\vec{x}$, i.e., is a disjunction of conjunctions of equality constraints. A basic action theory $\mathcal{D}$ is a *basic action theory over a generalized fluent database (BAT-GFDB)* iff it also includes the set of unique-names axioms for constants $\mathcal{E}$, and $\mathcal{D}_0$ is a GFDB.

Theorem 1. *Let ϕ be a definitional KB. There exists a generalized fluent database ϕ' such that $\mathcal{E} \models \phi \equiv \phi'$.*

Kanellakis *et. al* (1995) show that for such KBs there is also a decidable procedure for evaluating queries. Note also that equivalence of generalized fluent databases can also be decided, formed as an appropriate query. With these tools available we now proceed to show how solutions for progression and projection can be obtained for BAT-GFDBs.

Progression of BAT-GFDBs

In order to do a one-step progression of a BAT $\mathcal{D}$ with respect to a ground action α we need to replace $\mathcal{D}_0$ in $\mathcal{D}$ by a suitable set $\mathcal{D}_\alpha$ of sentences uniform in $do(\alpha, S_0)$ so that the original theory $\mathcal{D}$ and the theory $(\mathcal{D} - \mathcal{D}_0) \cup \mathcal{D}_\alpha$ are equivalent with respect to how they describe the situation $do(\alpha, S_0)$ and the situations in the future of $do(\alpha, S_0)$.

In a seminal paper Lin and Reiter (1997) gave a model-theoretic definition for the progression $\mathcal{D}_\alpha$ of $\mathcal{D}_0$ wrt α and $\mathcal{D}$ that achieves this goal. Finding such a $\mathcal{D}_\alpha$ is a difficult task and it has been shown that second-order logic may be required in the general case (Lin and Reiter 1997; Vassos and Levesque 2013). Nonetheless, for the definitional KBs, and as a consequence also for the special case of generalized fluent databases, there is a simple way to progress. Note that an SSA, that has the form $F(\vec{x}, do(a, s)) \equiv \Phi(\vec{x}, a, s)$, characterizes the conditions under which F has a specific value at situation $do(a, s)$ as a function of situation s and action a. A progression can then be obtained by replacing fluent atoms $F(\vec{o}, S_0)$ in the right-hand side of the SSA by their definition in the KB (Reiter 2001). It is interesting to look into how this method works when $\mathcal{D}_0$ is a GFDB.

Note that since each $\Phi(\vec{x_i}, \alpha, S_0)$ in the right-hand side of SSAs is in general unrestricted, e.g., may include quantifiers, $\mathcal{D}_\alpha$ is not guaranteed to be in the form of a generalized fluent database even though $\mathcal{D}_0$ is. The point in using a form like the generalized fluent database is that it allows us to perform query evaluation using the methods and existing technologies in constraint databases instead of performing more general theorem proving. Therefore, we want progression to preserve the form of $\mathcal{D}_0$. The previous substitution method does well in preserving the form of a *definitional KB* but does not preserve the form in the case of a GFDB.

The idea then is to consider generalized tuples as the "base" formulas that we use to express any generalized fluent relation. This is similar to a regular database where we would update $\mathcal{D}_0$ into a $\mathcal{D}_\alpha$ such that for every fluent a finite list of tuples is specified. Theorem 1 then provides a way to transform, by means of quantifier elimination, the resulting $\mathcal{D}_\alpha$ of the substitution method into the form of a GFDB.

Theorem 2. *Let $\mathcal{D}$ be a BAT over a generalized fluent database and α a ground action. Then there exists a first-order progression $\mathcal{D}_\alpha$ of $\mathcal{D}_0$ wrt α and $\mathcal{D}$ that is in the form of a generalized fluent database.*

As a result, we can *iteratively* progress a BAT-GFDB and characterize the state of any ground situation as a GFDB.

Finally, since every definitional KB can be expressed as a generalized fluent database, this also illustrates a subtle detail about progression. Both a progression $\mathcal{D}_\alpha$ according to the substitution method and a progression $\mathcal{D}'_\alpha$ according to Theorem 2 qualify as logically correct progressions of $\mathcal{D}_0$ and are logically equivalent (under the assumption of $\mathcal{E}$). Nonetheless, $\mathcal{D}_0$ is more of a *logical specification of the changes* that need to be made due to action α and $\mathcal{D}'_\alpha$ more of a *materialized update* of these changes.

Another way to look at it is that the progression based on the substitution method is purely syntactic (linear to the size of $\mathcal{D}_0$) and does not involve any form of evaluation; in a sense, the fluents are not updated to a new truth value but, rather, the new truth values are still specified with respect to the initial situation. As there are to date no implemented systems for performing progression, this difference is less easy to identify because, as far as the logical specification is concerned, $\mathcal{D}_0$ is a well-behaved progression. In practice, though, we would expect that unless a method that goes along the lines of Theorem 2 is followed, the progressed theory would in fact look much similar to the original theory $\mathcal{D}$, and queries over the current state would perform similar to using a *regression* technique over the original theory $\mathcal{D}$.

Projection over BAT-GFDBs

The *(simple) projection problem* is the task of *predicting* whether a condition holds at a particular time in the future after a series of ground actions have been executed (Reiter 2001). The following is a straightforward result.

Theorem 3. *Let $\mathcal{D}$ be a BAT-GFDB, $\alpha_1, \ldots, \alpha_n$ a sequence of ground actions, and $\phi(s)$ a first-order formula uniform in s. Then determining whether or not the following holds is decidable: $\mathcal{D} \models \phi(do(\alpha_n, \cdots do(\alpha_1, S_0)))$.*

This is not a new result and can be proven by means of regression and the fact that $\mathcal{E}$ is decidable. Our previous analysis also shows that simple projection queries over a BAT-GFDB can be decided by iteratively progressing $\mathcal{D}_0$ wrt $\alpha_1, \ldots, \alpha_n$ according to Th. 2 and then evaluating the query over the resulting generalized fluent database following the

method of (Kanellakis, Kuper, and Revesz 1995). Depending on the type of queries, and the frequency that actions occur, either approach may be preferred under conditions.

We now proceed to show a major result about the decidability of richer projection queries over BAT-GFDBs that may also quantify over future situations. A *generalized* version of the projection problem is when ϕ may refer to any number or combination of future situations. For example, $\forall s(do(\alpha, S_0) \sqsubseteq s \supset \psi(s))$ states that after executing action α condition $\psi(s)$ will hold in all situations s in the future, i.e., that it will always remain true.

We consider the language $\mathcal{L}_p$ of *generalized projection queries* φ. $\mathcal{L}_p$ is defined on top of the language $\mathcal{L}_n$, whose formulas ϕ are as follows: $\phi := x = c \mid x = y \mid F(\vec{x}, s) \mid F(\vec{x}, \sigma) \mid \neg\phi \mid \phi \wedge \phi \mid \exists x.\phi$, for F a fluent symbol, c a constant, and σ a ground situation term. $\mathcal{L}_p$ formulas are defined as: $\varphi := \phi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists s.\sigma \sqsubseteq s \wedge \varphi$, where $\phi \in \mathcal{L}_n$ is any formula uniform in s or in a ground situation term σ, whose free variables (if any) are only of sort situation.

We also consider a class of BAT-GFDBs which we call C-bounded. To define it, let T_V be the (finite) set of all generalized tuples ψ that use equality constraints with (only variable) symbols from the finite set of variables V, and s.t. ψ does not contain multiple occurrences of some equality constraint. Notice that since generalized tuples are conjunctions (thus multiple occurrences of a conjunct do not change their semantics), T_V essentially contains all the possible generalized tuples one can build using symbols from V.

Definition 3. Let $\mathcal{D}$ be a BAT-GFDB and B a natural number. A ground situation term σ is said to be *constant-bounded by B in $\mathcal{D}$*, C-bounded by B for short, iff for every fluent $F_i(\vec{x}, s) \in \mathcal{F}$, it is the case that $\mathcal{D} \models \bigvee_{\Psi \in 2^{T_V}} \exists \vec{y}.F(\vec{x}, \sigma) \equiv \bigvee_{\psi \in \Psi} \psi$, where: V is partitioned into X and Y, with X the set of variables occurring in $\vec{x}$, Y any set of variable symbols such that $|Y| = B$, and $\vec{y}$ are the free variables of ψ coming from Y. $\mathcal{D}$ is said to be *constant-bounded by a finite bound B*, C-bounded by B for short, iff every ground situation term of $\mathcal{D}$ that is executable is also C-bounded by B.

Notice that Ψ above is a set of generalized tuples, thus the formula $\bigvee_{\psi \in \Psi} \psi$ is a generalized relation. Intuitively, Def. 3 requires that the definition of each fluent in σ is a generalized relation mentioning at most B distinct constants.

For this class of theories, we have the following result.

Theorem 4. *Given a BAT-GFDB $\mathcal{D}$ that is C-bounded by some B, and a generalized projection query φ that is a sentence in $\mathcal{L}_p$, it is decidable to check whether $\mathcal{D} \models \varphi$.*

The proof starts by introducing the notion of *transition system over generalized fluent databases (GFDB-TS)*. These are transition systems with states labelled by generalized fluent databases and transitions labelled by ground actions. The aim of GFDB-TSs is to capture the information, represented as GFDBs, about both the states associated with the executable situations of an action theory, and their transitions, together with the respective actions. In particular the GFDB-TS $T_{\mathcal{D}}$ *induced* by an action theory $\mathcal{D}$ can be defined, which accounts for the evolution of the states of the theory starting from S_0.

Then, a definition of satisfaction of $\mathcal{L}_p$ sentences wrt GFDB-TSs is provided, and it is proven that, for any generalized projection query φ, $\mathcal{D} \models \varphi$ iff $T_{\mathcal{D}} \models \varphi$. Thus, one can check whether $\mathcal{D} \models \varphi$ by using the induced GFDB-TS $T_{\mathcal{D}}$ instead of the theory $\mathcal{D}$. However, this does not guarantee decidability, as $T_{\mathcal{D}}$ is infinite-state, in general.

The final step of the proof consists in defining a procedure to build a finite-state GFDB-TS $\hat{T}_{\mathcal{D}}$ such that, for any sentence $\varphi \in \mathcal{L}_p$, $T_{\mathcal{D}} \models \varphi$ iff $\hat{T}_{\mathcal{D}} \models \varphi$. The construction exploits the key observations that if $\mathcal{D}$ is C-bounded then: *(i)* it is first-order progressable; and *(ii)* its states can be partitioned into finitely many equivalence classes of logically equivalent GFDBs. Essentially, the procedure amounts to iteratively progressing $\mathcal{D}$ from S_0, by executing all possible ground actions obtained using a finite subset $H \subseteq \mathcal{C}$ of constants, until a state whose label is logically equivalent to that of some state visited in the past is generated. Once $\hat{T}_{\mathcal{D}}$ is available, checking whether $\hat{T}_{\mathcal{D}} \models \varphi$ can be done by applying a straightforward recursive procedure.

Conclusions

In this paper we looked into situation calculus action theories over generalized fluent databases with equality constraints (GFDB), connecting the situation calculus with constraint query languages. We showed that GFDBs characterize the class of definitional KBs with fluent predicates only, and that for action theories over such KBs (BAT-GFDBs), the KBs are *closed under progression*. Our results show that simple projection queries over BAT-GFDBs are decidable in general. Also, extending the notion of *boundedness* proposed in (De Giacomo, Lespérance, and Patrizi 2012), we introduced the notion of *C-boundedness* and showed that, under this, a wide class of generalized projection queries that include quantification over situations is decidable.

Acknowledgements The authors acknowledge support of the EU Project FP7-ICT 318338 (OPTIQUE) and the Sapienza Award 2013 "Spiritlets" project.

References

De Giacomo, G.; Lespérance, Y.; and Patrizi, F. 2012. Bounded situation calculus action theories and decidable verification. In *Proc. of KR'12*.

Enderton, H., and Enderton, H. B. 2001. *A Mathematical Introduction to Logic*. Academic Press, 2nd edition.

Gu, Y., and Soutchanski, M. 2007. Decidable reasoning in a modified situation calculus. In *Proc. of IJCAI'07*.

Kanellakis, P. C.; Kuper, G. M.; and Revesz, P. Z. 1995. Constraint query languages. *Journal of Computer and System Sciences* 51(1):26–52.

Lin, F., and Reiter, R. 1997. How to progress a database. *Artificial Intelligence* 92(1-2):131–167.

Reiter, R. 2001. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Vassos, S., and Levesque, H. J. 2013. How to progress a database III. *Artificial Intelligence* 195:203–221.

Vassos, S., and Patrizi, F. 2013. A Classification of First-Order Progressable Action Theories in Situation Calculus. In Rossi, F., ed., *Proc. of IJCAI'13*.

An Abductive Reasoning Approach to the Belief-Bias Effect

Luís Moniz Pereira[*1]
Emmanuelle-Anna Dietz[†1,2]
Steffen Hölldobler[†2]

[1]Centro de Inteligência Artificial (CENTRIA), Departamento de Informática
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
[2]International Center for Computational Logic (ICCL), TU Dresden, D-01062 Dresden, Germany

Abstract

The tendency to accept or reject arguments based on own beliefs or prior knowledge rather than on the reasoning process is called the belief-bias effect. A psychological syllogistic reasoning task shows this phenomenon, wherein participants were asked whether they accept or reject a given syllogism. We discuss one case which is commonly assumed to be believable but not logically valid. By introducing abnormalities, abduction and background knowledge, we model this case under the weak completion semantics. Our formalization reveals new questions about observations and their explanations which might include some relevant prior abductive contextual information concerning some side-effect. Inspection points, introduced by Pereira and Pinto, allow us to express these definitions syntactically and intertwine them into an operational semantics.

Keywords: Abductive Reasoning; Side-effects; Inspection Points; Human Reasoning; Belief-Bias Effect

1 Introduction

Whenever discovering abductive explanations for some given primary observation, one may wish to check too whether some other given additional secondary observations are logical consequences of the abductive explanations found for the primary observation. The primary explanations may be used to explain the secondary observations, whereas the secondary explanations cannot be used to explain the primary observations. We show this type of reasoning requires the characterization of a new abduction concept and mechanism, that of *contextual abduction*. We formalize contextual abduction and employ it to understand and justify the belief-bias effect in human reasoning by addressing syllogisms.

Evans, Barston, and Pollard (1983) made a psychological study showing possibly conflicting processes in human reasoning. Participants were presented syllogisms and had to decide whether they were logically valid. Consider S_{add}:

PREMISE1 *No addictive things are inexpensive (not costly).*
PREMISE2 *Some cigarettes are inexpensive (not costly).*
CONCLUSION *Therefore, some addictive things are not cigarettes.*

Although the conclusion does not necessarily follow from the premises, participants assumed the syllogism to be logically valid. Evans, Barston, and Pollard (1983) concluded that they have been unduly influenced by their own beliefs. In the following, we explain the belief-bias effect by abductive reasoning and its side-effects.

2 Preliminaries

We restrict ourselves to datalog (first-order) programs, ie. the set of terms consists only of constants and variables. A logic program $\mathcal{P}$ is a finite set of clauses of the form

$$A \leftarrow A_1 \wedge \ldots \wedge A_n \wedge \overline{B_1} \wedge \ldots \wedge \overline{B_m},$$

where A and A_i, $1 \leq i \leq n$, are atoms and $\overline{B_j}$, $1 \leq j \leq m$, are negated atoms. A is the head and $A_1 \wedge \ldots \wedge A_n \wedge \overline{B_1} \wedge \ldots \wedge \overline{B_m}$ is the body of the clause. We abbreviate the body of a clause by $body$. $A \leftarrow \top$ and $A \leftarrow \bot$ are special cases of clauses denoting positive and negative facts, respectively. If an argument is written with an upper case letter, it is a variable; otherwise it is a constant. In the sequel, we assume $\mathcal{P}$ to be ground, containing all the ground instances of its clauses. The set of all atoms occurring in $\mathcal{P}$ is $\mathsf{atoms}(\mathcal{P})$. An atom is undefined in $\mathcal{P}$ if it is not the head of some clause in $\mathcal{P}$ and the corresponding set of these atoms is $\mathsf{udef}(\mathcal{P})$.

2.1 Three-Valued Łukasiewicz Semantics

Table 1 shows the three-valued Łukasiewicz (1920) semantics. Interpretations are represented by pairs $\langle I^\top, I^\bot \rangle$, where

$$I^\top = \{A \in B_\mathcal{P} \mid A \text{ is mapped to } \top\},$$
$$I^\bot = \{A \in B_\mathcal{P} \mid A \text{ is mapped to } \bot\}.$$

$\mathcal{B}_\mathcal{P}$ is the Herbrand base wrt $\mathcal{P}$. A model of $\mathcal{P}$ is an interpretation which maps each clause occurring in $\mathcal{P}$ to $\top$.

2.2 Weak Completion Semantics

Weak completion semantics (Hölldobler and Kencana Ramli 2009a) seems to adequately model Byrne's (1989) suppression task (Dietz, Hölldobler, and Ragni 2012) and Wason's (1968) selection task (Dietz, Hölldobler, and Ragni 2013). Consider the following transformation for $\mathcal{P}$:

1. Replace all clauses with the same head $A \leftarrow body_1, \ldots, A \leftarrow body_n$ by $A \leftarrow body_1 \vee \ldots \vee body_n$.
2. For all atoms A, if $A \in \mathsf{udef}(\mathcal{P})$ then add $A \leftarrow \bot$.
3. Replace all occurrences of $\leftarrow$ by $\leftrightarrow$.

*lmp@fct.unl.pt
†{dietz,sh}@iccl.tu-dresden.de
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

F	$\overline{F}$
$\top$	$\bot$
$\bot$	$\top$
U	U

$\wedge$	$\top$	U	$\bot$
$\top$	$\top$	U	$\bot$
U	U	U	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$

$\vee$	$\top$	U	$\bot$
$\top$	$\top$	$\top$	$\top$
U	$\top$	U	U
$\bot$	$\top$	U	$\bot$

$\leftarrow_{Ł}$	$\top$	U	$\bot$
$\top$	$\top$	$\top$	$\top$
U	U	$\top$	$\top$
$\bot$	$\bot$	U	$\top$

$\leftrightarrow_{Ł}$	$\top$	U	$\bot$
$\top$	$\top$	U	$\bot$
U	U	$\top$	U
$\bot$	$\bot$	U	$\top$

Table 1: $\top$, $\bot$, and U denote *true*, *false*, and *unknown*, respectively.

The resulting set of equivalences is called the *completion* of $\mathcal{P}$ (Clark 1978). If Step 2 is omitted, then the resulting set is called the *weak completion* of $\mathcal{P}$ ($\mathsf{wc}\,\mathcal{P}$).

Hölldobler and Kencana Ramli (2009b) showed that each weakly completed program admits a least model. Stenning and van Lambalgen (2008) devised an operator which has been generalized for first-order programs by Hölldobler and Kencana Ramli (2009a): Let I be an interpretation in $\Phi_{\mathcal{P}}(I) = \langle J^{\top}, J^{\bot}\rangle$, where
$J^{\top} = \{A \mid$ there exists $A \leftarrow body \in \mathcal{P}$ with $I(body) = \top\}$,
$J^{\bot} = \{A \mid$ there exists $A \leftarrow body \in \mathcal{P}$ and
for all $A \leftarrow body \in \mathcal{P}$ we find $I(body) = \bot\}$.
As shown by Hölldobler and Kencana Ramli, the least fixed point of $\Phi_{\mathcal{P}}$ is identical to the least model of the weak completion of $\mathcal{P}$ ($\mathsf{lm}_{Ł}\mathsf{wc}\,\mathcal{P}$). Starting with the empty interpretation $I = \langle \emptyset, \emptyset\rangle$, $\mathsf{lm}_{Ł}\mathsf{wc}\,\mathcal{P}$ can be computed by iterating $\Phi_{\mathcal{P}}$.[1]

3 Reasoning in an Appropriate Logical Form

We follow the approach of Stenning and van Lambalgen (2005; 2008), who proposed to model human reasoning by a two step process: Firstly, human reasoning is to be modeled by setting up an *appropriate representation* and, secondly, *reasoning* wrt this representation.

3.1 Integrity Constraints

One should observe that the first premise *no addictive things are inexpensive* is equivalent to *if something is inexpensive, then it is not addictive*. As weak completion semantics does not allow negative heads in clauses, for every negative conclusion $\overline{p(X)}$ we need to introduce an auxiliary formula $p'(X)$ together with the clause $p(X) \leftarrow \overline{p'(X)}$. We obtain a (preliminary) representation of the first premise of S_{add}:

$$\begin{aligned} add'(X) &\leftarrow inex(X), \\ add(X) &\leftarrow \overline{add'(X)}, \end{aligned}$$

where $add(X)$, $add'(X)$, and $inex(X)$ denote that X is addictive, not addictive, and inexpensive, respectively.

With the introduction of these auxiliary atoms, the need for integrity constraints arises. An interpretation which maps both, $add(a)$ and $add'(a)$, to true for some constant a should not be a model. This can be represented by a specific type of *integrity constraint* (IC), a logical formula of the form $\bot \leftarrow IC\_body$, where IC_body is a conjunction of atoms. For our running example we obtain the integrity constraint:

$$\bot \leftarrow add(X) \wedge add(X)'.$$

However, the Φ operator does not consider clauses of this form. We need to apply a two step approach: firstly, compute the least model and secondly, verify whether it respects the IC, where an interpretation I respects IC iff for each $\bot \leftarrow IC\_body$ in IC, we find that $I(IC\_body) \in \{\bot, \mathsf{U}\}$.[2]

For the following examples, whenever there is a $p(X)$ and its $p(X)'$ counterpart, we implicitly assume its corresponding integrity constraint, IC_p: $\bot \leftarrow p(X) \wedge p(X)'$.

3.2 Abnormalities

Stenning and van Lambalgen proposed the idea to implement conditionals by default licenses for implications. This can be achieved by adding an *abnormality predicate* to the antecedent of the implication and initially assuming that the abnormality predicate is false. Following this idea, the initial program representing PREMISE1 in S_{add} is extended to:

If something is inexpensive and not abnormal,
then it is not addictive.
Nothing (by default) is abnormal (regarding PREMISE1*).*

3.3 Background Knowledge and Belief-Bias

A direct first-order representation of PREMISE2 is

There exists a cigarette which is inexpensive. (1)

which includes two pieces of information. Firstly, a more specific one, namely that there exists something, say a, which is a cigarette. Secondly, as Evans et al. have discussed, it contains background knowledge or belief that humans seem to have, which, in this context, we assume to be

Cigarettes are inexpensive (2)

compared to other addictive things, which implies (1) and biases the reasoning towards a representation. This belief bias together with the idea to represent conditionals by default licenses for implication leads to

If something is a cigarette and not abnormal,
then it is inexpensive.
Nothing (by default) is abnormal (regarding PREMISE2*).*

Additionally, we have a second piece of background knowledge, viz. it is commonly known that

Cigarettes are addictive. (3)

This belief together with (2), leads to

If something is a cigarette,
then it is abnormal (regarding PREMISE1*).*

The information contained in the two premises of S_{add} is encoded as the program $\mathcal{P}_{add}$ as follows:

$$\begin{aligned} cig(a) &\leftarrow \top, \\ add'(X) &\leftarrow inex(X) \wedge \overline{ab_1(X)}, \\ add(X) &\leftarrow \overline{add'(X)}, \\ ab_1(X) &\leftarrow \bot, \\ ab_1(X) &\leftarrow cig(X), \\ inex(X) &\leftarrow cig(X) \wedge \overline{ab_2(X)}, \\ ab_2(X) &\leftarrow \bot. \end{aligned}$$

[1] Dietz, Hölldobler, and Wernhard (2014) showed that the weak completion semantics corresponds to the well-founded semantics (Van Gelder, Ross, and Schlipf 1991) for tight logic programs.

[2] This corresponds to the definition applied for the well-founded semantics in (Pereira, Aparício, and Alferes 1991).

Finally, we impose $IC_{add} : \bot \leftarrow add(X) \wedge add'(X)$.

4 Abductive Framework

Following Kakas, Kowalski, and Toni (1993) we consider an *abductive framework* consisting of a program $\mathcal{P}$ as knowledge base, a collection of atoms $\mathcal{A}$ of abducibles syntactically represented by the set of the (positive and negative) facts for each undefined ground atom in $\mathcal{P}$, a set of integrity constraints IC, and the logical consequence relation $\models_{Ł}^{lmwc}$, where $\mathcal{P} \models_{Ł}^{lmwc} F$ if and only if $\mathsf{lm}_{Ł}\mathsf{wc}\,\mathcal{P}(F) = \top$ for a formula F. An *observation* is a non-empty set of literals. An explanation $\mathcal{E}$ is a consistent subset of $\mathcal{A}$.

Definition 1 *Let* $\langle \mathcal{P}, \mathcal{A}, IC, \models_{Ł}^{lmwc} \rangle$ be an abductive framework, $\mathcal{O}$ an observation, $\mathcal{E}$ an explanation, IC a set of integrity constraints, and $\models_{Ł}^{lmwc}$ the consequence relation defined for all formulas F.

- $\mathcal{O}$ is **explained by** $\mathcal{E}$ **given** $\mathcal{P}$ **and** IC iff $\mathcal{P} \cup \mathcal{E} \models_{Ł}^{lmwc} O$, where $\mathcal{P} \not\models_{Ł}^{lmwc} \mathcal{O}$ and $\mathsf{lm}_{Ł}\mathsf{wc}\,(\mathcal{P} \cup \mathcal{E})$ respects IC.
- $\mathcal{O}$ is **explained given** $\mathcal{P}$ **and** IC iff there exists an $\mathcal{E}$ such that $\mathcal{O}$ is explained by $\mathcal{E}$ given $\mathcal{P}$ and IC.

We distinguish between two forms of reasoning:

- F **follows skeptically from** $\mathcal{P}$**,** IC **and** $\mathcal{O}$ iff $\mathcal{O}$ can be explained given $\mathcal{P}$ and IC, and for all minimal (or otherwise preferred) explanations $\mathcal{E}$ we find that $\mathcal{P} \cup \mathcal{E} \models_{Ł}^{lmwc} \mathcal{O}$.
- F **follows credulously from** $\mathcal{P}$**,** IC **and** $\mathcal{O}$ iff there exists a minimal (or otherwise preferred) explanation $\mathcal{E}$ such that $\mathcal{P} \cup \mathcal{E} \models_{Ł}^{lmwc} \mathcal{O}$.

An important extension of abduction pertains to the issue that, whenever discovering explanations for some given primary observation, one may wish to check too whether some other given additional secondary observations are true, being a logical consequence of the abductive explanations found for the primary observation.[3]

5 Contextual Abductive Explanations

Let's consider again $\mathcal{P}_{add}$, where its weak completion consists of the equivalences:

$$\begin{array}{lcl} cig(a) & \leftrightarrow & \top, \\ add'(a) & \leftrightarrow & inex(a) \wedge \overline{ab_1(a)}, \\ add(a) & \leftrightarrow & \overline{add'(a)}, \\ ab_1(a) & \leftrightarrow & cig(a), \\ inex(a) & \leftrightarrow & cig(a) \wedge \overline{ab_2(a)}, \\ ab_2(a) & \leftrightarrow & \bot. \end{array}$$

Its least model, $\mathsf{lm}_{Ł}\mathsf{wc}\,\mathcal{P}_{add}$, is

$\langle \{cig(a), inex(a), add(a), ab_1(a)\}, \{add'(a), ab_2(a)\}\rangle$,

from which we cannot derive the CONCLUSION in S_{add}. Obviously, the CONCLUSION is something about an object which is not a. The first part of this conclusion is an observation, let's say about b: $\mathcal{O}_{add(b)} = \{add(b)\}$, which we need to explain as described in Section 4. The set of abducibles is: $\mathcal{A}_{(\mathcal{P}_{add} \cup \mathcal{O}_{add(b)})} = \{cig(b) \leftarrow \top, cig(b) \leftarrow \bot\}$.

We have the following minimal explanations for $\mathcal{O}_{add(b)}$:

$\mathcal{E}_{\overline{cig(b)}} = \{cig(b) \leftarrow \bot\}$ and $\mathcal{E}_{cig(b)} = \{cig(b) \leftarrow \top\}$.

[3] Reuse of contextual abductions, by resorting to an implementation of tabled abduction for well-founded semantics, is reported in (Saptawijaya and Pereira 2013).

The corresponding least models of the weak completion are:

$\mathsf{lm}_{Ł}\mathsf{wc}\,(\mathcal{P}_{add} \cup \mathcal{E}_{\overline{cig(b)}}) = \langle \{add(b)\}, \{cig(b), inex(b), ...\}\rangle$,

$\mathsf{lm}_{Ł}\mathsf{wc}\,(\mathcal{P}_{add} \cup \mathcal{E}_{cig(b)}) = \langle \{add(b), cig(b), inex(b), ...\}, ...\rangle$.

Under credulous reasoning we conclude, given explanation $\mathcal{E}_{\overline{cig(b)}}$ that the CONCLUSION of S_{add} is true, as there exists something addictive which is not a cigarette.

However, $\mathcal{O}_{add(b)}$ can also be explained by $\mathcal{E}_{cig(b)}$. We would prefer to reflect that the first explanation describes the usual case and the second describes the exceptional one.

6 Inspection Points

In the example we can easily identify from the context what is the exceptional case. But this is not stated in our program yet. For this purpose we apply inspection points, originally presented in (Pereira and Pinto 2011). We now discuss this approach and show how inspection points can be modeled in our example, concerning its abducibles. Given an atom A, we introduce the following two reserved (meta-)predicates:

$$\mathsf{inspect}(A) \qquad \text{and} \qquad \mathsf{inspect}_{\neg}(A),$$

that are special cases of abducibles. They differ from usual abducibles in the way that they can only be abduced whenever A or $\neg A$ have been abduced somewhere else already: an abductive solution or explanation $\mathcal{E}$ is only valid when for each $\mathsf{inspect}(A)$ it contains, respectively $\mathsf{inspect}_{\neg}(A)$, it also contains a corresponding A, respectively $\neg A$.

One should observe that for a treatment of inspection points for all literals in a program and not just the abducible ones, we would simply need to adopt the program transformation technique in (Pereira and Pinto 2011), which recursively relays inspection of non-terminal literals to the base inspection of terminals.

6.1 Usual Contextual Abduction

Let us modify our previous example with inspection points. The new program, $\mathcal{P}_{add}^{insp}$, has a new ab_1-clause:

$$\begin{array}{l} (\mathcal{P}_{add} \setminus \{ab_1(X) \leftarrow cig(X)\}) \cup \\ \qquad \{ab_1(X) \leftarrow inspect(cig(X))\}. \end{array}$$

Suppose again that b is addictive, i.e. $\mathcal{O}_{add(b)} = \{add(b)\}$. As $cig(b)$ is unknown, $inspect(cig(b))$ is false, and $ab_1(b)$ is false rather than unknown that is, its falsity is obtained because nothing is known about $cig(b)$, and there exists no observation to be explained and which abduces $cig(b)$. The only minimal explanation for $\mathcal{O}_{add(b)}$ is generated by $inex(b)$ being false, achieved by $\mathsf{lm}_{Ł}\mathsf{wc}\,(\mathcal{P}_{add}^{insp} \cup \mathcal{E}_{\overline{cig(b)}})$:

$\langle \{add(b)\}, \{add'(b), cig(b), ab_1(b), inex(b), ab_2(b)\}\rangle$.

Now, even under skeptical reasoning, we conclude that there exists an addictive thing which is not a cigarette.

6.2 Contextual Side-effects

Let us extend S_{add} with the conditional:

If something is sold in the streets and there is no reason to believe it is cigarettes, then it is illegal.

Accordingly, our new program $\mathcal{P}_{ill}^{insp}$ is defined as:

$\mathcal{P}_{add}^{insp} \cup \{ill(X) \leftarrow streets(X) \wedge inspect_{\neg}(cig(X))\}$.

We only conclude that something is illegal when we have abduced somewhere else that it is not a cigarette. Note, that we can abduce $\mathcal{E}_{streets(b)} = \{streets(b) \leftarrow \top\}$ to

partly explain $\mathcal{O}_{ill(b)} = \{illegal(b)\}$. Assume additionally, we observe $\mathcal{O}_{add(b)}$ which is explained by $\mathcal{E}_{\overline{cig(b)}}$. The $\mathsf{lm}_{\text{Ł}}\mathsf{wc}\,(\mathcal{P}^{insp}_{ill} \cup \mathcal{E}_{\overline{cig(b)}} \cup \mathcal{E}_{streets(b)})$ is:

$$\langle\{add(b), streets(b), illegal(b), inspect_{\neg}(cig(b))\}, \{add'(b), cig(b), ab_1(b), inex(b), ab_2(b)\}\rangle$$

thus, $illegal(b)$ is true. We define side-effects as follows:

Definition 2 *Given a background knowledge $\mathcal{P}$ and two observations $\mathcal{O}_1$ and $\mathcal{O}_2$, both explained given $\mathcal{P}$ and IC.*

- *$\mathcal{O}_2$ is a **necessary contextual side-effect of** $\mathcal{O}_1$ **given** $\mathcal{P}$ **and** IC iff for all explanations $\mathcal{E}_1$ for $\mathcal{O}_1$, there exists some explanation $\mathcal{E}_2$ for $\mathcal{O}_2$ containing abducibles of the form $inspect(A)$, or respectively $inspect_{\neg}(A)$, and for all such abducibles for which abducible A, respectively $\neg A$, is not in $\mathcal{E}_2$, and at least one exists, then it is in $\mathcal{E}_1$.*
- *$\mathcal{O}_2$ is a **strict necessary contextual side-effect of** $\mathcal{O}_1$ **given** $\mathcal{P}$ **and** IC iff $\mathcal{O}_2$ is a necessary contextual side-effect of $\mathcal{O}_1$ given $\mathcal{P}$ and IC and all explanations $\mathcal{E}_1$ for $\mathcal{O}_1$ are also all the explanations for $\mathcal{O}_2$.*
- *$\mathcal{O}2$ is a **possible contextual side-effect of explained** $\mathcal{O}_1$ **given** $\mathcal{P}$ **and** IC iff there exists an explanation $\mathcal{E}_1$ for $\mathcal{O}_1$ and an explanation $\mathcal{E}_2$ for $\mathcal{O}_2$ containing abducibles of the form $inspect(A)$, or respectively $inspect_{\neg}(A)$, and for all such abducibles for which abducible A, respectively $\neg A$, is not in $\mathcal{E}_2$, at least one exists, then it is in $\mathcal{E}_1$.*
- *$\mathcal{O}_2$ is a **strict possible contextual side-effect of** $\mathcal{O}_1$ **given** $\mathcal{P}$ **and** IC iff $\mathcal{O}_2$ is a possible contextual side-effect of $\mathcal{O}_1$ given $\mathcal{P}$ and IC and there exists an explanation $\mathcal{E}_1$ for $\mathcal{O}_1$ such that $\mathcal{E}_1$ is also an explanation for $\mathcal{O}_2$.*

The idea behind necessary contextual side-effects, is, that every explanation $\mathcal{E}_1$ for $\mathcal{O}_1$ affords us with one *complete* explanation under which some *incomplete* explanation $\mathcal{E}_2$ for $\mathcal{O}_2$ is necessarily completed. The idea behind possible contextual side-effects, is, that there is at least one explanation $\mathcal{E}_1$ for $\mathcal{O}_1$ which affords us with one *complete* explanation under which some *incomplete* explanation $\mathcal{E}_2$ for $\mathcal{O}_2$ can be completed. Summing up, $\mathcal{O}_{add(b)}$ is a strict necessary contextual side-effect of explained $\mathcal{O}_{inex(b)}$ given $\mathcal{P}^{insp}_{add}$ and IC_{add} and $\mathcal{O}_{ill(b)}$ is a possible contextual side-effect of $\mathcal{O}_{add(b)}$ given $\mathcal{P}^{insp}_{ill}$ and IC_{add} but not vice versa.

7 Conclusion

Taking weak completion semantics as starting point, we showed with a running example the need for possible extensions in abductive reasoning. Introducing the concept of inspection points in our framework by applying reserved (meta-)predication for all abductive ground atoms, gives us the possibility to distinguish between different kinds of abducibles in a logic program. This distinction allows to straightforwardly implement the concepts of contextual side-effects. Torasso et al. (1991) do abduction with the completion semantics and do not address side-effects.

8 Acknowledgments

We thank Pierangelo DellAcqua for comments and are grateful to the 3 reviewers. The stay of the 2nd author at UNL was supported by DAAD within the FITweltweit program.

References

Byrne, R. M. J. 1989. Suppressing valid inferences with conditionals. *Cognition* 31:61–83.

Clark, K. L. 1978. Negation as failure. In Gallaire, H., and Minker, J., eds., *Logic and Data Bases*, volume 1. New York, NY: Plenum Press. 293–322.

Dietz, E.-A.; Hölldobler, S.; and Ragni, M. 2012. A computational logic approach to the suppression task. In Miyake, N.; Peebles, D.; and Cooper, R. P., eds., *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, 1500–1505. Austin, TX: Cognitive Science Society.

Dietz, E.-A.; Hölldobler, S.; and Ragni, M. 2013. A computational logic approach to the abstract and the social case of the selection task. In *Proceedings of the 11th International Symposium on Logical Formalizations of Commonsense Reasoning*.

Dietz, E.-A.; Hölldobler, S.; and Wernhard, C. 2014. Modeling the suppression task under weak completion and well-founded semantics. *J. of Applied Non-Classical Logics*. accepted.

Evans, J.; Barston, J. L.; and Pollard, P. 1983. On the conflict between logic and belief in syllogistic reasoning. *Memory & Cognition* 11(3):295–306.

Hölldobler, S., and Kencana Ramli, C. D. 2009a. Logic programs under three-valued Łukasiewicz semantics. In Hill, P. M., and Warren, D. S., eds., *Logic Programming, 25th International Conference, ICLP 2009*, volume 5649 of *LNCS*, 464–478. Heidelberg: Springer.

Hölldobler, S., and Kencana Ramli, C. D. 2009b. Logics and networks for human reasoning. In Alippi, C.; Polycarpou, M. M.; Panayiotou, C. G.; and Ellinas, G., eds., *International Conference on Artificial Neural Networks, ICANN 2009, Part II*, volume 5769 of *LNCS*, 85–94. Heidelberg: Springer.

Kakas, A. C.; Kowalski, R. A.; and Toni, F. 1993. Abductive logic programming. *Journal of Logic and Computation* 2(6):719–770.

Łukasiewicz, J. 1920. O logice trójwartościowej. *Ruch Filozoficzny* 5:169–171. English translation: On three-valued logic. In: Łukasiewicz J. and Borkowski L. (ed.). (1990). *Selected Works*, Amsterdam: North Holland, pp. 87–88.

Pereira, L. M., and Pinto, A. M. 2011. Inspecting side-effects of abduction in logic programs. In Balduccini, M., and Son, T. C., eds., *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays in honour of Michael Gelfond*, volume 6565 of *LNAI*. Springer. 148–163.

Pereira, L. M.; Aparício, J. N.; and Alferes, J. J. 1991. Hypothetical reasoning with well founded semantics. In Mayoh, B., ed., *Scandinavian Conference on Artificial Intelligence: Proc. of the SCAI'91*. Amsterdam: IOS Press. 289–300.

Saptawijaya, A., and Pereira, L. M. 2013. Tabled abduction in logic programs. *Theory and Practice of Logic Programming* 13(4-5-Online-Supplement).

Stenning, K., and van Lambalgen, M. 2005. Semantic interpretation as computation in nonmonotonic logic: The real meaning of the suppression task. *Cognitive Science* 6(29):916–960.

Stenning, K., and van Lambalgen, M. 2008. *Human Reasoning and Cognitive Science*. A Bradford Book. Cambridge, MA: MIT Press.

Torasso, P.; Console, L.; Portinale, L.; and Dupré, D. T. 1991. On the relationship between abduction and deduction. *Journal of Logic and Computation* 1(5):661–690.

Van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM* 38(3):619–649.

Wason, P. 1968. Reasoning about a rule. *Quarterly Journal of Experimental Psychology* 20(3):273–281.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Minimal Change in AGM Revision for Non-Classical Logics

Marcio M. Ribeiro and Renata Wassermann
University of São Paulo - Brazil
marciomr@usp.br *renata@ime.usp.br*

Abstract

In this paper, we address the problem of applying AGM-style belief revision to non-classical logics. We discuss the idea of minimal change in revision and show that for non-classical logics, some sort of minimality postulate has to be explicitly introduced. We also present two constructions for revision which satisfy the AGM postulates and prove the representation theorems including minimality postulates.

Introduction

Belief revision is a sub-field of knowledge representation that studies the dynamics of epistemic states. Alchourrón, Gärdenfors and Makinson (1985) set the basis of a tradition in this field which became known as the *AGM paradigm* due to the authors initials.

Following the AGM tradition, the *epistemic state* of a rational agent is represented as a set of sentences closed under a consequence operator ($K = Cn(K)$) called *belief set*. Three main operations over belief sets are studied: *expansion*, *revision* and *contraction*. The former is the simplest and is defined as $K + \alpha = Cn(K \cup \{\alpha\})$. We can extend it to deal with expansion by a set of formulas in a natural way by defining $K + A = Cn(K \cup A)$.

Contraction ($-$) is the operation used when one intends to open his mind over some proposition and *revision* ($*$) is the operation used when one intends to consistently incorporate a new piece of knowledge to his belief set. Both are typically defined via sets of rationality postulates. Furthermore, revision can be defined in terms of the two other operations using the following equation, known as the *Levi identity*:

$$K * \alpha = (K - \neg\alpha) + \alpha$$

This identitiy assumes the underlying logic to be closed under negation. However, this is not the case for many interesting logics which are used in practice. In the last decade, there was an effort to adapt the AGM paradigm to non-classical logics of interest in Artificial Intelligence, notably Description Logics (Flouris, Plexousakis, and Antoniou 2005) and Horn logics (Delgrande 2008). The use of the Levi identity in order to define constructions for revision has induced the common practice of concentrating on contraction operators and not revision. It should be noted, however, that on early AI work, the focus was on revision, and not on contraction:

> "To choose their actions, reasoning programs must be able to make assumptions and subsequently revise their beliefs when discoveries contradict these assumptions."(Doyle 1979)

The main goal of this work is to generalize AGM revision for a wide class of logics. Our first attempt in this direction, presented in (Ribeiro and Wassermann 2009; 2010), was applicable only for a certain class of logics, which does not include most Description Logics (DLs) and Horn logic.

After presenting the motivation and formal preliminaries, we investigate the role of minimality postulates for AGM revision in non-classical logics, the main result being that, for non-classical logics, AGM postulates for revision fail to guarantee even the weakest form of minimality. We argue that some minimality postulates such as relevance or core-retainment must be added to the list of rationality postulates for AGM revision in non-classical logics.

Before presenting the main results, we introduce some new postulates and argue that these are the adequate postulates for non-classical AGM belief revision.

Two general approaches for belief revision in non-classical logics that satisfy some minimality postulate are presented. They are general in the sense that they are applicable to a wide class of logics, but also in the sense that they do not take advantage of any singularity of a specific logic. We provide representation results for both approaches.

Finally, we briefly comment on related work, present some conclusions and point towards future work.

Formal Preliminaries

For the purposes of this article, a logic is a pair $\langle \mathscr{L}, Cn \rangle$ such that $\mathscr{L}$ is any enumerable set called *language* whose elements are called *sentences* and $Cn : 2^{\mathscr{L}} \rightarrow 2^{\mathscr{L}}$ is a function called *consequence operator* that is considered to be Tarskian i.e. for every $A, B \in 2^{\mathscr{L}}$ if $A \subseteq B$ then $Cn(A) \subseteq Cn(B)$ (*monotonicity*), $A \subseteq Cn(A)$ (*inclusion*) and $Cn(A) = Cn(Cn(A))$ (*idempotence*).

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Throughout this paper, sentences will be denoted by lower case Greek letters (α, β etc), sets of sentences will be denoted by upper case letters (A, B etc) and the letter K will be reserved for *belief sets*.

A logic is called *supra-classical* if it contains every valid inference of Classical Propositional Logic (CPL); it is called *compact* if for every $A \cup \{\alpha\} \in 2^{\mathscr{L}}$ it follows that if $\alpha \in Cn(A)$ then there is a finite subset A_F of A such that $\alpha \in Cn(A_F)$. Most work in belief revision literature assumes the underling logic $\langle \mathscr{L}, Cn \rangle$ to satisfy certain properties: the language $\mathscr{L}$ is supposed to be closed under the standard connectives $\wedge$, $\vee$, $\rightarrow$ and $\neg$; the consequence operator is supposed to be compact, supra-classical and to satisfy the deduction theorem.[1] This set of properties are called the *AGM assumptions*.

Minimal Change in AGM Revision

In this section, we discuss AGM revision and the concept of minimal change.

In their seminal work, Alchourrón, Gärdenfors and Makinson (1985) proposed the following set of postulates for revision:

closure: $K * \alpha = Cn(K * \alpha)$.

success: $\alpha \in K * \alpha$.

inclusion: $K * \alpha \subseteq K + \alpha$

vacuity: If $K + \alpha \neq \mathscr{L}$ then $K + \alpha \subseteq K * \alpha$.

consistency: If $K * \alpha = \mathscr{L}$ then $Cn(\alpha) = \mathscr{L}$.

extensionality: If $Cn(\alpha) = Cn(\beta)$ then $K * \alpha = K * \beta$.

Examining these postulates one may notice the lack of a *minimality postulate* i.e. a postulate that states the principle of informational economy: "one should not give up beyond necessity" (Harman 1986). This principle, certainly desirable for revision, seems to be missing in the AGM paradigm.

Consider the following trivial construction for revision, where in case of inconsistency, one throws away everything but the new input:

$$K * \alpha = \begin{cases} Cn(\alpha) \textit{ if } K + \{\alpha\} = \mathscr{L} \\ K + \alpha \textit{ otherwise} \end{cases}$$

Alchourrón and Makinson (1982) have already argued that the AGM postulates for revision are not enough to prevent this trivial revision. Here, we get back to the original AGM notion of minimal change. In AGM contraction, minimality is given by the *recovery* postulate:

(recovery) $K \subseteq (K - \alpha) + \alpha$

This postulate does not say anything about the size of the resulting set, or that it should be maximal with respect to inclusion. It only states that whatever is given up in a contraction should be recoverable by expanding with the same formula. The AGM postulates for contraction (including recovery) allow for the (full) meet contraction (Alchourrón and Makinson 1982), which by using the Levi identity, yields the trivial revision.

[1] $\beta \in Cn(A \cup \{\alpha\})$ if and only if $\alpha \rightarrow \beta \in Cn(A)$.

On the context of contraction operations, Makinson and Hansson staged a significant debate on the importance of a minimality postulate. Makinson (Makinson 1987) argued that the postulate of recovery has several odd properties and should be removed from the list of AGM postulates for contraction. He proposed an alternative operation called *withdrawal* that satisfies all AGM postulates for contraction but recovery. Hansson (Hansson 1991), on the other hand, argued that withdrawal suffers from the lack of a minimality postulate. That is, giving up recovery altogether would mean to abandon the desirable principle of informational economy for contraction. Agreeing with the disadvantages of recovery, Hansson proposed to exchange it with some new postulate that stated minimality directly. Two postulates were proposed, relevance and core-retainment, and both were proved to be equivalent to recovery in classical propositional logic. In (Ribeiro et al. 2013) the authors reassembled this controversy and showed that this equivalence does not hold in general (e.g. it fails for Horn logic and most DLs). Hence, in general, minimality of change has to be guaranteed by either relevance or core-retainment, but not recovery. Actually, Flouris (2006) has shown that several logics are not *AGM-compliant*, i.e., they do not admit any contraction operation satisfying recovery together with the other AGM contraction postulates.

In the belief bases literature (Hansson 1991), one can find two postulates for minimality in revision analogous to the ones for contraction. Because of the strong analogy they have the same names as the their contraction counterpart:

(core-retainment) If $\beta \in K \setminus K * \alpha$ then there is $X \subseteq K$ such that $X + \alpha \neq \mathscr{L}$ and $X + \{\alpha, \beta\} = \mathscr{L}$.

(relevance) If $\beta \in K \setminus K * \alpha$ then there is X such that $K \cap (K * \alpha) \subseteq X \subseteq K$ and $X + \alpha \neq \mathscr{L}$, but $X + \{\alpha, \beta\} = \mathscr{L}$.

It is trivial to see that core-retainment follows from relevance, but the converse is not necessarily the case.

Although these minimality postulates can be found in the belief base literature, in AGM revision they are not mentioned. In fact, AGM revision does not state any postulate similar to any of these postulates of minimality explicitly. However, we can show that for logics satisfying the AGM assumptions, both relevance and core-retainment follow from the other AGM postulates.

Theorem 1. *If the underlying logic satisfies the AGM assumptions (i.e. be supra-classical, compact and satisfy the deduction theorem), relevance and core-retainment follow from the AGM postulates for revision.*

Corollary 2. *For logics satisfying the AGM assumptions, relevance does not prevent the trivial revision.*

Since most of the belief revision literature deals with supra-classical logics, formulating relevance explicitly does not bring anything new. However, for many non-classical logics this is not the case:

Theorem 3. *In general neither core-retainment nor relevance follow from the AGM postulates for revision.*

Following Hansson (Hansson 1991) and Harmann (Harman 1986) we argue that the principle of informational economy is desirable for a revision.

Surprisingly enough, the proof above shows that adding core-retainment to the AGM postulates blocks the trivial revision for some non-classical logics.

Besides introducing some postulate for minimality, we will introduce two postulates:

(strong inclusion) $K * \alpha \subseteq (K \cap K * \alpha) + \alpha$

(uniformity) If for all $X \subseteq K$, $X + \alpha = \mathscr{L}$ iff $X + \beta = \mathscr{L}$ then $K \cap K * \alpha = K \cap K * \beta$

The former is a stronger version of inclusion related to Harper identity and the latter is for syntactic independence. In the presence of these postulates vacuity, extensionality and inclusion became superfluous. Hence, we suggest the following set of postulates for revision in non-classical logics: *success*, *closure*, *strong inclusion*, *consistency*, *uniformity* and either *relevance* or *core-retainment*. Results from next section corroborate this choice of postulates.

Revision Constructions in Non-classical Logics

Following the AGM tradition, we will present in this section two constructions for revision. The first one is based on remainder sets, maximal subsets of the original belief set that together with the input are inconsistent. The second construction is based on the idea of kernels, minimal subsets of the original belief set that are inconsistent with the input. We will prove representation theorems for both constructions.

Partial Meet Revision

This section presents a construction for revision called *negation free partial meet revision*. It is an adaptation of partial meet revision (Alchourrón, Gärdenfors, and Makinson 1985) for logics that are not closed under negation and it was first introduced in (Ribeiro and Wassermann 2009), but with severe limitation on applicability. It will be reintroduced here together with a new, more general, representation theorem.

A *negation free remainder set* $K \downarrow \alpha$ is defined as the set of all maximal subsets of K that are consistent with the input α.[2] Formally:

Definition 4 (negation free remainder set). $X \in K \downarrow \alpha$ *iff:*

1. $X \subseteq K$
2. $X + \alpha \neq \mathscr{L}$
3. *If* $X \subset X' \subseteq K$ *then* $X' + \alpha = \mathscr{L}$

A *selection function* γ selects a non-empty subset of $K \downarrow \alpha$ if possible i.e. if $K \downarrow \alpha \neq \emptyset$. Otherwise it returns $\{K\}$.

Any selection function induces the following revision operation called *negation free partial meet revision*:

$$K *_\gamma \alpha = \bigcap \gamma(K \downarrow \alpha) + \alpha$$

For any compact logic that satisfies inconsistent explosion (i.e. that contains a formula $\bot$ such that $Cn(\bot) = \mathscr{L}$) this construction is fully characterized by a certain set of postulates.

Representation Theorem 5 (negation free partial meet revision). *For any compact logic that satisfies inconsistent explosion, $*$ is a negation free partial meet revision if and only if it satisfies* closure*,* success*,* strong inclusion*,* consistency*,* relevance *and* uniformity.

Kernel Revision

In the literature of belief base revision, we can find an alternative approach for constructing revision operation based on what is called kernels. Instead of finding the maximal subsets of the belief set that are consistent with the input, the kernel approach depends on finding the minimal subsets that are inconsistent with the input.

Formally, the *negation free kernel* $K \Downarrow \alpha$ is defined as follows:

Definition 6 (negation free kernel). $X \in K \Downarrow \alpha$ *iff:*

1. $X \subseteq K$.
2. $X + \alpha = \mathscr{L}$.
3. *If* $X' \subset X$ *then* $X' + \alpha \notin \mathscr{L}$.

Each element of $K \Downarrow \alpha$ will be called here an α*-kernel*. An *incision function* σ is a function that selects at least one element of each non-empty α-kernel. Formally it is any function such that $\sigma(K \Downarrow \alpha) \subseteq \bigcup K \Downarrow \alpha$ and if $\emptyset \neq X \in K \Downarrow \alpha$ then $X \cap \sigma(K \Downarrow \alpha) \neq \emptyset$.

Any incision function induces the construction of a revision operation:

$$K *_\sigma \alpha = (K \setminus \sigma(K \Downarrow \alpha)) + \alpha$$

In words, the incision function selects at least one element of each α-kernel in $K \Downarrow \alpha$. The elements selected by this function are removed from K and the result is expanded by α. This construction is the revision induced by σ.

One of the main differences between this and the previous representation theorem is the underlying minimality postulate. While relevance is the minimality postulate associated with negation free partial meet revision, core-retainment is the minimality postulate associated with negation free kernel revision.

Representation Theorem 7 (negation free kernel revision). *For any compact logic that satisfies inconsistent explosion, $*$ is a negation free kernel revision if and only if it satisfies* closure*,* success*,* strong inclusion*,* consistency*,* core-retainment *and* uniformity.

Related Work

There have been several proposals in the literature to adapt the AGM framework to non-classical logics, almost all of them dealing with very specific logics. Most of the recent work was concentrated around DLs and Horn Logics.

[2] This was called an *inconsistency-based remainder set* in (Delgrande 2008).

Delgrande and Peppas (2011) have proposed a construction for model-based revision of Horn theories and proved that it can be characterized by the eight AGM postulates (they include the supplementary postulates) together with an extra postulate for acyclicity of derivations, needed in the Horn case. It is interesting to note that they do not propose any minimality postulate. Zhuang, Pagnucco and Zhang (2013) show that they can obtain a revision characterized by the same postulates using a variation of the Levi identity and an operation for contraction of Horn theories. Their construction uses the idea of Horn strengthening and is thus specific for Horn Logics.

Qi, Liu and Bell (2006) have worked on revision operators for DLs based on the idea of weakening axioms to avoid inconsistencies. Again, this is a very specific method, tailored for specific logics. Qi et al. (2008) have proposed a kernel construction for revision in DLs, but they work with bases and not theories. Their operation satisfies a variant of core-retainment. There is, however, no proof that the postulates fully characterize the operation. More recently, Wang, Wang, and Topor (2010) have also proposed constructions for revision which satisfy most of the AGM postulates, but they focus on a very restricted logic of the DL-Lite family.

Conclusion

Traditional constructions for revision are based on contraction and the Levi identity. These constructions are not useful for logics that are not closed under negation. In this paper, we have shown two alternative constructions that do not depend on negation: negation free kernel revision (NFK) and negation free partial meet revision (NFPM).

Each of these constructions is associated to a certain minimality postulate: core-retainment and relevance respectively. For classical logics, both postulates follow from the other AGM revision postulates and, hence, are typically omitted. In general, however, they must be explicitly introduced.

Besides the postulates for minimality, we introduced two other postulates: uniformity (for syntactic independence) and a stronger version of inclusion.

NFPM revision had been already considered in a previous paper, but the representation theorem for NFPM proved in (Ribeiro and Wassermann 2009) depended on the logic being distributive. The version of the representation theorem for NFPM in this paper, on the other hand, is applicable to a wide class of logics, namely, any compact logic that satisfies inconsistent explosion. This is a much wider class of logics that includes, for example, Horn Logic and most DLs.

Besides fixing the representation theorem for NFPM, a construction based on kernel revision (NFK revision) was presented and the representation theorem was proved for the same class of logics.

Differently from related work, our approach solves the revision problem in non-classical logic from a general perspective. Our constructions and characterizations are applicable to a wide class of logics and do not rely on particularities of any specific one.

Future work includes a deeper study of the relation between contraction and revision in non-classical logics, as well as applying our results to Horn and DLs and comparing them with the specific proposals found in the literature for these logics.

References

Alchourrón, C., and Makinson, D. 1982. On the logic of theory change: contraction functions and their associated revision functions. *Theoria* 48:14–37.

Alchourrón, C.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change. *Journal of Symbolic Logic* 50(2):510–530.

Delgrande, J. P., and Peppas, P. 2011. Revising horn theories. In Walsh, T., ed., *IJCAI*, 839–844. IJCAI/AAAI.

Delgrande, J. P. 2008. Horn clause belief change: Contraction functions. In Brewka, G., and Lang, J., eds., *Principles of Knowledge Representation and Reasoning (KR2008)*, 156–165. AAAI Press.

Doyle, J. 1979. A truth maintenance system. *Artif. Intell.* 12(3):231–272.

Flouris, G.; Plexousakis, D.; and Antoniou, G. 2005. On applying the AGM theory to DLs and OWL. In Gil, Y.; Motta, E.; Benjamins, V. R.; and Musen, M. A., eds., *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, 216–231. Springer.

Flouris, G. 2006. *On Belief Change and Ontology Evolution*. Ph.D. Dissertation, University of Crete.

Hansson, S. O. 1991. Belief contraction without recovery. *Studia Logica* 50(2):251–260.

Harman, G. 1986. *Change In View: Principles of Reasoning*. Bradford Books. MIT Press.

Makinson, D. 1987. On the status of the postulate of recovery in the logic of theory change. *Journal of Philosophical Logic* 16:383–394.

Qi, G.; Haase, P.; Huang, Z.; and Pan, J. Z. 2008. A kernel revision operator for terminologies. In Baader, F.; Lutz, C.; and Motik, B., eds., *Description Logics*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Qi, G.; Liu, W.; and Bell, D. A. 2006. Knowledge base revision in description logics. In Fisher, M.; van der Hoek, W.; Konev, B.; and Lisitsa, A., eds., *JELIA*, volume 4160 of *Lecture Notes in Computer Science*, 386–398. Springer.

Ribeiro, M. M., and Wassermann, R. 2009. AGM revision in description logics. In *Proceedings of the 1st Workshop on Automated Reasoning about Context and Ontology Evolution (ARCOE 2009)*.

Ribeiro, M. M., and Wassermann, R. 2010. More about AGM revision in description logics. In *Proceedings of the 2nd Workshop Automated Reasoning about Context and Ontology Evolution (ARCOE 10)*.

Ribeiro, M. M.; Wassermann, R.; Flouris, G.; and Antoniou, G. 2013. Minimal change: Relevance and recovery revisited. *Journal of Artificial Intelligence* 201:59–80.

Wang, Z.; Wang, K.; and Topor, R. W. 2010. Revising general knowledge bases in description logics. In Lin, F.; Sattler, U.; and Truszczynski, M., eds., *KR*. AAAI Press.

Zhuang, Z. Q.; Pagnucco, M.; and Zhang, Y. 2013. Definability of horn revision from horn contraction. In Rossi, F., ed., *IJCAI*. IJCAI/AAAI.

Predicting Performance of OWL Reasoners: Locally or Globally?

Viachaslau Sazonau and **Uli Sattler** and **Gavin Brown**
The University of Manchester
Oxford Road, Manchester, M13 9PL, UK
{*sazonauv, sattler, gbrown*}*@cs.manchester.ac.uk*

Abstract

We propose a novel approach for performance prediction of OWL reasoners that selects suitable, small ontology subsets, and then extrapolates reasoner's performance on them to the whole ontology. We investigate intercorrelation of ontology features using PCA and discuss various error measures for performance prediction.

Introduction

An ontology is a machine-processable representation of knowledge about a domain of interest. Ontologies are encoded in knowledge representation languages which are formally based on logic. A common ontology language is the Web Ontology Language (Horrocks, Patel-Schneider, and Van Harmelen 2003), OWL. We consider only OWL ontologies. Since an ontology is essentially a set of logical formulae, one can reason over it. As a result, one can derive inferences, or entailments, encoding implicit knowledge. Tools that implement reasoning services are called *reasoners*. A standard service invoked by ontology engineers is *ontology classification*, i.e. the computation of an inferred class hierarchy.[1] We focus on ontology classification.

Although there exist many reasoners, there is no single reasoner that performs best on all given inputs. The same ontology may take just several seconds to reason over for some reasoners and require hours to process for others (Gonçalves, Parsia, and Sattler 2012). Recent results of the competition between 14 reasoners taking place at the OWL Reasoner Evaluation Workshop, ORE 2013,[2] show that there is no single winner in all ontology categories. The huge difference in performance of reasoners on a given input often surprises ontology designers.

During ontology design and maintenance, reasoning performance may vary significantly and in surprising ways: axiom additions or removals may cause a significant increase or decrease of classification time, often in contrast to intuitive expectations (Gonçalves, Parsia, and Sattler 2012). Recent studies of performance variability suggest that there are ontologies which are *performance homogeneous* and others are *performance heterogeneous* for a reasoner (Gonçalves, Parsia, and Sattler 2012). An ontology $\mathcal{O}$ is performance heterogeneous if classification time of $\mathcal{O}' \subset \mathcal{O}$ is not linearly correlated with its relative size. The latter suggests that interaction effects come into play when certain axioms are part of the ontology and, consequently, loose their impact on reasoner's performance when those axioms are removed.

These observations make performance prediction of a reasoner on a given input challenging and often impossible even for experienced and skilled ontology engineers. However, the ability to predict performance is attractive. Firstly, reasoner developers equipped with a predictor would be able to find out which ontologies are harder for a reasoner. This can speed up tests of reasoners and facilitate new optimizations. Secondly, an estimate of classification time might allow an ontology engineer to decide whether the reasoning task is worth waiting for or not. As a practical example, the progress bar of an ontology editor, such as Protégé 4,[3] could be made more reliable with a good performance predictor. It could also supply auto-picking the fastest reasoner for a given ontology from the list of available reasoners.

The contributions of this work are three-fold. Firstly, we suggest an approach to analyse intercorrelation of ontology features using PCA, given a corpus of ontologies, and observe that ontology features are highly intercorrelated so that 57 features can be "faithfully" replaced by one or two features. Secondly, we introduce a new method for performance prediction which is competitive with existing performance predictors. Thirdly, we discuss different prediction accuracy measures. All details can be found in the technical report (Sazonau, Sattler, and Brown 2013).

Feature Analysis

There exist recent attempts to apply supervised machine learning to performance prediction of a reasoner on a given ontology. In (Kang, Li, and Krishnaswamy 2012), the authors define 27 ontology features, or *metrics*, of 4 types: ontology level (*ONT*), class hierarchy level (*CH*), anonymous class expressions (*ANE*) and properties (*PRO*). A feature vector $\mathbf{x}_i$ is extracted from each ontology $\mathcal{O}_i$, $i = 1, ..., N$ in the corpus. It is assigned a label $y_i = b(\mathcal{O}_i, \mathcal{R})$, $i =$

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] Throughout, we use OWL terminology where a class is a concept, a property is a DL role (Horrocks, Patel-Schneider, and Van Harmelen 2003).

[2] http://ore2013.cs.manchester.ac.uk/competition/

[3] http://protege.stanford.edu/

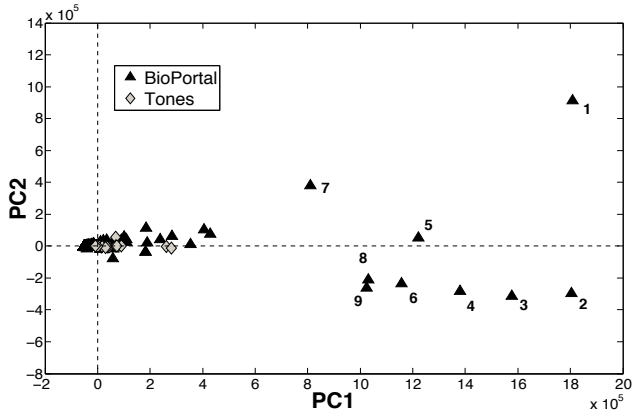


Figure 1: Comparing BioPortal and Tones using PCA

$1 \dots N$, which is the performance bin of $\mathcal{O}_i$ given a reasoner $\mathcal{R}$. Kang et al. attempt to predict the performance bin $\hat{y} = \hat{b}(\mathcal{O}, \mathcal{R})$ for an ontology $\mathcal{O}$ based on its feature vector $\mathbf{x}$, or "profile".[4] To do so, they train a model using feature vectors $\mathbf{X}$ of *all training ontologies* in the corpus with their *labels* $\mathbf{y} = (y_1, y_2, ..., y_N)$ per reasoner, i.e. the "global" corpus view $[\mathbf{X} \mid \mathbf{y}^T]$. We call such a technique a *global approach*. For basics of predicting algorithm runtime, we refer the interested reader to a survey of machine learning methods for three NP-complete problems (Hutter et al. 2014).

As we will see, it is considerably hard or even impossible to define the "ideal" set of ontology features for performance prediction of reasoners. Nonetheless, we need to take features of different scales into account, e.g. EOG and SOV,[5] used by Kang et al. Competing approaches exist in machine learning to address this problem (Tuv, Borisov, and Torkkola 2006). Another problem that should be noticed is that a corpus of ontologies should be selected carefully. A corpus, significantly biased towards easy ontologies, may cause misleading generalizations. Although dealing with such bias is generally difficult due to the lack of hard ontologies, we need to take this into account.

We skip some features from Kang et al. and add our own features. The inferred SubClass relationships are not included because this requires classification and, thus, makes little sense for performance prediction. Overall, we consider $d = 57$ ontology features including 33 new features. Intuitively, more features should allow us to capture more information about the object. However, a smaller set of features can produce a more accurate model in practice (Guyon and Elisseeff 2003). It can be achieved either by eliminating weakly useful features from the original set, hence, *selecting* features, or by transforming the features into a lower dimensional space minimizing information loss, known as *constructing* features (Guyon and Elisseeff 2003). A common feature construction method is *Principal Component Analysis*, PCA (Jolliffe 2002). In a nutshell, PCA helps to identify the presence of *intercorrelation* of features and provides a new, smaller set of uncorrelated features.

We have chosen the NCBO BioPortal[6] as an ontology corpus for experiments since it is a natural set of ontologies primarily maintained by biomedical domain experts. We exclude empty ontologies, duplicates, multiple versions of the same ontology, and ontologies with loading errors. As a result, we obtain the BioPortal corpus of 357 ontologies of various sizes. We also examine an additional corpus, the Tones Ontology Repository.[7] We filter it out via exactly the same procedure. As a result, we obtain the Tones corpus of 196 ontologies.

Given a data set $[\mathbf{X} \mid \mathbf{y}^T]$, PCA deals only with a set of feature vectors $\mathbf{X}$ and *ignores* labels $\mathbf{y}$. It produces a set of PCs, PC_j, with respective variances, λ_j. We apply PCA to BioPortal and obtain a set of PCs, PC_j^B. To check our findings, we also apply PCA to Tones, which induces another set of PCs, PC_j^T, and compare the results. We use PC_j^X to denote both PC_j^B and PC_j^T. As a result, we found out that the first 3 out of 57 components explain ≈99% of all variance for both data sets. Hence, we can hypothesize that the original 57 features are *highly intercorrelated*. Another important thing to notice is that PC_1^B explains $\approx 87\%$ and PC_1^T explains $\approx 96\%$ of total variance for BioPortal and Tones, respectively.

An additional advantage of PCA is that it allows us to visualise a data set in a lower dimensional space. Since just few components have significant variances, we can reasonably visualise the corpora in two dimensional space (PC_1^X, PC_2^X) and explore their spread in the projection space. This way, we preserve $\approx 94\%$ of total variance for BioPortal and $\approx 98\%$ of total variance for Tones. As Figure 1 shows, ontologies of both corpora are mainly spread along the first coordinate with smaller deviations along the second coordinate and even smaller along others.

BioPortal has more outliers, i.e. ontologies which differ from the main cluster of similar ontologies. There are 9 evident outliers out of 357 ontologies. They include 9 out of 10 biggest ontologies in the corpus. In addition, these outliers are amongst the computationally hardest ontologies for all three reasoners. None of them are the heterogeneous ontologies from (Gonçalves, Parsia, and Sattler 2012). It is important to note that the outliers have been identified *without any performance measurements*.

It is worth noticing that the same 6 original features are the only ones that contribute significantly to PC_1^X for both corpora. Moreover, these features have similar contribution coefficients for both corpora. This implies that PC_1^B is *approximately equal* to PC_1^T. Exploring feature vectors, we observe that these features significantly correlate with ontology size.[8] Hence, one can be interested how PC_1^X is affected by it. To measure this, we calculate the correlation coefficient between the first component and ontology size. It turns out that the first component correlates surprisingly well with ontology size and no other components do. As a conse-

[4] Not to be confused with OWL profiles such as OWL 2 EL

[5] http://www.csse.monash.edu/∼yli/metrics_perf/

[6] http://bioportal.bioontology.org/

[7] http://owl.cs.manchester.ac.uk/repository/

[8] In the following, ontology size is the number of axioms

quence, one can ask an interesting question: is ontology size a good performance indicator? Can we predict performance better with 57 features or with size alone? We will answer these questions below.

A Local Approach to Performance Prediction

Here we sketch an approach for predicting performance of a reasoner on a given ontology, given classification time of that reasoner on some of the ontology subsets, ideally few and small, which we call *samples*. We call such a technique a *local approach*.

A reasonable way to construct samples is to represent an ontology as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices and $\mathcal{E}$ is a set of directed edges. Each vertice $v \in \mathcal{V}$ is labelled with some set of axioms, called a *clump*. Edges $\mathcal{E}$ represent relationships between clumps. From such a representation we can assemble sets of clumps in many possible ways and use these sets as samples for predictions.

Obviously, the quality of such performance prediction depends on our notion of clumps, their relations, and the assembling mechanism used. For example, a trivial way is to take small arbitrary ontology subsets as samples without any relations between them. However, they are not suitable performance indicators because they can be so only by chance, which is low, given that the size of an ontology is sufficiently large. A more informed way to define clumps is Atomic Decomposition, AD (Del Vescovo 2013). In this case, we ensure cohesion of samples.

Given $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, sampling starts from a single atom. A next sample is constructed by extending the current sample with a new atom. A sensible way of choosing a next atom can cause significant increases of classification time of a sample. The sampling algorithm is described in (Sazonau, Sattler, and Brown 2013). It produces a set of samples with their classification times per ontology.

This local data is then used to make predictions for the whole ontology via regression analysis. Thus, where a global approach gathers data from many ontologies and interpolates for unseen ontologies, a local approach collects data only from part of a given ontology and extrapolates to its full size. In contrast to a global approach, it does not rely on the global view of a corpus to exploit similarities/dissimilarities between ontologies and it uses no features except sample sizes.

Performance Prediction Experiments

In this work, we investigate three commonly used reasoners: Hermit 1.3.8 (Motik, Shearer, and Horrocks 2007), Pellet 2.3.0 (Sirin et al. 2007), and JFact 1.0.0, a Java version of FaCT++ (Tsarkov and Horrocks 2006). In addition to the performance bins specified by Kang et al., we add one more bin E in order to distinguish the hardest ontologies with classification times longer than 1,000 seconds. Thus, we consider the following 5 bins: $A \in (0s, 1s)$, $B \in [1s, 10s)$, $C \in [10s, 100s)$, $D \in [100s, 1,000s)$, $E \in [1,000s, +\infty)$. The performance statistics on BioPortal is shown in Table 1, where the "Exception" bin means that a reasoner raises an exception on those ontologies and does not finish its job.

Bins	JFact	Hermit	Pellet
A	250	222	247
B	50	53	48
C	16	21	21
D	11	13	11
E	23	16	28
Exception	7	32	2

Table 1: Performance statistics on BioPortal (number of ontologies per bin)

We aim to test a global and local performance predictor and compare them. Consequently, we need a procedure to evaluate a performance predictor. The first possibility is based on performance bins. The model predicts a bin for each ontology from the testing set. Whenever it predicts a wrong bin for an ontology, this is counted as an error. Once the model is tested on all ontologies, an average error is calculated which we call the *average classification error*, ACE. Kang et al. use the classification accuracy, $1 - ACE$, to evaluate a model. When predicting performance bins, we suggest to additionally consider a more precise way of evaluation - the *confusion matrix* (Fawcett 2006), which is especially useful for unbalanced data sets. It essentially counts the number of ontologies of actual bin b which are classified to bin $\hat{b}$ for each possible pair $(b, \hat{b})$.

A third way to assess a model is to measure raw differences between the actual and predicted classification time for each testing ontology. The *mean absolute error*, MAE, is a common measure that can be used for this purpose. Nonetheless, hard ontologies, which are rare, can contribute more to MAE than all remaining ontologies together. Thus, all aforementioned measures have their disadvantages, no single measure is enough for predictor evaluation, and we should rather consider them all.

We conduct two performance prediction experiments for the global approach. In the first one, we extract all 57 features from each ontology in BioPortal and train a model, *Glob.57f*, using the resulting data set. In the second one, we record the ontology size (in number of axioms) per ontology and use it as the only feature to train a model, *Glob.1f*. The latter is based on our earlier observations and PCA results. For the local approach we construct samples per ontology as explained above. We restrict samples to have relative sizes $\leq 10\%$ of the full ontology considering the following tradeoff: higher size limits should give more accurate predictions but take longer to compute. Prediction results of the global and local approach are gathered in Table 2.

As Table 2 shows, the local approach delivers better prediction accuracy than the global one in all cases, for both ACE and MAE (the lowest errors are in bold). Extracting 57 ontology features makes surprisingly little sense, especially for bin predictions (ACE): the differences in errors between Glob.57f and Glob.1f are small. One can also notice that Pellet is harder to predict than others for all approaches via either ACE or MAE. It is also worth mentioning that all reasoners are mostly *linearly* predictable.

The confusion matrices, the one for JFact is given by Table 3, show that Glob.57f and Glob.1f are slightly more accurate than the local approach on the easiest bin A, while the

Reasoner	ACE		
	Glob.57f	**Glob.1f**	**Local**
JFact	0.230 (0.048)	0.211 (0.045)	**0.169**
Hermit	0.231 (0.053)	0.237 (0.047)	**0.224**
Pellet	0.301 (0.048)	0.301 (0.048)	**0.250**
	MAE, sec		
JFact	61 (24)	69 (18)	**43**
Hermit	70 (25)	89 (23)	**42**
Pellet	106 (31)	127 (27)	**69**

Table 2: Comparison of global and local approaches on BioPortal (error deviations are in brackets where appropriate)

Real bin	**Predicted bin**				
	A	B	C	D	E
A	**245/242/232**	4/7/16	0/0/1	0/0/0	1/1/1
B	31/23/15	**19/27/28**	0/0/7	0/0/0	0/0/0
C	2/0/0	9/9/0	**3/2/11**	0/0/3	2/5/2
D	1/1/0	2/2/0	3/4/1	**0/0/6**	5/4/4
E	2/2/1	5/4/1	4/5/3	0/3/4	**12/9/14**

Table 3: Confusion matrix for bin predictions of JFact on BioPortal (Glob.57f/Glob.1f/Local)

latter is more correct on all harder bins B, C, D, E; especially on bins C, D. If easy ontologies (bin A) are excluded, it turns out that the remaining ontologies cause extremely higher errors of performance predictions for all approaches.

Discussion of Results and Future Work

To conclude, we have observed that performance prediction of reasoners is a hard and interesting problem. In this work, we propose a novel approach, called local, to solve it. It is based on carefully selecting ontology samples, ideally small, and then using them to predict reasoning performance on the full ontology. The AD, as an ontology graph, is useful to construct such samples: samples of sizes $\leq 10\%$ allow us to obtain good predictions via simple extrapolations. In comparison to a global approach, a local approach gives more accurate performance predictions, does not rely on an ontology corpus and, therefore, cannot be biased by it. Moreover, a local approach reveals interesting information about reasoners' behaviour: linear/nonlinear predictability on the corpus. However, it requires reasoning over samples to make a prediction.

Although selecting an unbiased corpus remains a difficult problem (Matentzoglu, Bail, and Parsia 2013), we definitely need to acknowledge the bias and interpret observations correctly. Moreover, the error measures for evaluating predictors should be chosen carefully because there is no single best measure. A reasonable way is to use several measures evaluating different aspects of prediction accuracy.

One of our main findings is that, for both BioPortal and Tones, multiple ontology features are efficiently represented by ontology size. Predictions using ontology size alone are comparable to predictions using all 57 ontology features. Therefore, ontology features for performance prediction should be defined and used cautiously. Firstly, we should not mix features of different scales in the feature vector because low-scale features can be "hidden" by others. Nonetheless, if these features are informative, suitable techniques, able to handle different scales, must be used. Secondly, we have to investigate intercorrelation of features because the features may turn out to carry little useful information and even degrade the accuracy.

As future work, we consider local approach improvements. The maximal sample size can be estimated for each ontology separately because 10% samples are not indicative enough to make good predictions for some ontologies. In addition, there may exist more suitable regression models for extrapolation. One can also study appropriate methods to cope with ontology features of different scales with respect to performance prediction, e.g. ensemble learning. Finally, we can work on assembling "good" corpora using dimensionality reduction techniques such as PCA.

References

Del Vescovo, C. 2013. *The Modular Structure of an Ontology: Atomic Decomposition and its Applications*. Ph.D. Dissertation, School of Computer Science, University of Manchester.

Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27(8):861 – 874.

Gonçalves, R. S.; Parsia, B.; and Sattler, U. 2012. Performance heterogeneity and approximate reasoning in description logic ontologies. In *Proc. of ISWC-12*, volume 7649 of *LNCS*, 82–98.

Guyon, I., and Elisseeff, A. 2003. An introduction to variable and feature selection. *J. of Mach. Learning Res.* 3:1157–1182.

Horrocks, I.; Patel-Schneider, P.; and Van Harmelen, F. 2003. From SHIQ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics* 1(1):7–26.

Hutter, F.; Xu, L.; Hoos, H. H.; and Leyton-Brown, K. 2014. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence* 206:79–111.

Jolliffe, I. T. 2002. *Principal Component Analysis*. Springer, second edition.

Kang, Y.-B.; Li, Y.-F.; and Krishnaswamy, S. 2012. Predicting reasoning performance using ontology metrics. In *Proc. of ISWC-12*, volume 7649 of *LNCS*, 198–214.

Matentzoglu, N.; Bail, S.; and Parsia, B. 2013. A corpus of OWL DL ontologies. In Eiter, T.; Glimm, B.; Kazakov, Y.; and Krötzsch, M., eds., *Description Logics*, volume 1014 of *CEUR Workshop Proceedings*, 829–841. CEUR-WS.org.

Motik, B.; Shearer, R.; and Horrocks, I. 2007. A hypertableau calculus for $\mathcal{SHIQ}$. In *Proc. of DL-07*, 419–426. Bozen/Bolzano University Press.

Sazonau, V.; Sattler, U.; and Brown, G. 2013. Predicting performance of OWL reasoners: Locally or globally? Technical report, School of Computer Science, University of Manchester.

Sirin, E.; Parsia, B.; Cuenca Grau, B.; Kalyanpur, A.; and Katz, Y. 2007. Pellet: A practical OWL-DL reasoner. *J. of Web Semantics* 5(2):51–53.

Tsarkov, D., and Horrocks, I. 2006. FACT++ Description Logic reasoner: System description. In *Proc. of IJCAR-06*, volume 4130 of *LNCS*, 292–297. Springer-Verlag.

Tuv, E.; Borisov, A.; and Torkkola, K. 2006. Best subset feature selection for massive mixed-type problems. In *Proc. of IDEAL-06*, 1048–1056. Springer-Verlag.

Canonical Logic Programs Are Succinctly Incomparable with Propositional Formulas

Yuping Shen and **Xishun Zhao***
Institute of Logic and Cognition
Department of Philosophy
Sun Yat-sen University
510275 Guangzhou, P.R. China
{*shyping, hsszxs*}*@mail.sysu.edu.cn*

Abstract

Canonical (logic) programs (CP) refer to the class of *normal* programs (LP) augmented with connective *not not*, and are equally expressive as *propositional formulas* (PF). In this paper we address the question of whether CP and PF are *succinctly incomparable*. Our main result shows that the PARITY problem *only* has exponential CP representations, while it can be polynomially represented in PF. In other words, PARITY *separates* PF from CP. Simply speaking, this means that exponential size blowup is generally inevitable when translating a set of PF formulas into a (logically) equivalent CP program (without introducing new variables). Furthermore, since it has been shown by Lifschitz and Razborov that there is also a problem which separates CP from PF (assuming $\mathsf{P} \not\subseteq \mathsf{NC}^1/\mathsf{poly}$), it follows that the two formalisms are indeed succinctly incomparable.

1 Introduction

The relationship between *(logic) programs* under *answer set semantics* (ASP) (Lifschitz 2008; Brewka, Eiter, and Truszczynski 2011) and *propositional satisfiability* (SAT) (Biere et al. 2009) gains a lot of attention in the literature. In 2006, Lifschitz and Razborov proved that an exponential size blowup is generally inevitable when translating a *normal program* (LP) to a (logically) equivalent set of *propositional formulas* (PF) (without introducing additional variables). More precisely, they showed that (a variant of) the P-complete problem PathSystem (PATH) has polynomial size LP representations, however, it *cannot* be polynomially represented in PF (assuming $\mathsf{P} \not\subseteq \mathsf{NC}^1/\mathsf{poly}$) (Lifschitz and Razborov 2006), i.e., PATH *separates* LP from PF.

As noted in (Lifschitz and Razborov 2006), PF can be considered as a special case of the class of *(nondisjunctive) nested programs* (NLP, without classical negation $\neg$) (Lifschitz 1999). Therefore, NLP is *stronger* than PF in terms of the *succinctness* criterion (or the *"comparative linguistics" approach*) (Gogic et al. 1995):

> That is, we consider formalism A to be stronger than formalism B if and only if any knowledge base (KB) in B has an equivalent KB in A that is only polynomially longer, while there is a KB in A that can be translated to B only with an exponential blowup.

So the following footnote in (Lifschitz 2008) seems convincing:

> ...ASP appears to be stronger than SAT in the sense of the "comparative linguistics" approach to knowledge representation...

However, ASP involves many kinds of programs, the above statement probably needs further clarification. Particularly, the class of so-called *(nondisjunctive) canonical programs* (CP[1]) (Lifschitz 1999; Lee 2005; Lee, Lifschitz, and Yang 2013), is a minimal form of ASP that is equally expressive as PF, but looks more likely *not* succinctly stronger. So a question naturally arises: *Does there exist a problem that separates PF from CP?* If there is such a problem, then CP and PF are *succinctly incomparable* (assuming $\mathsf{P} \not\subseteq \mathsf{NC}^1/\mathsf{poly}$).

In this paper we address the question and give a *positive* answer. Our main result shows that the PARITY problem separates PF from CP. Simply speaking, this means exponential size blowup is generally inevitable when translating a set of PF formulas into an equivalent CP program. The PARITY problem asks whether a binary string contains an odd number of 1's, and it is well-known that (i) PARITY$\in \mathsf{NC}^1/\mathsf{poly}$[2], i.e., it has polynomial PF representations; (ii) PARITY$\notin \mathsf{AC}^0$, i.e., it cannot be represented by *polynomial* size (boolean) circuits with *constant depth* and *unbounded fan-in* (Arora and Barak 2009; Jukna 2012).

To show PARITY separates PF from CP, we provide a procedure that simplifies every PARITY program Π into a shorter program Π' s.t. Π' is equivalent to its *completion* $Comp(\Pi')$ (Erdem and Lifschitz 2003). Such completions are essentially constant depth, unbounded fan-in circuits. According to PARITY$\notin \mathsf{AC}^0$, these circuits must be of exponential size, consequently, there are no polynomial size PARITY programs in CP. In the rest of the paper, we shall introduce the basic concepts, key steps of our proof and discuss the importance of succinctness research in the theory and practice of Knowledge Representation (KR).

*Corresponding author.
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] Extends LP with connective *not not*.

[2] $\mathsf{NC}^1/\mathsf{poly}$ (or *non-uniform* NC^1) *exactly* contains languages computable (representable) by polynomial size PF formulas.

2 Background

Canonical Programs

The following notations are adopted from (Lifschitz 1999; Lee 2005). A *rule element* e is defined as

$$e := \top \mid \bot \mid x \mid not\ x \mid not\ not\ x$$

in which $\top, \bot$ are 0-ary connectives, x is a *(boolean) variable* (or an *atom*) and not is a unary connective[3]. A *(nondisjunctive canonical) rule* is an expression of the form

$$H \leftarrow B \tag{1}$$

where the *head* H is either a variable or the connective $\bot$, and the *body* B is a finite set of rule elements. A *canonical program* Π is a finite set of rules. E.g., the following is a canonical program:

$$\begin{array}{ll} x_1 \leftarrow not\ not\ x_1, & x_3 \leftarrow not\ x_1, not\ x_2, \\ x_2 \leftarrow not\ not\ x_2, & x_3 \leftarrow x_1,\ x_2. \end{array} \tag{2}$$

A canonical program Π is *normal* if it contains no connectives $not\ not$. A normal program Π is *basic* if it contains no occurrences of connective not.

The *satisfaction relation* $\models$ between a set of variables I and a rule element is defined as follows:

- $I \models \top$ and $I \not\models \bot$,
- $I \models x$ iff $I \models not\ not\ x$ iff $x \in I$,
- $I \models not\ x$ iff $x \notin I$.

Say I satisfies a set of rule elements B if I satisfies each rule element in B. We say I is *closed* under a program Π, if I is closed under every rule in Π, i.e., for each rule $H \leftarrow B \in \Pi$, $I \models H$ whenever $I \models B$. Let Π be a basic program and let $Cn(\Pi)$ denotes the *minimal* set (in terms of set inclusion) closed under Π, we say I is an *answer set* of Π if $I = Cn(\Pi)$. Note that a basic program has exactly one answer set.

The *reduct* Π^I of a program Π w.r.t. I is a set of rules obtained from Π via: (i) Replacing each $not\ not\ x$ with $\top$ if $I \models x$, and with $\bot$ otherwise; (ii) Replacing each $not\ x$ with $\top$ if $I \not\models x$, and with $\bot$ otherwise. Observe that Π^I must be a basic program. We say I is an answer set of Π if $I = Cn(\Pi^I)$, i.e., I is an answer set of Π^I. E.g., the following single rule program:

$$x \leftarrow not\ not\ x \tag{3}$$

has two answer sets $\emptyset$ and $\{x\}$.

For a set of rule elements B, define $var(B) = \{e \in B : e \text{ is a variable}\}$. E.g., $var(\{x_1, not\ x_2, not\ not\ x_3\}) = \{x_1\}$. The *signature* $sig(\Pi)$ of a program Π is the set of all involved variables in Π. By $Ans(\Pi)$ we denote the set of all answer sets of Π. E.g., if Π is (2), then $sig(\Pi) = \{x_1, x_2, x_3\}$ and $Ans(\Pi) = \{\{x_1, x_2, x_3\}, \{x_1\}, \{x_2\}, \{x_3\}\}$. The *size* $|\Pi|$ of Π is the number of its rules. As a convention, Π_n refers to a program with signature $\{x_1, \ldots, x_n\}$, i.e., $sig(\Pi_n) = \{x_1, \ldots, x_n\}$.

It is easy to see that by using rules of the form (3) and appropriate *constraints* of the form $\bot \leftarrow B$, it is easy to give an arbitrary set of answer sets over $sig(\Pi_n)$, in other words, CP has exactly the same expressive power as PF.

[3]By (Lifschitz 1999), $not\ not\ not\ x$ can be replaced by $not\ x$.

Problem Representation and Succinctness

A *(binary) string* is a finite sequence of *bits* from $\{0, 1\}$. A string w of length n (i.e., $w \in \{0, 1\}^n$) defines a subset of variables $\{x_1, \ldots, x_n\}$. E.g., 1010 stands for $\{x_1, x_3\}$. Therefore, a set of variables I and a string w can be regarded as the same. A *problem* (or *language*) L is a set of strings.

Definition 2.1 (Problem Representation). *A problem L can be* represented *in a class of programs (or formulas, etc) $\mathcal{C}$ (i.e., $L \in \mathcal{C}$), if there exists a* sequence *of programs $\{\Pi_n\}$ ($n = 1, 2, \ldots$) in $\mathcal{C}$ that* computes *L, i.e., for every string $w \in \{0, 1\}^n$,*

$$w \in L \Leftrightarrow w \in Ans(\Pi_n).$$

Say L has polynomial representations in $\mathcal{C}$ (i.e., $L \in$ Poly-$\mathcal{C}$), if $L \in \mathcal{C}$ and $|\Pi_n|$ is bounded by a polynomial $p(n)$.

Recall that the PARITY problem is the set of binary strings with an odd number of 1's. By PARITY_n we denote PARITY strings of length n. Clearly, (2) represents PARITY_3 since its answer sets are $111, 100, 010$ and 001.

Definition 2.2 (Succinctness (Gogic et al. 1995; French et al. 2013)). *Let $\mathcal{C}, \mathcal{C}'$ be two classes of programs such that for every problem L, $L \in \mathcal{C} \Leftrightarrow L \in \mathcal{C}'$. Say $\mathcal{C}$ is* at least as succinct as *$\mathcal{C}'$ (i.e., $\mathcal{C}' \preceq \mathcal{C}$), if for every problem L,*

$$L \in \text{Poly-}\mathcal{C}' \Rightarrow L \in \text{Poly-}\mathcal{C}.$$

If $L \in$ Poly-$\mathcal{C}$ but $L \notin$ Poly-$\mathcal{C}'$ (i.e., $\mathcal{C} \not\preceq \mathcal{C}'$), then L separates *$\mathcal{C}$ from $\mathcal{C}'$. If $\mathcal{C}' \preceq \mathcal{C}$ and $\mathcal{C} \not\preceq \mathcal{C}'$, then $\mathcal{C}$ is* strictly more succinct than *$\mathcal{C}'$ (i.e., $\mathcal{C}' \prec \mathcal{C}$). Moreover, $\mathcal{C}, \mathcal{C}'$ are* succinctly incomparable *if there is a problem L that separates $\mathcal{C}$ from $\mathcal{C}'$, and vice versa (i.e., $\mathcal{C} \not\preceq \mathcal{C}'$ and $\mathcal{C}' \not\preceq \mathcal{C}$).*

PARITY Representations and Completion

Note that (2) suggests a "pattern" for PARITY_n: The first part of the program (e.g., the "*not not*" rules in (2)) generates all possible strings of length $n-1$, the second part identifies the last bit to form an odd string of length n. Following the pattern, we can give a sequence of *exponential* size CP programs for PARITY, i.e., PARITY$\in$CP.

On the other hand, it is a textbook result that PARITY has polynomial size PF representations with classical connectives $\{\wedge, \vee, \neg\}$, i.e., PARITY$\in \mathsf{NC}^1/\mathsf{poly}$ (or Poly-PF). Remind that the size of a formula is the number of its connectives. E.g., $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$ represents PARITY_2, and the formula can be recursively applied for arbitrary PARITY_n. It is also well-known that PARITY$\notin \mathsf{AC}^0$, i.e., PARITY cannot be represented by a sequence of polynomial size circuits $\{C_n\}$ in which these circuits C_n have unbounded fan-in and constant depth.

Simply speaking, a sequence of polynomial size PF formulas $\{\phi_n\}$ represents an AC^0 language, if the number of variables occur in a *conjunction* or *disjunction* of ϕ_n is *unrestricted*, and the distance of the longest path from the root to a leaf in the tree structure of each ϕ_n is *fixed*. E.g., PARITY cannot be represented by a sequence of polynomial size CNFs $\{\psi_n\}$ since $\{\psi_n\}$ represents an AC^0 language. For more details about circuits, please see (Arora and Barak 2009; Jukna 2012).

The *completion* $Comp(\Pi)$ (Erdem and Lifschitz 2003) of a CP program Π consists of a set (or a conjunction) of PF formulas (we slightly abuse the connective $\equiv$): (i) $x \equiv \tilde{B}_1 \vee \tilde{B}_2 \vee \cdots \vee \tilde{B}_m$, where $x \leftarrow B_1, \ldots, x \leftarrow B_m$ are all rules in Π with head x, and each $\tilde{B}_i$ is the conjunction of rule elements in B_i with connective *not* replaced by $\neg$; (ii) $x \equiv \bot$, if x is not a head of any rule in Π; (iii) $\neg\tilde{B}$, if a rule $\bot \leftarrow B$ is in Π.

Proposition 2.1. *The completion $Comp(\Pi)$ of an arbitrary canonical program Π is a constant depth, unbounded fan-in circuit whose size is polynomially bounded by $|\Pi|$.*

It is well-known that an answer set of Π is also a model of its completion, but the inverse generally does not hold. E.g., the completion $\{x_1 \equiv \neg x_2 \vee (x_2 \wedge x_1), x_2 \equiv \neg x_1 \vee (x_1 \wedge x_2)\}$ of the PARITY$_2$ program:

$$\begin{array}{ll} x_1 \leftarrow not\ x_2, & x_2 \leftarrow not\ x_1, \\ x_1 \leftarrow x_2, not\ not\ x_1, & x_2 \leftarrow x_1, not\ not\ x_2, \end{array} \quad (4)$$

has an even string model 11, i.e., (4) is not *completion-equivalent*. In the next section, we will show how to simplify a PARITY program to be completion-equivalent.

3 Simplifying PARITY Programs

Let B be a finite set of rule elements built on signature $V = \{x_1, \ldots, x_n\}$. By $S(B)$ we denote the set $\{I \subseteq V : I \models B\}$. E.g., let $V = \{x_1, x_2, x_3, x_4\}$ and $B = \{x_2, not\ x_3, not\ not\ x_4\}$, then $S(B) = \{\{x_1, x_2, x_4\}, \{x_2, x_4\}\} = \{1101, 0101\}$. We say B is *consistent* if there is a set of variables I s.t. $I \models B$. We say B *covers* a variable $x \in V$ iff $x \in B$ or $not\ x \in B$ or $not\ not\ x \in B$. If B covers every variable in V then B *fully covers* V. E.g., $B = \{x_1, not\ x_2, not\ not\ x_3\}$ fully covers $V = \{x_1, x_2, x_3\}$. Clearly, B is consistent and fully covers V iff $S(B)$ contains a unique string.

In the following we assume that all rules $x \leftarrow B$ have consistent body B and $x \notin var(B)$. It is easy to see that the assumption does not affect the generality.

Lemma 3.1. *Let Π_n be a PARITY$_n$ program. If there is a rule $x \leftarrow B$ in Π_n s.t. $not\ not\ x \in B$ and $S(B)$ contains a unique even string, then dropping $x \leftarrow B$ from Π_n results in a PARITY$_n$ program Π'_n.*

E.g., by Lemma 3.1, (4) can be simplified to:

$$x_1 \leftarrow not\ x_2, \quad x_2 \leftarrow not\ x_1. \quad (5)$$

Lemma 3.2. *Let Π_n be a PARITY$_n$ program. If there is a rule $x \leftarrow B$ in Π_n s.t. $not\ not\ x \in B$ and $S(B)$ contains a unique odd string, then dropping $not\ not\ x$ from B results in a PARITY$_n$ program Π'_n.*

Standard Programs and the Main Theorem

A PARITY$_n$ program Π_n is *standard* if for each rule $x \leftarrow B \in \Pi_n$, $not\ not\ x \notin B$ whenever $S(B \cup \{x\})$ contains a unique string. E.g., (2) and (5) are standard programs. By Lemma 3.1 and 3.2, it is not hard to see:

Proposition 3.1. *Let Π_n be a PARITY$_n$ program. Then there is a standard PARITY$_n$ program Π'_n s.t. $|\Pi'_n| \leq |\Pi_n|$.*

Moreover, note that (2) and (5) are completion-equivalent PARITY programs. This is guaranteed by the following proposition:

Proposition 3.2. *Let Π_n be a standard PARITY$_n$ program. Then Π_n is equivalent to its completion $Comp(\Pi_n)$.*

The proof idea of Proposition 3.2 is that every standard PARITY$_n$ program Π_n can be equivalently rewritten to Π'_n by replacing each $x \in var(B)$ with $not\ not\ x$ for every rule body B in Π_n. By the Lin-Zhao Theorem (Lin and Zhao 2004) or the (generalized) *Fages Theorem* (Erdem and Lifschitz 2003; You, Yuan, and Zhang 2003), Π'_n is equivalent to its completion $Comp(\Pi'_n)$. And then the Proposition follows from the fact that $Comp(\Pi'_n) = Comp(\Pi_n)$.

Lemma 3.3 (Main Lemma). *Let Π_n be a PARITY$_n$ program. Then there is a PARITY$_n$ program Π'_n s.t. Π'_n is equivalent to $Comp(\Pi'_n)$ and $|\Pi'_n| \leq |\Pi_n|$.*

Theorem 3.1 (PARITY$\notin$Poly-CP). *PARITY has no polynomial size CP representations.*

Proof. Assume the contrary that there is a sequence of PARITY$_n$ programs $\{\Pi_n\}$ s.t. $|\Pi_n|$ is bounded by a polynomial $p(n)$. By Lemma 3.3, there is a sequence of completion-equivalent PARITY$_n$ programs $\{\Pi'_n\}$ in which $|\Pi'_n|$ is also bounded by the polynomial $p(n)$. By Proposition 2.1, $\{\Pi'_n\}$ represents a language in AC^0. This contradicts PARITY$\notin \mathsf{AC}^0$. □

Corollary 3.1. *PARITY separates PF from CP.*

Corollary 3.2. *Suppose $\mathsf{P} \not\subseteq \mathsf{NC}^1/\mathsf{poly}$. Then CP and PF are succinctly incomparable.*

4 Discussion and Conclusion

In this paper we show that the PARITY problem separates PF from CP, and the two formalisms are succinctly incomparable (assuming $\mathsf{P} \not\subseteq \mathsf{NC}^1/\mathsf{poly}$). Interestingly, our main result may at first appear counter-intuitive: the P-complete problem PATH has Poly-CP representations, while this does not hold for an "easy" problem PARITY. Actually, there is no contradiction. As noted in (Abiteboul, Hull, and Vianu 1995; Dantsin et al. 2001), a *complete* problem in a complexity class can be represented in a formalism $\mathcal{C}$, *does not* imply that *all* problems in that class can be represented in $\mathcal{C}$.

Generally speaking, the research of succinctness (Gogic et al. 1995; Coste-Marquis et al. 2004; Grohe and Schweikardt 2005; French et al. 2013) gives us a deeper understanding about KR formalisms, for it reveals their (in)abilities of concisely representing different problems under the condition that the encoded models are the *same*. This is particularly interesting if the formalisms are *equally expressive* and share the same reasoning complexity.

E.g., in addition to CP and NLP, programs with *cardinality constraints* and *choice rules* (CC, without classic negation $\neg$) (Simons, Niemelä, and Soininen 2002), *(simple) definite causal theories*[4] (S/DT) (Giunchiglia et al. 2004) and

[4] A theory is simple if each rule body is a conjunction of literals.

two-valued programs (TV) (Lifschitz 2012) are as expressive as PF and NP-complete for consistency checking. But they have a non-trivial succinctness picture, see Fig. 1.

Besides the theoretical interests, succinctness also tells us something like "which for what is the best" in choosing KR formalisms for a given application. E.g., one should choose ASP instead of SAT (or DT) if the application involves reasoning about PATH or Transitive Closure[5], because the former provides compact representations to avoid unnecessary overload. Recall that from the complexity viewpoint, even *one* extra variable may *double* the search space for intractable problems.

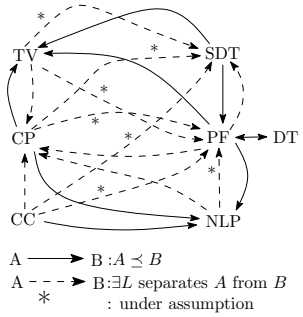


Figure 1: Succinctness Pic.

In future work we plan to establish the missing connections in Fig. 1, moreover, we will consider the succinctness of more expressive formalisms like programs with predicate symbols or higher-order atoms (Gebser, Schaub, and Thiele 2007), etc.

Acknowledgement

We are grateful to the anonymous reviewers for their valuable comments. Thanks to Shiguang Feng, Yan Zhang, Jiankun He, Guangrui Dang and Xiaolong Liang for their helpful discussions. The research was partially supported by NSFC Grant 61272059, MOE Grant 11JJD720020, NSSFC Grant 13&ZD186 and the Fundamental Research Funds for the Central Universities Grant 1409025.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Arora, S., and Barak, B. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2009. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Communications of the ACM* 54(12):92–103.

Coste-Marquis, S.; Lang, J.; Liberatore, P.; and Marquis, P. 2004. Expressive power and succinctness of propositional languages for preference representation. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning*, 203–212. AAAI Press.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys* 33(3):374–425.

[5] An NL-*complete* problem. It is believed that $\mathsf{NL} \not\subseteq \mathsf{NC}^1/\mathsf{Poly}$.

Erdem, E., and Lifschitz, V. 2003. Tight logic programs. *Theory and Practice of Logic Programming* 3(4):499–518.

French, T.; van der Hoek, W.; Iliev, P.; and Kooi, B. P. 2013. On the succinctness of some modal logics. *Artificial Intelligence* 197:56–85.

Gebser, M.; Schaub, T.; and Thiele, S. 2007. Gringo: A new grounder for answer set programming. In *Proceedings of the 9th International Conference of Logic Programming and Nonmonotonic Reasoning (LNCS4483)*, 266–271. Springer.

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; Turner, H.; and Lifschitz, J. L. V. 2004. Nonmonotonic causal theories. *Artificial Intelligence* 153(1-2):49–104.

Gogic, G.; Kautz, H. A.; Papadimitriou, C. H.; and Selman, B. 1995. The comparative linguistics of knowledge representation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 862–869. Morgan Kaufmann Publishers Inc.

Grohe, M., and Schweikardt, N. 2005. The succinctness of first-order logic on linear orders. *Logical Methods in Computer Science* 1(1).

Jukna, S. 2012. *Boolean Function Complexity - Advances and Frontiers*. Springer.

Lee, J.; Lifschitz, V.; and Yang, F. 2013. Action language $\mathcal{BC}$: Preliminary report. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 983–989. AAAI Press.

Lee, J. 2005. A model-theoretic counterpart of loop formulas. In *Proceedings of the 19th international joint conference on Artificial intelligence*, 503–508. Morgan Kaufmann Publishers Inc.

Lifschitz, V., and Razborov, A. 2006. Why are there so many loop formulas? *ACM Transactions on Computational Logic* 7(2):261–268.

Lifschitz, V. 1999. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence* 25:369–389.

Lifschitz, V. 2008. What is answer set programming? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1594–1597. MIT Press.

Lifschitz, V. 2012. Two-valued logic programs. In *Technical Communications of the 28th International Conference on Logic Programming*, 259–266. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Lin, F., and Zhao, Y. 2004. ASSAT: computing answer sets of a logic program by SAT solvers. *Artificial Intelligence* 157(1 - 2):115 – 137.

Simons, P.; Niemelä, I.; and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artificial Intelligence* 138(1-2):181–234.

You, J.-H.; Yuan, L.-Y.; and Zhang, M. 2003. On the equivalence between answer sets and models of completion for nested logic programs. In *Proceedings of the 18th international joint conference on Artificial intelligence*, 859–864. Morgan Kaufmann Publishers Inc.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Heuristic Guided Optimization for Propositional Planning

Andreas Sideris and **Yannis Dimopoulos**
Department of Computer Science
University of Cyprus
Nicosia, Cyprus

Abstract

Planning as Satisfiability is an important approach to Propositional Planning. A serious drawback of the method is its limited scalability, as the instances that arise from large planning problems are often too hard for modern SAT solvers.

This work tackles this problem by combining two powerful techniques that aim at decomposing a planning problem into smaller subproblems, so that the satisfiability instances that need to be solved do not grow prohibitively large. The first technique, *incremental goal achievement*, turns planning into a series of boolean optimization problems, each seeking to maximize the number of goals that are achieved within a limited planning horizon. This is coupled with a second technique, called *heuristic guidance*, that directs search towards a state which satisfies all goals.

Introduction

Planning as Satisfiability (Kautz and Selman 1996; Kautz, Selman, and Hoffmann 2006) is an important method for optimal propositional planning. One of the advantages of the method is that it finds plans of optimal makespan. However, it often happens that the instances generated from planning problems, grow so large that they are unsolvable even by the most advanced SAT solvers.

This motivated a number of recent attempts to improve scalability at the cost of sacrificing the optimality of the generated solutions. One such example is `Madagascar` (Rintanen 2012), that enhanced a SAT solver with a planning specific variable selection heuristic, to generate suboptimal plans quickly. In a different spirit, the `PSP` system (Sideris and Dimopoulos 2012), introduced the idea of *planning as optimization*. `PSP` employs the increasing planning horizon technique, but unlike classic `SATPLAN`, it maximizes the goals that are achieved within each horizon, and adds them as intermediate goals in the problem model. However, as the plan length can grow arbitrarily long, at some point it is expected that both approaches will be confronted with performance difficulties.

In this work we take a different path. The basic idea is to split a planning problem into smaller subproblems, of a size that is not prohibitive for the underlying solver. `PSP-H`, the new planning system that is described here, shares with `PSP` the planning as optimization perspective, but it differs in a number of important ways. The first is the *incremental goal achievement* which, at a high level, works as follows. `PSP-H` first generates a sub-plan that, starting from the initial state, achieves a subset, of predefined size, of the problem goals. The state that results after the execution of the actions of this sub-plan, becomes the new initial state, and a new sub-plan that satisfies a larger subset of goals is computed. The procedure iterates and links together the subplans that are generated along the way. Therefore, instead of solving the original planning problem, `PSP-H` solves a series of smaller subproblems.

A downside of this greedy approach is that it focuses on maximizing the number of achieved goals in a limited planning horizon, and it ignores completely goals that cannot be achieved within this horizon. In order to overcome the limitations that this would place on the effectiveness of the system, `PSP-H` is enhanced by a second technique called *heuristic guidance*, which imposes an additional requirement on the intermediate states that are computed by the `PSP-H` algorithm. The property that these states need to satisfy is that there must be a *relaxed* plan from each such state to the final state. `PSP-H` employs three different relaxation methods that are all based on ignoring some of the problem constraints, but they differ in their strength.

The first relaxation method is the well-known delete lists relaxation, whereas the second method ignores all action mutexes. The last relaxation, which is stronger than the first but weaker than the second, ignores action mutexes and uses a subset of the fact mutexes that are heuristically selected.

`PSP-H` is a incomplete and suboptimal planner implemented on top of the `PSP` system. Our experimental evaluation on a number of domains taken from planning competitions, demonstrates that `PSP-H` can solve challenging problems, that require long plans. Moreover, a preliminary comparison with `Madagascar` shows that the new system can solve more problems in some domains, and generate better quality solutions both in term of plan length and number of actions.

Some of the techniques employed in `PSP-H` are similar to those used in other planning systems. For instance, the local optimization method of `PSP-H` bares some resemblance to the enforced hill-climbing approach of the

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

FF planner (Hoffmann and Nebel 2001). The CPT system (Vidal and Geffner 2006) combines heuristic estimators with constraint programming techniques in the context of optimal temporal planning. Finally, a study of the relation between the relaxation methods of PSP−H and the red-black heuristics of (Katz, Hoffmann, and Domshlak 2013a; 2013b) may lead to a tighter integration of heuristic and constrained optimization based planning.

Background

We assume familiarity with the planning as satisfiability framework. We discuss in brief pseudo-boolean optimization and the planning as optimization method of PSP.

A (STRIPS) planning problem is a triple $P =< I, G, \mathcal{A} >$, where I is the set of facts that hold in the initial state, G are the goals, and $\mathcal{A}$ is a set of actions. Each action has preconditions, add effects, and delete effects. In the SAT model of a planning problem, time-stamped propositional atoms (or variables) represent the action and facts of the problem. An atom $A(T)$, where A is an action, corresponds to the decision of whether action A is taken or not at time T, and analogously for variables of the form $f(T)$, where f is a fact.

PSP−H uses the SMP (Sideris and Dimopoulos 2010) encodings of a planning problem, that derives its clauses from the *planning graph* (Blum and Furst 1997) of the problem. Among the clauses of the SMP model there are those that correspond to fact and action *mutexes*, that are derived from the planning graph.

A *(Linear) Pseudo-boolean constraint* (PB-constraint) is an inequality of the form $\sum a_i x_i \geq b$, where x_i is a boolean variable, and a_i, b integers. A *Pseudo-boolean optimization* (PBO) problem on a set of variables X consists of a set of PB-constraints on X, together with a linear function on boolean variables from X. In PBO we seek a value assignment on the variables of X that maximizes the value of the boolean function, called objective function, and satisfies all the PB-constraints. The current implementation of PSP−H uses minisat+ (Eén and Sörensson 2006), to translate PB-constraints into CNF, and then invokes precosat (Biere 2009), that solves the SAT instance.

PSP−H is implemented as an extension of the PSP planning system (Sideris and Dimopoulos 2012). PSP follows a similar to the classic solve and expand approach, with the difference that it seeks to maximize the number of goals that can be achieved within a fixed planning horizon of, say, k steps. If this number is smaller than some user supplied value, the planning horizon is extended to $k + 1$ steps, and the procedure iterates. Moreover, the goals that are attained within the k steps are added to the problem as *intermediate goals*, that must be established in the $k + 1$ steps long plan that is generated in the next iteration. For a more detailed description of PSP the reader is referred to (Sideris and Dimopoulos 2012).

The PSP−H algorithm

PSP−H takes as input a (STRIPS) planning problem $(I, G, \mathcal{A})$, and a number of parameter values in the range [0..1]. The main technique implemented in PSP−H is the *incremental goal achievement*, that is embedded in its optimization phase. The core of the computation that is carried out in this phase is the solution of a series of PBO problems, where the maximization objective is $g_1(T)+\ldots+g_l(T)$, where $g_1, \ldots, g_l$ are the goal atoms and T the planning horizon. Obviously, the highest value that can be assumed by the objective function is l, which we denote by f_{max}.

PSP−H is also supplied by the user with an objective function value threshold $thres$. It starts with an initial planning horizon o that is determined by information that is extracted from the underlying planning graph as in the classic SATPLAN system. PSP−H solves the PBO with horizon o and obtains an optimal value f for the objective function. If $f < thres * f_{max}$, the planning horizon is set to $o + 1$, and the procedure repeats. If for some horizon o_b a solution is found with a value f_b for the objective function such that $f_b \geq thres * f_{max}$, a $plan$ is extracted from the solution. At this point, the state I' that results after the execution of the actions in $plan$ becomes the new initial state, and PSP−H repeats the process with a new planning problem $P' =< I', G, \mathcal{A} >$, where $P =< I, G, \mathcal{A} >$ and I are the original problem and initial state respectively. Moreover, the objective value threshold $thres$ is increased and set to $thres+restart\_rate$, where $restart\_rate$ is a user supplied value, called *restart rate*. As the maximization objective is defined on the fact variables that correspond to goals, the increase in the threshold translates into an increase in the number of goals that are achieved. Hence the term *incremental goal achievement*.

The optimization phase of PSP−H terminates when the value $thres$ exceeds another user supplied value $finphase$. Its output is a plan that is formed by concatenating the partial plans that are generated during its iterations. The execution of the action in this plan defines a state I_{sat}. Then PSP−H, similarly to PSP, enters the satisfaction phase, where a new planning problem with initial state I_{sat}, is solved by SMP.

The periodic replacement of the initial state of the planning problem described above, is called *restart strategy*, where the value of the $restart\_rate$ parameter determines the frequency of the restarts. With the restart strategy PSP−H attempts to limit the length of the planning horizon, and hence the size of the satisfiability instances that are solved by the system. As it is often the case with greedy search methods, a difficulty with incremental goal achievement is that it tends to generate locally optimal plans, i.e. plans that maximize the number of attained goals in the limited planning horizon, and ignores goals that are not achievable within that horizon. This bias may lead to poor quality solutions, as well as insolvability when actions interact in complex ways, as in the case of the presence of non-renewable resources.

To tackle this problem, PSP−H combines incremental goal achievement with a heuristic method that is intended to

account for the effort that is needed to achieve the remaining goals. This is accomplished by a problem model that consists of the usual layers of action and fact propositions that correspond to different times/points, but it is divided in two parts: the *constrained* and the *relaxed* part.

The constrained part is the standard PB model of the problem, i.e. a straightforward translation of the SMP clauses into linear inequalities. The relaxed part is also obtained from the SMP encoding, but some of the constraints of the model are omitted. In fact, as depicted in Figure 1, the relaxed part is further divided into three segments that differ in the mutual exclusion information they contain.

The first segment of the relaxed part is the *action relaxation*, which is the SMP model without action mutex clauses, and it is similar to the "$\exists$ encoding" of (Rintanen, Heljanko, and Niemelä 2006). The last segment is the *full relaxation* that contains neither action nor fact mutexes. This segment corresponds to the well-known delete-lists relaxation heuristic that forms the basis of many heuristic search planners. All goals are added as unit clauses in a final layer $q + 1$ (layer numbering is taken from figure 1).

Between the above two segments lies the *intermediate relaxation* segment. It is less constrained (more relaxed) than the action relaxation, but more constrained (less relaxed) than the full relaxation. It contains no action mutexes, and only a subset of the fact mutexes that is heuristically selected by PSP−H. This selection process is based on the analysis of the multi-valued variable representation of the problem that is carried out with the help of the preprocessor of LAMA (Richter and Westphal 2010). For some problem variables all corresponding mutexes are omitted, for others all mutexes are added, whereas for some variables x with domain values $x_1, \ldots, x_n$, instead of the constraint $\sum_{i=1}^{n} x_i <= 1$, PSP−H adds the weaker inequality $\sum_{i=1}^{n} x_i <= k$, for heuristically selected values of k. Due to space limitation, a more complete description of the intermediate relaxation falls outside the scope of this paper.

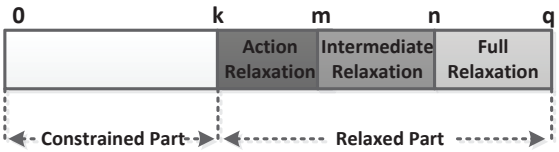


Figure 1: The different part of the encoding in PSP−H.

From the above discussion it must have become apparent that the problem model becomes less constrained as we move from left to right in figure 1. All the constraints of the different parts must be satisfied during the optimization phase of PSP−H, while the objective function that is maximized is $g_1(k) + \ldots + g_l(k)$, i.e. the goals refer to the last layer of the constraint part. The action variables of the first k layers (ie. the constrained part) that are assigned true in a solution generated by PSP−H in an iteration of the optimization phase, form a valid plan that achieves some of the goals at time k. The execution of the actions of this plan defines a valid state s_k at time k. As noted in the beginning of this section, this state s_k may become the initial state of the new problem that is solved in the next iteration of the optimization phase. The PSP−H encoding ensures that a state that satisfies all goals is reachable from s_k within $q + 1 - k$ *relaxed* steps. Therefore, PSP−H searches for a state s_k for which there is not only a valid plan from the current initial state, but there is also a relaxed, heuristically specified, plan from s_k to a final state. Hence the term *heuristic guided optimization*.

The computation of the length of the full relaxation segment takes into account a user supplied value for parameter h_init_perc, and is carried out as follows. PSP−H first removes all delete-lists from all problem actions, and then builds a planning graph until all goals are reachable. If the number of layers of this graph is len, the length of the relaxation part (corresponding to value $q - n$ in figure 1) is set to $len * h\_init\_perc$.

The length of the other segments of the relaxed part are computed in a similar way. First, PSP−H builds a planning graph of the original problem, and extends it until all goals are reachable. If the graph has len layers, the length of the action relation part is set to $len * r\_init\_perc1$ and the intermediate relaxation part to $len * r\_init\_perc2$, where again r_init_perc1 and r_init_perc2 are input parameter values.

In order to enhance the effects of heuristic guided optimization, PSP−H shortens the relaxed part periodically as the planner approaches a final state. The reduction always starts with the full relaxation segment, while its rate is determined by an input parameter h_red_rate. Roughly speaking, each time that the value of the objective function that is maximized in the optimization phase improves by h_red_rate, the full relaxation segment is reduced by one step. When its length reaches the value 1, the intermediate relaxation segment is reduced in a similar way. Action relaxation is never reduced.

Experimental evaluation

PSP−H is a prototype system that has been implemented to test the ideas that are presented in this work. It has been built on top of SATPLAN, from which it inherits the very slow and memory demanding planning graph construction phase. This imposes serious limitations on the size of the problem that can be solved by PSP−H. In fact, in many domains the current version of PSP−H spends more time in manipulating data structures than in constraint solving and optimization.

PSP−H has been evaluated on a number of domains from various planning competitions and compared with LAMA and Madagascar. All experiments were run on a server with 24 X5690 cores at 3.47GHz running under CentOS. Both LAMA and Madagascar were run with their default parameter values, and a CPU cutoff limit of 3600 seconds. In all experiments with PSP−H, the optimization phase terminated after 300 CPU seconds, in which case the planning horizon was extended by one step. PSP−H was also run with a CPU limit of 3600 seconds.

Table 1 summarizes the number of problems solved by each system. The first column lists the domains of the ex-

Domain	Num. of Probl.	LAMA	Madag	PSP-H
Pipesworld	25	25	22	25
Satellite	22	22	22	22
Storage	29	20	29	29
Openstacks	30	30	18	18
Elevators	30	30	30	30
Transport	19	19	19	18
Visitall	28	28	19	21
Barman	40	40	28	40
Total	223	214	187	202

Table 1: Number of problems solved by the planners within a 3600 seconds CPU limit. The second column lists the number of problems tried in each domain.

perimental evaluation, with the name with which they are known in the literature. The second column lists the number of problems included from each domain, as it was determined by PSP-H's memory consumption. Problems which exceed the memory capabilities of PSP-H are excluded from the study. The third and fourth column present the problems solved within the CPU cutoff limit by LAMA and Madagascar respectively. The last column refers to a characteristic run of PSP-H (the specific parameter values of this run are not of particular interest) that feature frequent restarts, thus making the satisfiability problems easier for the underlying boolean optimizer.

A comparison of PSP-H's column with that of LAMA, clearly shows that the heuristic planner outperforms PSP-H. In fact, the only domain where PSP-H (and Madagascar) solves more problems is Storage. On the other hand, PSP-H can solve more problems than Madagascar in Pipesworld, but its advantage shows more clearly in the Visitall and Barman domains, and this is not accidental. The plans in these domains are long, and it seems that the techniques employed by PSP-H address successfully this problem, at least in certain cases. We note that the longest plan that PSP-H was able to synthesize comes from Visitall and was 248 steps long.

Table 2 presents results about the quality of the plans that were generated by the systems for the largest 5 problems in each domain that were solved by all planners. The numbers in parentheses are the sums of the number of actions in all the 5 problems, whereas the numbers outside parentheses sum the parallel plan lengths, and hence they are not meaningful for LAMA which generates sequential plans. Clearly, PSP-H generates better quality plans in all domains except Satellite. On the other hand, in some domains with low or no action parallelism, such as Openstacks, Visitall and Barman, PSP-H generates plans that have less actions than those of LAMA. However, in domains with high parallelism, LAMA's plans contain considerably fewer actions.

Domain	LAMA	Madag	PSP-H
Pipesworld	(167)	193 (432)	152 (281)
Satellite	(366)	99 (457)	109 (527)
Storage	-	450 (1339)	194 (725)
Openstacks	(662)	448 (606)	396 (572)
Elevators	(839)	428 (2056)	382 (1629)
Transport	(341)	432 (1161)	220 (666)
Visitall	(448)	535 (535)	447 (447)
Barman	(814)	896 (1184)	561 (780)
Total	(3637)	3031 (6431)	2267 (4902)

Table 2: Sums of the solution length (number of parallel steps) and in parentheses the sums of the number of actions for the largest five problems in each domain. In the last line the total counts for all domains except Storage.

References

Biere, A. 2009. P{re,ic}oSAT@SC'09. In *SAT Competition 2009*, http://fmv.jku.at/precosat/.

Blum, A., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artif. Intell.* 90(1-2):281–300.

Eén, N., and Sörensson, N. 2006. Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation* 2(1-4):1–26.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Int. Res.* 14(1):253–302.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013a. Red-black relaxed plan heuristics. In *AAAI*, 489–495.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013b. Who said we need to relax all variables? In *ICAPS*, 126–134.

Kautz, H. A., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic and stochastic search. In *AAAI*, 1194–1201.

Kautz, H. A.; Selman, B.; and Hoffmann, J. 2006. SATPLAN: Planning as satisfiability. In *Booklet of the 5th Planning Competition*.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res. (JAIR)* 39:127–177.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artif. Intell.* 170(12-13):1031–1080.

Rintanen, J. 2012. Planning as satisfiability: Heuristics. *Artif. Intell.* 193:45–86.

Sideris, A., and Dimopoulos, Y. 2010. Constraint propagation in propositional planning. In *ICAPS*, 153–160.

Sideris, A., and Dimopoulos, Y. 2012. Propositional planning as optimization. In *ECAI*, 732–737.

Vidal, V., and Geffner, H. 2006. Branching and pruning: An optimal temporal POCL planner based on constraint programming. *Artif. Intell.* 170(3):298–335.

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

First-Order Default Logic Revisited

Yi Zhou
Artificial Intelligence Research Group
School of Computing, Engineering and Mathematics
University of Western Sydney, Australia

Abstract

Reiter's original proposal for default logic is unsatisfactory for open default theories because of Skolemization and grounding. In this paper, we reconsider this long-standing problem and propose a new world view semantics for first-order default logic. Roughly speaking, a world view of a first-order default theory is a maximal collection of structures satisfying the default theory where the default part is fixed by the world view itself. We show how this semantics generalizes classical first-order logic and first-order answer set programming, and we discuss its connections to Reiter's semantics and other related semantics. We also argue that first-order default logic under the world view semantics provides a rich framework for integrating classical logic based and rule based formalisms in the first-order case.

Introduction

As a predominant approach for nonmonotonic reasoning, default logic has attracted many researchers since Reiter's seminal work (Reiter 1980). Reiter's semantics targets on general first-order default theories in two steps. First, an extension semantics is defined for closed default theories in which no free variable occurs in any default rules. Then, general default theories with free variables (also called open default theories) are converted into closed ones by Skolemization and grounding so that the above extension semantics can be applied.

However, as pointed out by Poole (1987) and many other researchers (Lifschitz 1990; Baader and Hollunder 1995; Kaminski 1995; 1999), Reiter's semantics may lead to counterintuitive phenomena for open default theories. For instance, Lifschitz (1990) proposed an example with a single default rule $\frac{:M\neg P(x)}{\neg P(x)}$ and a single first-order assertion $P(a)$. One should expect that we have the following information hidden in the default theory

$$\forall x\,(P(x) \leftrightarrow (x = a)).$$

However, in Reiter's extension semantics, the default rule will only be grounded on a and another Skolem constant c different than a. Therefore, one can only conclude $P(a) \wedge \exists x\,\neg P(x)$.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The unsatisfactory performance of Reiter's original semantics for open first-order default theories is indeed a serious semantic issue. As pointed out by Reiter himself (1980),

"the genuinely interesting cases involve open defaults."

In this paper, we reconsider this long standing problem and propose a new world view semantics. Instead of first-order theories, we use collections of structures sharing the same domain and function interpretations (called *views*) as the candidate solution concept. Roughly speaking, a world view of a first-order default theory is a maximal view (collection of structures) satisfying the default theory where the default part is fixed by the world view itself. Although related, this semantics is essentially different from Reiter's extension semantics, even for closed default theories. However, they coincide on some restricted classes.

Our work is also strongly motivated from the attempt to combine classical logic based formalisms and rule based formalisms, particularly the recent development in ontology engineering for adding rules onto the description logics layer (Baader and Hollunder 1995; Motik et al. 2006; Eiter et al. 2008; Motik and Rosati 2010; Lukasiewicz 2010; Zhou and Zhang 2012). We argue that default logic provides a rich framework for this task. First, we show that it generalizes both classical logic and first-order answer set programming. Then, we show how default logic can handle both monotonic reasoning and nonmonotonic reasoning and flexibly switch between open world reasoning and closed world reasoning.

The main contribution of this paper is to provide a new world view semantics for first-order default logic. Although this is a long standing problem, we believe that research in this direction will further deepen our understandings and shed new insights in

- default logic and nonmonotonic reasoning,
- dealing with free variables in complex knowledge representation formalisms, and
- integration of classical logic based and rule based formalisms in the first-order case.

Syntax

We assume that the readers are familiar with the basic notions and notations in classical first-order logic. A (first-

order) *default rule* (*rule* for short if clear from the context) is of the following form

$$\alpha_1 \mid \ldots \mid \alpha_n \leftarrow \beta_1, \ldots, \beta_m, \texttt{Not}\ \gamma_1, \ldots, \texttt{Not}\ \gamma_l, \quad (1)$$

where $\alpha_i, 1 \leq i \leq n$, $\beta_j, 1 \leq j \leq m$ and $\gamma_k, 1 \leq k \leq l$ are first-order formulas. A *default theory* is a set of rules of the form (1).

Intuitively, rule (1) means that if an agent believes that every $\beta_j, 1 \leq j \leq m$ is true, and the agent does not believe that any of $\gamma_k, 1 \leq k \leq l$ is true, then the agent must believe that at least one of $\alpha_i, 1 \leq i \leq n$ is true.

Given a rule r of the form (1), the set $\{\alpha_1, \ldots, \alpha_n\}$ is called the *head* of r, denoted by $Head(r)$; the set $\{\beta_1, \ldots, \beta_m\}$ is called the *positive body* of r, denoted by $Pos(r)$; the set $\{\texttt{Not}\ \gamma_1, \ldots, \texttt{Not}\ \gamma_l\}$ is called the *negative body* of r, denoted by $Neg(r)$;[1] also the set $\{\beta_1, \ldots, \beta_m, \texttt{Not}\ \gamma_1, \ldots, \texttt{Not}\ \gamma_l\}$ is called the *body* of r, denoted by $Body(r)$. Clearly, $Body(r) = Pos(r) \cup Neg(r)$.

Notice that the form (1) follows the well adopted syntactic form of rules in logic programming (in particular answer set programming), which actually differs from Reiter's original syntax of default rules as follows

$$\frac{\alpha(\overrightarrow{x}) : M\beta_1(\overrightarrow{x}), \ldots, M\beta_m(\overrightarrow{x})}{\omega(\overrightarrow{x})}, \quad (2)$$

where $\alpha(\overrightarrow{x})$, $\beta_i(\overrightarrow{x}), 1 \leq i \leq m$ and $\omega(\overrightarrow{x})$ are formulas and $\overrightarrow{x}$ is the set of free variables occurred in all these formulas. Intuitively, a rule of form (2) means that if an agent believes that $\alpha(\overrightarrow{x})$ is true and her/his beliefs are consistent with any of $\beta_j, 1 \leq j \leq m$, then the agent must believe that $\omega(\overrightarrow{x})$ is true. In addition, in Reiter's original syntax, a default theory is a pair (D, W), where D is a set of rules of the form (2) and $W = \{S_1, \ldots, S_k, \ldots\}$ is a set of sentences.

Our syntax is actually a generalization of Reiter's syntactic form in the sense that a rule of the Reiter's form

$$\frac{\alpha(\overrightarrow{x}) : M\beta_1(\overrightarrow{x}), \ldots, M\beta_m(\overrightarrow{x})}{\omega(\overrightarrow{x})}, \quad (3)$$

can be rewritten as

$$\omega(\overrightarrow{x}) \leftarrow \alpha(\overrightarrow{x}), \texttt{Not}\ \neg\beta_1(\overrightarrow{x}), \ldots, \texttt{Not}\ \neg\beta_m(\overrightarrow{x}) \quad (4)$$

in form (1), and a sentence $S \in W$ in a default theory (D, W) can be simply rewritten as

$$S \leftarrow$$

in form (1). Notice that the component $M\beta_j(\overrightarrow{x})$ in form (2) is actually corresponding to $\texttt{Not}\ \neg\beta_j(\overrightarrow{x})$ but not $\texttt{Not}\ \beta_j(\overrightarrow{x})$ in (4). Here, $\neg$ is the classical negation operator.

We say that a rule r of the form (1) is a *fact* if $Pos(r) = Neg(r) = \emptyset$, a *constraint* if $Head(r) = \emptyset$, *positive* if $Neg(r) = \emptyset$, a *logic programming rule* if all formulas occurred in r (i.e. $\alpha_i, 1 \leq i \leq n$, $\beta_j, 1 \leq j \leq m$, $\gamma_k, 1 \leq k \leq l$) are atoms, *closed* if all formulas occurred in r are closed.

We say that a default theory is a *classical theory* if every rule in it is a fact, a *logic program* if every rule in it is a logic programming rule, *normal* if every rule in it is normal, *closed* if every rule in it is closed.

[1]The head, the positive body and the negative body of a default rule are also called consequent/conclusion, prerequisite and justification respectively in the literature.

Defining the World View Semantics

We start to define the world view semantics for first-order default logic.

Definition 1 (Configuration). Let τ be a vocabulary. A *configuration* of τ is a tuple $\Delta = (A, c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}, f_1^{\mathcal{A}}, \cdots, f_l^{\mathcal{A}})$, where A is the domain, $c_1^{\mathcal{A}}, \cdots, c_m^{\mathcal{A}}$ are the constant mappings and $f_1^{\mathcal{A}}, \cdots, f_l^{\mathcal{A}}$ are the function interpretations.

A configuration Δ can be extended into a structure $\mathcal{M}$ by interpreting the predicates. In this case, we say that Δ is the configuration of $\mathcal{M}$, and conversely, $\mathcal{M}$ is a structure on Δ. We use $\Delta^{\mathcal{M}}$ to denote the configuration of $\mathcal{M}$. Then notion of configuration is analogous to the notion of pre-interpretation introduced by Kaminski (1995).

Definition 2 (View). A *view* of a vocabulary τ is a collection of τ-structures with the same configuration, i.e., sharing the same domain, constant mappings and function interpretations. More precisely, let Δ be a configuration, a view $\mathcal{W}$ on Δ is a collection $\mathcal{M}_1, \ldots, \mathcal{M}_t, \ldots$ of structures such that for all t, $\Delta^{\mathcal{M}_t} = \Delta$.

Here, we say that Δ is the configuration of $\mathcal{W}$. Sometimes we use $\Delta^{\mathcal{W}}$ to denote the configuration of $\mathcal{W}$ for better readability. Intuitively, a view represents an agent's belief in the sense that each structure in the view is corresponding to a possible world.

We now define the satisfaction relation between views associated with assignments and default rules. Let r be a rule of the form (1), $\mathcal{W} = \{\mathcal{M}_1, \ldots, \mathcal{M}_t, \ldots\}$ a view and η an assignment. We say that $\mathcal{W}$ *satisfies* r on η, denoted by $\mathcal{W} \models r[\eta]$, if

(†) $\mathcal{W} \models Pos(r)[\eta]$ and $\mathcal{W} \not\models Neg(r)[\eta]$ implies that $\mathcal{W} \models Head(r)[\eta]$.

As the default part (i.e. the negative body) of a default rule plays a role of assumption, we also need a different notion to define the satisfaction relation by fixing this part. Let r be a rule of the form (1), $\mathcal{W} = \{\mathcal{M}_1, \ldots, \mathcal{M}_t, \ldots\}$ and $\mathcal{W}' = \{\mathcal{M}'_1, \ldots, \mathcal{M}'_t, \ldots\}$ two views, and η an assignment. We say that $\mathcal{W}'$ *satisfies* r on η by fixing the default part with $\mathcal{W}$, denoted by $(\mathcal{W}', \mathcal{W}) \models r[\eta]$, if

(‡) $\mathcal{W}' \models Pos(r)[\eta]$ and $\mathcal{W} \not\models Neg(r)[\eta]$ implies that $\mathcal{W}' \models Head(r)[\eta]$.

Let Π be a default theory, η an assignment and $\mathcal{W}$ and $\mathcal{W}'$ two views. We say that $\mathcal{W}$ *satisfies* Π on η, denoted by $\mathcal{W} \models\!\!\models \Pi[\ \eta]$, if for all $r \in \Pi$, $\mathcal{W} \models r[\eta]$. We say that $\mathcal{W}$ *satisfies* Π, denoted by $\mathcal{W} \models\!\!\models \Pi$, if for all assignments η, $\mathcal{W} \models\!\!\models \Pi[\ \eta]$. We say that $\mathcal{W}'$ *satisfies* Π on η by fixing the default part with $\mathcal{W}$, denoted by $(\mathcal{W}', \mathcal{W}) \models\!\!\models \Pi[\ \eta]$, if for all $r \in \Pi$, $(\mathcal{W}', \mathcal{W}) \models r[\eta]$. We say that $\mathcal{W}'$ *satisfies* Π by fixing the default part with $\mathcal{W}$, denoted by $(\mathcal{W}', \mathcal{W}) \models \Pi$, if for all assignments η, $(\mathcal{W}', \mathcal{W}) \models\!\!\models \Pi[\ \eta]$.

The following property holds straightforwardly from the definitions.

Proposition 1. *Let Π be a default theory and $\mathcal{W}$ a view. Then, $(\mathcal{W}, \mathcal{W}) \models \Pi$ iff $\mathcal{W} \models \Pi$.*

Definition 3 (World View). Let Π be a default theory and $\mathcal{W}$ a view. We say that $\mathcal{W}$ is a *world view* of Π if $\mathcal{W} \models \Pi$

and there does not exist another view $\mathcal{W}'$ such that $\mathcal{W} \subset \mathcal{W}'$ and $(\mathcal{W}', \mathcal{W}) \models \Pi$.

Intuitively, a world view of a default theory is a maximal collection of structures satisfying the default theory, where the default part is fixed by the world view itself. Definition 3 can be presented in an alternative way closer to Reiter's original fixed-point definition. Let Π be a default theory and $\mathcal{W}$ a view such that $\mathcal{W} \models \Pi$, by $\Gamma(\mathcal{W})$, we denote the maximal view (in terms of set inclusion) satisfying the following conditions:

- $\mathcal{W} \subseteq \Gamma(\mathcal{W})$.
- $(\Gamma(\mathcal{W}), \mathcal{W}) \models \Pi$.

Notice that a drawback of this definition is that one needs to assure that such a $\Gamma(\mathcal{W})$ always exists (although this is indeed the case under our context). Clearly, $\mathcal{W}$ is a world view of Π iff $\mathcal{W}$ is the fixed point of the operator Γ, i.e., $\mathcal{W} = \Gamma(\mathcal{W})$.

The world view semantics has many origins from the literature. The term world view is borrowed from (Gelfond 1994), although the semantics is defined in a very different manner. Definition 3 actually follows the well known Gelfond-Lifschitz reduct semantics for propositional answer set programming, that is, an answer set of a propositional logic program is a maximal set of propositional atoms satisfying the reduct of the program relative to this set. Here, $(\mathcal{W}', \mathcal{W}) \models \Pi$ is analogous to $\mathcal{W}' \models \Pi^{\mathcal{W}}$, where $\Pi^{\mathcal{W}}$ is the reduct of Π relative to $\mathcal{W}$. However, we cannot use the notation of reduct $\Pi^{\mathcal{W}}$ here in our world view semantics as it is not a real program or default theory.

Of course, Definition 3 shares some basic ideas of Reiter's original semantics. On one side, the default part plays a role of assumptions, which is fixed by a candidate. On the other side, the world view semantics can also be defined in a fixed point style as shown before. However, the main difference between the world view semantics and Reiter's original semantics is that the solution concept for the former is defined upon views (i.e. collections of structures), while the one for the latter is defined upon first-order theories (i.e. collections of sentences). Note that collections of structures and first-order theories correspond to each other in the sense that all the models of a first-order theory form a collection of structures. We shall discuss later in the paper why we need to define upon collections of structures rather than collections of sentences for first-order open default theories, and the detailed relationships between the world view semantics and Reiter's original one.

Relationships to Reiter's Semantics

In this section, we study the relationships between the world view semantics and Reiter's extension semantics (Reiter 1980). Clearly, for open default theories, the world view semantics is very different from Reiter's extension semantics. In fact, this is indeed the case even for closed default theories as well. The key reason why these two semantics behave quite differently, even for closed default theories, is that all views are forced to have the same configuration. This is because a configuration itself includes some information, for instance, the domain size and the mapping of terms.

Nevertheless, these two semantics coincide if we only consider the first-order sentences whose satisfiability is independent of a configuration. We say that a first-order sentence ϕ is *configuration independent* if that ϕ is consistent implies that ϕ has a model on every configuration. We say that a closed default theory is *configuration independent* if all formulas occurring in every rule in the default theory are configuration independent.

Theorem 1. *Let Π be a configuration independent closed default theory. If a first-order theory E is an extension of Π, then for any configuration Δ, the set of models of E on Δ is a world view of Π. Conversely, if a view $\mathcal{W}$ on a configuration Δ is a world view of Π, then there exists an extension E of Π such that $\mathcal{W}$ is the set of models of E on Δ.*

Corollary 2. *Let Π be a propositional default theory. Then, a formula F is equivalent to an extension of Π iff the set of models of F is a world view of Π.*

Classical First-Order Logic + First-Order Answer Set Programming

In this section, we show how the world view semantics generalizes both classical first-order logic and first-order answer set programming. We also argue that first-order default logic provides a rich framework for integrating them.

Theorem 3. *Let S be a set of first-order sentences and Δ a configuration such that S has models on Δ. Then, S has a unique world view on Δ, which is the class of all models of S on Δ.*

Next, we show that the stable model semantics for first-order answer set programming is a special case of the world view semantics for first-order default theories as well. A first-order logic program is a special case of first-order default rules except that the formulas occurring in rules are atoms. The semantics for first-order logic programs can be defined through by a transformation into second-order logic (Ferraris, Lee, and Lifschitz 2011; Zhou and Zhang 2011).

Theorem 4. *Let Π be a logic program. If a structure $\mathcal{M}$ is a stable model of Π, then $\mathcal{M}^+$ is a world view of Π, where $\mathcal{M}^+$ contains all structures on the same configuration bigger than $\mathcal{M}$.[2] Conversely, if a view $\mathcal{W}$ is a world view of Π, then the structure on the same configuration that interprets each predicate P as $\bigcap_{\mathcal{M} \in \mathcal{W}} P^{\mathcal{M}}$ is a stable model of Π.*

Theorems 3 and 4 show that both classical first-order logic and first-order answer set programming can be considered as special cases of first-order default logic. Moreover, we argue that default logic not only generalizes classical logic and answer set programming but also integrates them in a natural way. In the propositional case, propositional default logic provides an ideal framework for integrating classical propositional logic and propositional answer set programming. Lifted into the first-order case, first, the syntax of first-order default theory combines the syntax of both classical logic and logic programming. For semantics, Theorem 3 and

[2] We say that a structure $\mathcal{M}'$ is bigger than another structure $\mathcal{M}$ if for all predicates P, $P^{\mathcal{M}} \subseteq P^{\mathcal{M}'}$.

Theorem 4 show respectively that classical first-order logic and first-order disjunctive logic programming under the stable model semantics can be viewed as special cases of first-order default logic. Moreover, classical logic and logic programming actually form two different dimensions of first-order default logic, where the classical logic dimension is corresponding to the interrelationships among structures in a world view and the logic programming dimension is corresponding to the interrelationships among different world views.

Related Work

A number of alternative semantics have been proposed to address the unsatisfactory semantics for open defaults (Poole 1987; Lifschitz 1990; Baader and Hollunder 1995; Kaminski 1995; 1999). As far as we know, this problem was firstly reported by Poole (1987), who also provided an alternative solution by using Hilbert's ε-symbol. However, Baader and Hollunder (1995) pointed out that, although improved, this semantics is still not that satisfactory as it cannot escape from Skolemization. Instead, Baader and Hollunder considered a weak semantics, in which default rules only apply to known individuals.

The idea of using collections of structures as the solution concept for default logic is not new. This was presented by Guerreiro and Casanova for closed default theories (Guerreiro and Casanova 1990), and later extended by Lifschitz for open default theories restricted to a fixed universe (Lifschitz 1990). Lifschitz fixed a pre-given universe and extended the language by introducing a new constant for each element in the universe. An extension of a default theory is then defined as the set of sentences satisfied by a class of structures, which is a fixed point of an operator similar to Reiter's Γ operator but defined on classes of structures. Kaminski pointed out some problems with Lifschitz's approach, for instance, a default theory is not equivalent to its grounded version under the domain closure assumption (Kaminski 1995). Instead, Kaminski modified Lifschitz's semantics by fixing a pre-given infinite Herbrand universe as well as the interpretations of functions (Kaminski 1995).

In fact, the world view semantics can be regarded as a modification of Lifschitz's and Kaminski's approaches. However, there are three major differences. First, we do not introduce any new universe and we do not need to extend the language by introducing a new constant for each domain element in the new universe. Second, we do not ground on Herbrand universe nor on any other domains. Finally, we do not use the notion $Th(W)$ to denote the set of sentences satisfied by a structure or a class of structures. Although this notion is well defined, one has to be very careful as not every class of structures is corresponding to a set of first-order sentences. We regard the world view semantics as a purely model theoretical approach. Similar to the classical first-order semantics, the notions and notations used in the world view semantics are merely restricted in the language of the default theory.

Conclusion

The main contribution of this paper is the world view semantics for first-order default logic. Roughly speaking, a world view of a default theory is a maximal class of structures sharing the same configuration and satisfying the default theory, where the default part is fixed by the world view itself. A major rationale for using collections of structures as the solution concept is that first-order theories, such as extensions, are not powerful enough to capture all information hidden in a first-order default theory. The world view semantics is essentially different from Reiter's original proposal, even for closed default theories. Nevertheless, they coincide on configuration independent closed default theories. First-order default logic under the world view semantics provides a rich framework for knowledge representation and reasoning as it not only generalizes but also integrates classical first-order logic and first-order answer set programming.

References

Baader, F., and Hollunder, B. 1995. Embedding defaults into terminological knowledge representation formalisms. *J. Autom. Reasoning* 14(1):149–180.

Eiter, T.; Ianni, G.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2008. Combining answer set programming with description logics for the semantic web. *Artificial Intelligence* 172(12-13):1495–1539.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence* 175(1):236–263.

Gelfond, M. 1994. Logic programming and reasoning with incomplete information. *Ann. Math. Artif. Intell.* 12(1-2):89–116.

Guerreiro, R., and Casanova, M. 1990. An alternative semantics for default logic. In *NMR'90*.

Kaminski, M. 1995. A comparative study of open default theories. *Artif. Intell.* 77(2):285–319.

Kaminski, M. 1999. Open default theories over closed domains. *Logic Journal of the IGPL* 7(5):577–589.

Lifschitz, V. 1990. On open defaults. In *Computational Logic: Symposium Proceedings*, 230–237.

Lukasiewicz, T. 2010. A novel combination of answer set programming with description logics for the semantic web. *IEEE Trans. Knowl. Data Eng.* 22(11):1577–1592.

Motik, B., and Rosati, R. 2010. Reconciling description logics and rules. *Journal of the ACM* 57(5).

Motik, B.; Horrocks, I.; Rosati, R.; and Sattler, U. 2006. Can OWL and logic programming live together happily ever after? In *International Semantic Web Conference*, 501–514.

Poole, D. 1987. Variables in hypotheses. In *IJCAI*, 905–908.

Reiter, R. 1980. A logic for default reasoning. *Artif. Intell.* 13(1-2):81–132.

Zhou, Y., and Zhang, Y. 2011. Progression semantics for disjunctive logic programs. In *AAAI*.

Zhou, Y., and Zhang, Y. 2012. RDL: Enhancing description logic with rules. In *AI*, 567–578.



Invited Talks

Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning

Ontology-Based Monitoring of Dynamic Systems*

Franz Baader
Theoretical Computer Science, TU Dresden
Nöthnitzer Str. 46, 01062 Dresden, Germany
baader@tcs.inf.tu-dresden.de

Abstract

Our understanding of the notion "dynamic system" is a rather broad one: such a system has states, which can change over time. Ontologies are used to describe the states of the system, possibly in an incomplete way. Monitoring is then concerned with deciding whether some run of the system or all of its runs satisfy a certain property, which can be expressed by a formula of an appropriate temporal logic. We consider different instances of this broad framework, which can roughly be classified into two cases. In one instance, the system is assumed to be a black box, whose inner working is not known, but whose states can be (partially) observed during a run of the system. In the second instance, one has (partial) knowledge about the inner working of the system, which provides information on which runs of the system are possible.

In this paper, we will review some of our recent work that can be seen as instances of this general framework of ontology-based monitoring of dynamic systems. We will also mention possible extensions towards probabilistic reasoning and the integration of mathematical modeling of dynamical systems.

Introduction

In this paper we use the term "dynamic system" to denote a system that shows dynamic behavior in that it changes its states over time. Here "system" is seen in a broad sense, encompassing both man-made systems and natural systems: it may be a computer system, air traffic observed by radar, a patient in an intensive care unit, or a biological cell. We make no general assumptions about the form of the system's states, except that the states can be described using an appropriate ontology language. These descriptions may be incomplete (partial) in the sense that they do not uniquely determine a single state, but only restrict the possible states to a subset of all states.

In the case of a *black box* system, we have no information on how the system works internally, i.e., we do not know which state is transformed into which other state. In this setting, we assume that the system is observed by certain "sensors" (e.g., heart-rate and blood pressure monitors for a patient), and the results of sensing are stored in a fact base expressed using the given ontology language. Based on the information stored in the fact base, the monitor is supposed to detect certain predefined situations (e.g., heart-rate very high and blood pressure low), which require a reaction (e.g., fetch a doctor or give medication). More precisely, sensor readings are available for different points in time, and thus we obtain a time-stamped sequence of fact bases, each describing (possibly in an incomplete way) the state of the system at the respective time point. The situations to be described may be concerned with not just one state of the system, but a sequence of states (e.g., blood pressure decreasing and heart-rate increasing for a certain time). To describe such situations, the ontology language needs to be combined with an appropriate temporal logic.

In the case of a *white box* system, we have some knowledge about the inner working of the system, i.e., we have a specification of how states of the system are transformed into each other. This description may, however, be nondeterministic in the sense that a given state may not have a uniquely determined successor state, but a set of possible successor states. Given a (possibly incomplete) description of an initial state, the specification then determines a set of possible runs of the system, and we may ask whether some or all of these runs satisfy a certain property, formulated in an appropriate temporal logic.

Of course, we may also have a *combination* of both settings, where a (partial) specification of the system is available, and in addition one can observe the system during one of its runs.

In the following, we will review some of our recent work that can be seen as instances of the general framework of ontology-based monitoring of dynamic systems outlined above. Because of space limitations, we cannot introduce the technical definitions and results in detail. For these and also for detailed descriptions of related work we refer the reader to the cited papers. The ontology languages used in these papers are based on description logics. Again, we do not introduce them in detail, but refer the reader to "The Description Logic Handbook" (Baader et al. 2003) for a comprehensive introduction. We will finish this short paper with mentioning possible extensions of the described approaches towards probabilistic reasoning and the integration of mathematical modeling of dynamical system.

*Partially supported by the Cluster of Excellence "Center for Advancing Electronics Dresden" and the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing."
Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Monitoring Black Box Systems

As mentioned in the introduction, an important ingredient of such a monitoring approach is an appropriate temporal logic, which enables the specification of situations whose definition depends on several of the observed states of the system.

The temporal logic $\mathcal{ALC}$-LTL

Motivated by a situation awareness application (Baader et al. 2009), in which the system was air and marine traffic observed by radar, we have introduced the temporalized description logic $\mathcal{ALC}$-LTL (Baader, Ghilardi, and Lutz 2008; 2012) as such a logic. It combines the ontology language $\mathcal{ALC}$ with the propositional linear temporal logic LTL. Since the temporal structure underlying LTL are the natural numbers, this means that we consider a discrete flow of time. In contrast to propositional LTL, the states are not propositional valuations, but $\mathcal{ALC}$ interpretations, i.e., relational structures with unary predicates (concepts) and binary predicates (roles).

An important design decision for $\mathcal{ALC}$-LTL was, on the one hand, to restrict the application of temporal operators to $\mathcal{ALC}$ axioms, i.e., terminological axioms of the form $C \sqsubseteq D$ and assertional axioms of the form $C(a)$ and $r(a, b)$, where C, D are concepts, r a role, and a, b individuals. From a syntactic point of view, $\mathcal{ALC}$-LTL is obtained from propositional LTL by allowing the use of $\mathcal{ALC}$ axioms in place of propositional variables. For example, the $\mathcal{ALC}$-LTL formula

$$\Diamond\Box(\mathsf{UScitizen} \sqsubseteq \exists \mathsf{insured\_by}.\mathsf{HealthInsurer})$$

says that there is a future time point from which on US citizens will always have health insurance, and the formula

$$\Diamond\big((\exists \mathsf{finding}.\mathsf{Concussion})(\mathsf{BOB}) \wedge \mathsf{Conscious}(\mathsf{BOB})\,\mathsf{U}\,(\exists \mathsf{procedure}.\mathsf{Examination})(\mathsf{BOB})\big)$$

says that, sometime in the future, Bob will have a concussion with no loss of consciousness between the concussion and the examination. However, what cannot be expressed in $\mathcal{ALC}$-LTL is the general concept of a concussion with no loss of consciousness since expressing this concept would require the application of temporal operators within the concept:

$$\exists \mathsf{finding}.\mathsf{Concussion} \sqcap \mathsf{Conscious}\,\mathsf{U}\,\exists \mathsf{procedure}.\mathsf{Examination}.$$

On the other hand, the second important design decision was to allow for rigid concepts and roles, i.e., concepts/roles whose interpretation does not vary over time. For example, the concept Human and the role has_father should probably be rigid since a human being will stay a human being and have the same father over his/her lifetime, whereas Conscious should be a flexible concept (i.e., not rigid) since someone that is conscious at the moment need not always by conscious. Similarly, insured_by should be modeled as a flexible role. Using a logic that cannot enforce rigidity of concepts/roles may result in unintended models, and thus prevent certain useful inferences to be drawn. For example, the concept description $\exists \mathsf{has\_father}.\mathsf{Human} \sqcap \Diamond\,(\forall \mathsf{has\_father}.\neg\mathsf{Human})$ is only unsatisfiable if both has_father and Human are rigid. It should be noted that rigid concepts and roles provide some information about the inner working of the system since they tell us that their interpretation cannot be changed by state transitions.

While rigid concepts and roles increase the expressive power of the formalism, they are problematic from a computational point of view. In fact, in a logic that allows the application of temporal operators within concepts, rigid roles can even cause undecidability (Gabbay et al. 2003). In $\mathcal{ALC}$-LTL, rigid concepts and roles are less harmful, but they still increase the complexity of reasoning: the satisfiability problem in $\mathcal{ALC}$-LTL is 2-ExpTime-complete if both rigid concepts and roles are allowed; NExpTime-complete if only rigid concepts, but no rigid roles are allowed; and ExpTime-complete if neither concepts nor roles may be rigid.

These results are relevant for the monitoring of dynamic systems since it is easy to see that the monitoring problem can indeed be expressed as a satisfiability problem in $\mathcal{ALC}$-LTL: assume that ψ is an $\mathcal{ALC}$-LTL formula expressing a critical situation (e.g., the formula talking about Bob from above); that ϕ_t is an $\mathcal{ALC}$-LTL formula describing the observations made until time point t (e.g., observations made at certain time points by nurses or heart-rate sensors, and compiled into an $\mathcal{ALC}$-LTL formula using the next-operator); and ϕ is a conjunction of terminological axioms expressing (atemporal) background information (e.g., a medical ontology). Then the critical situation can be detected by testing whether the $\mathcal{ALC}$-LTL formula $\Box\phi \wedge \phi_t \wedge \neg\psi$ is satisfiable.

Runtime verification

A potential problem with the monitoring approach sketched above is that the size of the formula ϕ_t increases with every additional time point at which an observation is made, and thus one needs to solve ever larger satisfiability problems. For propositional LTL, this problem has been addressed in the runtime verification community: for a given LTL formula ψ a monitor M_ψ (i.e., a deterministic finite automaton with state output) is constructed whose size depends only on the formula describing the critical situation, and not on the number of time points (Bauer, Leucker, and Schallhart 2011). At each time point, the observations lead to a transition of the monitor, and the output of the state reached through this transition says whether ψ has been violated or not.

Building on results from (Baader, Ghilardi, and Lutz 2008; 2012), we have shown in (Baader, Bauer, and Lippmann 2009) that this approach can be extended from propositional LTL to $\mathcal{ALC}$-LTL. The main advantage of this extension over the propositional case is, on the one hand, that one can employ ontologies defining the important concepts of the application domain (e.g., concepts defined in a medical ontology) to describe critical situations. On the other hand, in contrast to runtime verification for propositional LTL, where for every time point one assumes to have complete information about the values of the propositional variables, the approach developed in (Baader, Bauer, and Lippmann 2009) can also deal with incomplete information. Improving on the results in (Baader, Bauer, and Lippmann

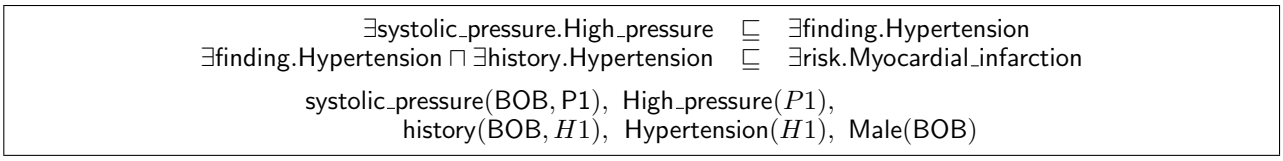


Figure 1: An ontology and a fact base

2009), where some of the monitor constructions were triple-exponential, we show in (Baader and Lippmann 2014) that, for a given $\mathcal{ALC}$-LTL formula ψ, a monitor M_ψ of double-exponential size can be constructed in double-exponential time. In general, this double-exponential blow-up cannot be avoided, even in the propositional case.

Temporalizing ontology-based data access

In (Baader, Borgwardt, and Lippmann 2013), we introduce an extension of $\mathcal{ALC}$-LTL where conjunctive queries using concepts and roles as predicates can be used in place of $\mathcal{ALC}$ axioms. For example, the conjunctive query

$$\exists y.\mathsf{risk}(x, y) \wedge \mathsf{Myocardial\_infarction}(y) \wedge \mathsf{Male}(x)$$

asks for male patients that are at risk to have a heart attack. The obvious difference to $\mathcal{ALC}$ axioms is the use of variables in conjunctive queries. On the one hand, there are the free variables (called answer variables): for these (x in the example) one wants to find named individuals such that replacing the variable by the individual name makes the query true. On the other hand, there are existentially quantified variables, which may refer to unnamed individuals. In our example, the myocardial infarction y need not have an explicit name in the fact base. Answering such queries w.r.t. an ontology is called *ontology-based data access*. Here one assume that one has a fact base, consisting of assertions, and an ontology, consisting of terminological axioms. As an example, assume that the fact base contains the assertions about the patient Bob shown in the lower part of Figure 1, which say that Bob has high blood pressure (obtained from sensor data), and is male and has a history of hypertension (obtained from the patient record). In addition, we have an ontology that says that patients with high blood pressure have hypertension and that patients that currently have hypertension and also have a history of hypertension are at risk for a heart attack, as shown in the upper part of Figure 1. The situation we want to recognize for a given patient x is whether this patient is a male person that is at risk for a heart attack, which is described by the conjunctive query from above. Given the information in the fact base and the axioms in the ontology, we can derive that Bob satisfies this query. Obviously, without the ontology this answer could not be derived.

The complexity of OBDA, i.e., the complexity of checking whether a given tuple of individuals is an answer of a conjunctive query in a fact base w.r.t. an ontology, has been investigated in detail for cases where the ontology is expressed in an appropriate description logic. One can either consider the *combined complexity*, which is measured in the size of the whole input (consisting of the query, the ontology, and the fact base), or the *data complexity*, which is measured in the size of the fact base only (i.e., the query and the ontology are assumed to be of constant size). The underlying assumption is that query and ontology are usually relatively small, whereas the size of the data may be huge. In the database setting (where there is no ontology and closed world assumption is used), answering conjunctive queries is NP-complete w.r.t. combined complexity and in AC^0 w.r.t. data complexity. For OBDA w.r.t. $\mathcal{ALC}$ ontologies, the complexity is considerably higher: ExpTime-complete w.r.t. combined complexity and coNP-complete w.r.t. data complexity (Calvanese, De Giacomo, and Lenzerini 1998; Lutz 2008; Calvanese et al. 2006).

In (Baader, Borgwardt, and Lippmann 2013), we have extended these results to temporal conjunctive queries, which are obtained from LTL by replacing propositional variables with conjunctive queries. The complexity again depends on whether rigid concepts/roles are allowed or not. For the combined complexity, the obtained complexity results are identical to the ones for $\mathcal{ALC}$-LTL, though the upper bounds are considerably harder to show. For data complexity, we obtain the same complexity as for atemporal OBDA (coNP-complete) if no rigid roles are allowed. With rigid roles, we have an ExpTime upper bound (in contrast to 2-ExpTime for combined complexity), but can only show a coNP lower bound.

Monitoring White Box Systems

In this setting, one assumes that one has some knowledge about the inner working of the system, i.e., a (possibly incomplete) specification of how states of the system are transformed into each other. In principle, such a specification could be provided by an appropriate temporal logic. However, there are some problems with this approach. First, such a specification may require a temporalized description logic in which temporal operators can be applied within concepts, which may cause computational problems. Second, in addition to saying what changes during a state transition, one also needs to specify what does not change. This so-called *frame problem* has been investigated in detail in the *reasoning about actions* community. In (Baader et al. 2005; Baader, Lippmann, and Liu 2010) we have introduced action formalisms based on description logics, which inherit Reiter's solution to the frame problem from situation calculus (Reiter 2001).

Based on an action theory defined in such a formalism, one can specify the inner working of a system (e.g., an autonomous robot) using the action programming language Golog (de Giacomo, Lespérance, and Levesque 2000). In this setting, rather than monitoring a single run of the system, we are interested in the *verification problem*: show that certain (temporal) properties are satisfied by any run of the

program. The first attempt to solve the verification problem for action theories based on description logics can be found in (Baader, Liu, and ul Mehdi 2010). However, instead of examining the actual execution sequences of a given Golog program, this approach considers infinite sequences of actions that are accepted by a given Büchi automaton $\mathcal{B}$. If this automaton over-approximates the program, i.e. all possible execution sequences of the program are accepted by $\mathcal{B}$, then any property that holds in all the sequences accepted by $\mathcal{B}$ is also a property that is satisfied by any execution of the program. As logic for specifying properties of infinite sequences of actions, the approach again uses $\mathcal{ALC}$-LTL. In (Baader and Zarrieß 2013), we improve on the results in (Baader, Liu, and ul Mehdi 2010) by directly considering all (finite and infinite) sequences of actions that are runs of a given Golog program.

Future Research

One important topic for future research are extensions towards probabilistic reasoning. Probabilities may come into our framework of ontology-based monitoring of dynamic systems for a variety of reasons: sensors used to populate the fact base may be erroneous with some probability; based on the observed run of the system, one may compute projections into the future, which are again equipped with a probability; the application of an action may have probabilistic outcomes (the action may, e.g., succeed only with some probability).

Another interesting topic is the integration of numerical sensor values. In the black box setting, these values can be represented using description logics with concrete domains (Lutz 2003), which may, however, cause computational problems. In the white box setting, one needs approaches for describing how the numerical values change of time. One possibility for expressing this is to use systems of differential equations, as done in mathematical modeling of dynamical systems (Scheinerman 2012).

References

Baader, F., and Lippmann, M. 2014. Runtime verification using a temporal description logic revisited. LTCS-Report 14-01, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, Germany. See http://lat.inf.tu-dresden.de/research/reports.html.

Baader, F., and Zarrieß, B. 2013. Verification of Golog programs over description logic actions. In *Proc. FroCoS'13*, LNCS 8152, 181–196. Springer-Verlag.

Baader, F.; Bauer, A.; and Lippmann, M. 2009. Runtime verification using a temporal description logic. In *Proc. FroCoS'09*, LNCS 5749, 149–164. Springer-Verlag.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge, UK: Cambridge University Press.

Baader, F.; Lutz, C.; Miličić, M.; Sattler, U.; and Wolter, F. 2005. Integrating description logics and action formalisms: First results. In *Proc. AAAI'05*, 572–577. AAAI Press.

Baader, F.; Bauer, A.; Baumgartner, P.; Cregan, A.; Gabaldon, A.; Ji, K.; Lee, K.; Rajaratnam, D.; and Schwitter, R. 2009. A novel architecture for situation awareness systems. In *Proc. Tableaux'09*, LNCS 5607, 77–92. Springer-Verlag.

Baader, F.; Borgwardt, S.; and Lippmann, M. 2013. Temporalizing ontology-based data access. In *Proc. CADE-24*, LNCS 7898, 330–344. Springer-Verlag.

Baader, F.; Ghilardi, S.; and Lutz, C. 2008. LTL over description logic axioms. In *Proc. KR'08*, 684–694. AAAI Press.

Baader, F.; Ghilardi, S.; and Lutz, C. 2012. LTL over description logic axioms. *ACM Trans. Comput. Log.* 13(3).

Baader, F.; Lippmann, M.; and Liu, H. 2010. Using causal relationships to deal with the ramification problem in action formalisms based on description logics. In *Proc. LPAR'10*, LNCS 6397, 82–96. Springer–Verlag.

Baader, F.; Liu, H.; and ul Mehdi, A. 2010. Verifying properties of infinite sequences of description logic actions. In *Proc. ECAI'10*, 53–58. IOS Press.

Bauer, A.; Leucker, M.; and Schallhart, C. 2011. Runtime verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.* 20(4).

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2006. Data complexity of query answering in description logics. In *Proc. KR'06*, 260–270. AAAI Press.

Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 1998. On the decidability of query containment under constraints. In *Proc. PODS'98*, 149–158. ACM Press.

de Giacomo, G.; Lespérance, Y.; and Levesque, H. J. 2000. Congolog, a concurrent programming language based on the situation calculus. *Artif. Intell.* 121(1-2):109–169.

Gabbay, D.; Kurusz, A.; Wolter, F.; and Zakharyaschev, M. 2003. *Many-dimensional Modal Logics: Theory and Applications*. Elsevier.

Lutz, C. 2003. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*, 265–296. King's College Publications.

Lutz, C. 2008. The complexity of conjunctive query answering in expressive description logics. In *Proc. IJCAR'08*, LNCA 5195, 179–193. Springer-Verlag.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press.

Scheinerman, E. R. 2012. *Invitation to Dynamical Systems*. Dover Publications.

Datalog+/–: Questions and Answers

Georg Gottlob and **Thomas Lukasiewicz** and **Andreas Pieris**

Department of Computer Science, University of Oxford, UK
firstname.lastname@cs.ox.ac.uk

• *What is Datalog$^\pm$ and what are its advantages?*

Datalog$^\pm$ is a rule-based formalism that combines the advantages of logic programming in Datalog with features for expressing ontological knowledge and advanced data modeling constraints. Datalog$^\pm$ provides a uniform framework for query answering and reasoning with incomplete data. It generalizes disparate other KR formalisms such as plain Datalog, description logics (DLs), in particular, DL-Lite and $\mathcal{EL}$, F-Logic Lite, highly relevant fragments of UML class diagrams, data-exchange formalisms, graph query languages such as SPARQL, and so on. Datalog$^\pm$ is a conceptually very simple and clear cut formalism that extends plain Datalog with features such as existential quantifiers, equalities, and the falsum ($\bot$) in rule heads and, at the same time, restricts the rule syntax so as to achieve decidability and, when required, tractability. Its closeness to logic programming allowed us to enrich it by non-monotonic negation under the stratified, well-founded, and stable model semantics.

• *How do Datalog$^\pm$ rules look like?*

A self-explanatory set of Datalog$^\pm$ rules follows:

$$\begin{array}{ll}\sigma_1 & : \ emp(X) \ \rightarrow \ \exists Y \ hasMgr(X,Y), emp(Y), \\ \sigma_2 & : \ hasMgr(X,Y), emp(X) \ \rightarrow \ emp(Y), \\ \sigma_3 & : \ emp(X), emp(Y) \ \rightarrow \ colleagueOf(X,Y), \\ \sigma_4 & : \ emp(X) \ \rightarrow \ \exists Y \ worksIn(X,Y), dept(Y).\end{array}$$

Such rules are also known as *tuple-generating dependencies (tgds)*. All variables that are not existentially quantified are assumed to be universally quantified. We just omit the universal quantifiers in front of such rules. As will be pointed out later, Datalog$^\pm$ rules may also contain equalities and the falsum symbol ($\bot$) in their heads.

• *Which reasoning tasks are considered with Datalog$^\pm$?*

The main reasoning task is query answering under the so-called *certain-answers semantics*. If $Q(\mathbf{X})$ is a conjunctive query (CQ) or a union of conjunctive queries (UCQ) with free variables $\mathbf{X}$, D a database over a domain (universe) Δ whose tuples are interpreted in the usual way as ground facts, and Σ a Datalog$^\pm$ program (a.k.a. Datalog$^\pm$ ontology), then the answer to Q consists of all those tuples $\mathbf{a}$ of Δ-elements such that $D \cup \Sigma \models Q(\mathbf{a})$. For Boolean queries, the answer is, accordingly, *true* or *false*. Other relevant reasoning tasks are instance checking, query containment and consistency (or satisfiability) checking. These latter tasks are, however, easily reduced to Boolean query answering.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

• *Which are the main decidable Datalog$^\pm$ languages?*

There are so far three main paradigms which extend plain Datalog with existential quantifiers in rule heads, and also guarantee the decidability of query answering:

Weakly-acyclic Datalog$^\pm$, based on weakly acyclic tgds introduced in the context of data exchange (Fagin et al. 2005), guarantees the existence of a finite universal model, which in turn implies the decidability of query answering. The rules $\{\sigma_2, \sigma_3, \sigma_4\}$ given above form a weakly-acyclic set of Datalog$^\pm$ rules. Extensions of weak-acyclicity have been studied, e.g., in (Marnette 2009; Grau et al. 2013).

Guarded Datalog$^\pm$ ensures the existence of treelike universal models, which in turn implies the decidability of query answering (Calì, Gottlob, and Kifer 2013). A rule is guarded if it has an atom which contains all the body-variables; e.g., the rules $\{\sigma_1, \sigma_2, \sigma_4\}$. An important subclass of guarded Datalog$^\pm$ is linear Datalog$^\pm$, where rules have only one body-atom, e.g., the rules $\{\sigma_1, \sigma_4\}$. Extensions of guardedness have been studied in (Baget et al. 2011).

Sticky Datalog$^\pm$ guarantees the termination of backward resolution, and thus the decidability of query answering (Calì, Gottlob, and Pieris 2012). The key idea underlying stickiness is that the body-variables which are in a join always are propagated (or "stick") to the inferred atoms; e.g., the set of rules $\{\sigma_1, \sigma_3, \sigma_4\}$. The main goal of stickiness was the definition of a language that allows for joins in rule-bodies, which are not always expressible via guarded rules.

Interestingly, the consolidation of the above paradigms leads to more expressive decidable formalisms. Notable examples are *glut-guardedness* (Krötzsch and Rudolph 2011), obtained by combining weak-acyclicity and guardedness, *weak-stickiness* (Calì, Gottlob, and Pieris 2012), obtained by joining weak-acyclicity and stickiness, and *tameness*, obtained by combining guardedness and stickiness (Gottlob, Manna, and Pieris 2013). Another important language is *shy* Datalog$^\pm$ (Leone et al. 2012).

• *What about equalities and the falsum ($\bot$) in rule heads?*

Weakly-acyclic Datalog$^\pm$ can be safely combined with arbitrary equality rules, a.k.a. *equality-generating dependen-*

Language	Data Complexity	Combined Complexity
Weakly-acyclic Datalog$^{\pm}$	PTIME-complete UB: (Fagin et al. 2005, Cor. 4.3) LB: (Dantsin et al. 2001, Thm. 4.4)	2EXPTIME-complete UB: implicit in (Fagin et al. 2005) LB: (Calì, Gottlob, and Pieris 2012, Thm. 5.1)
Guarded Datalog$^{\pm}$	PTIME-complete UB: (Calì, Gottlob, and Lukasiewicz 2012, Thm. 6) LB: (Calì, Gottlob, and Lukasiewicz 2012, Thm. 6)	2EXPTIME-complete UB: (Calì, Gottlob, and Kifer 2013, Thm.6.1) LB: (Calì, Gottlob, and Kifer 2013, Thm.6.1)
Sticky Datalog$^{\pm}$	in AC_0 UB: (Calì, Gottlob, and Pieris 2012, Thm. 3.5)	EXPTIME-complete UB: (Calì, Gottlob, and Pieris 2012, Thm. 3.3) LB: (Calì, Gottlob, and Pieris 2012, Thm. 3.4)

Table 1: Complexity of query answering; UB and LB stands for upper and lower bound, respectively.

cies (egds), without destroying decidability of query answering (Fagin et al. 2005). However, this is not true for the other languages mentioned above. For those languages, to guarantee decidability, one has to somewhat restrict the interaction between egds and existential rules, for instance, by admitting *non-conflicting* egds only, see (Calì, Gottlob, and Lukasiewicz 2012; Calì, Gottlob, and Pieris 2012). The falsum ($\bot$) in rule heads can be easily treated since such rules (a.k.a. negative constraints) can be conceived as queries. Thus, a language which guarantees the decidability of query answering can be safely extended with this feature.

• *What is the complexity of query answering under the main Datalog$^{\pm}$ languages?*

The data complexity (the query and the set of rules are considered fixed) and combined complexity of query answering under the main Datalog$^{\pm}$ languages (enriched with equality and the falsum) are summarized in Table 1. In each cell, we have indicated where to find the results.

• *What are typical applications of Datalog$^{\pm}$?*

As already said, Datalog$^{\pm}$ is a framework for knowledge representation and reasoning, and for a variety of other applications. Three of its basic applications are (1) reasoning over conceptual models; (2) ontology querying; and (3) querying the semantic web. Let us give some more details for each of the above applications:

1. UML class diagrams (UCDs) are a widely adopted formalism for modeling the intensional structure of a software system. A fundamental property of UCDs is consistency, showing that the system is realizable in practice. Guarded Datalog$^{\pm}$ is powerful enough to capture a core fragment of UCDs, and in particular each diagram which is the result of a reverse engineering process. Thus, consistency of re-engineered UCDs, a key problem in software engineering, can be reduced to reasoning under guarded Datalog$^{\pm}$. This fact allows us to exploit mature and efficient database technology for solving the consistency problem of re-engineered UCDs, which sensibly improves the performance of existing approaches (Gottlob, Orsi, and Pieris 2014).

2. Description Logics (DLs) are a family of knowledge representation formalisms. It is well-known that reasoning under expressive DLs is very hard or even undecidable. For this reason, several lightweight DLs have been proposed in the literature. Two of the most prominent lightweight DLs are DL-Lite$_{\mathcal{R}}$ (Calvanese et al. 2007) and $\mathcal{EL}$ (Baader, Brandt, and Lutz 2005), which provide the logical underpinning for the OWL 2 QL and OWL 2 EL profiles of OWL 2, respectively. Interestingly, guarded Datalog$^{\pm}$ provides a unifying framework for the above DLs without losing tractability of query answering. More precisely, query answering under DL-Lite$_{\mathcal{R}}$ or $\mathcal{EL}$ can be reduced to query answering under guarded Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012). Notice that guarded Datalog$^{\pm}$ is strictly more expressive than the above DLs. Sticky Datalog$^{\pm}$ is also more expressive than DL-Lite$_{\mathcal{R}}$ (Calì, Gottlob, and Pieris 2012).

Another example of a knowledge representation formalism which is captured by a slightly more expressive version of guarded Datalog$^{\pm}$, namely weakly-guarded Datalog$^{\pm}$, is F-Logic Lite (Calì, Gottlob, and Kifer 2013). F-Logic Lite (Calì and Kifer 2006) is an expressive restricted version of F-Logic, a well-known formalism, originally introduced for object-oriented deductive databases, but is now most frequently used for expressing Semantic Web ontologies.

3. The problem of querying RDF data is a central issue for the development of the Semantic Web. The query language SPARQL 1.1 was recently released, which includes entailment regimes for RDFS and OWL vocabularies, and a mechanism to express navigation patterns through regular expressions. However, there are still some useful navigation patterns that cannot be expressed in SPARQL 1.1, and the language lacks of a general mechanism to express recursive queries. With the aim of filling this gap, a query language called TriQ, based on weakly-guarded Datalog$^{\pm}$, has been recently proposed (Arenas, Gottlob, and Pieris 2014). In particular, if we focus on the OWL 2 QL profile of OWL 2, then every SPARQL query enriched with the above features can be naturally translated into a query expressed in TriQ, and also in a lightweight version of TriQ, called TriQ-Lite, which ensures tractability of query evaluation. Notice that TriQ captures EXPTIME; the upper bound has been established in (Arenas, Gottlob, and Pieris 2014), while the lower bound in (Gottlob, Rudolph, and Simkus 2014).

• *Apart from the modeling features mentioned above, what else is allowed in the existing Datalog$^{\pm}$ languages? For example, do you allow for negation in rule bodies?*

The most common semantics for nonmonotonic normal (logic) programs have been extended to Datalog$^{\pm}$ with negation in rule bodies. More precisely, we have studied guarded Datalog$^{\pm}$ with negation under the perfect model semantics (PMS) for the stratified case (Calì, Gottlob, and Lukasiewicz 2012), the well-founded semantics (WFS) (Gottlob et al. 2012a; Hernich et al. 2013), and the stable model semantics

(SMS) (Gottlob et al. 2014b). Actually, it is not difficult to define a negation semantics for Datalog$^{\pm}$, as is this inherited from its standard counterpart in normal logic programs with function symbols. In fact, existentially quantified variables can be replaced by Skolem terms in rule heads. The tricky part, however, is to show that Datalog$^{\pm}$ with negation under the new semantics remains decidable, and pinpoint the exact complexity of query answering.

The WFS comes in two flavors depending on whether the unique name assumption (UNA) is applied to Skolem terms or not. The first approach leads to the "equality-friendly" WFS (Gottlob et al. 2012a), while the second one leads to the standard WFS (Hernich et al. 2013). The SMS (Gelfond and Lifschitz 1988) is another predominating semantics for nonmonotonic normal programs. In (Gottlob et al. 2014b), we have defined and studied the SMS for guarded Datalog$^{\pm}$ under the UNA (note that an equality-friendly SMS with UNA also exists). There are cases in which the SMS leads to better query answers than the WFS. For example, given the fact $FiveStar(ritz)$ and the set of rules

$$FiveStar(X) \rightarrow Hotel(X),$$
$$FiveStar(X), \mathrm{not}\, Pool(X,Y) \rightarrow \exists Z\ Beach(X,Z),$$
$$FiveStar(X), \mathrm{not}\, Beach(X,Y) \rightarrow \exists Z\ Pool(X,Z),$$
$$Beach(X,Y) \rightarrow \exists Z\ SwimOpp(X,Z),$$
$$Pool(X,Y) \rightarrow \exists Z\ SwimOpp(X,Z),$$

only the atoms $FiveStar(ritz)$ and $Hotel(ritz)$ are entailed under the WFS, while the atom $SwimOpp(ritz)$ is additionally entailed under the SMS (as desired).

As several DLs can be embedded into Datalog$^{\pm}$, the decidability results for Datalog$^{\pm}$ with negation can also be applied to define decidable extensions of DLs with nonmonotonic negation. We have done this in particular for DL-Lite$_{\mathcal{R}}$ and DL-Lite$_{\mathcal{R},\sqcap}$ (Calvanese et al. 2007), as well as $\mathcal{ELHI}$ (Baader, Brandt, and Lutz 2005). The above example corresponds to the following set of DL axioms, expressed in an extension of $\mathcal{ELHI}$ by nonmonotonic negation:

$$\begin{aligned} FiveStar &\sqsubseteq \mathrm{Hotel}, \\ FiveStar \sqcap \mathrm{not}\exists Pool &\sqsubseteq \exists Beach, \\ FiveStar \sqcap \mathrm{not}\exists Beach &\sqsubseteq \exists Pool, \\ \exists Beach &\sqsubseteq \exists SwimOpp, \\ \exists Pool &\sqsubseteq \exists SwimOpp, \end{aligned}$$

• *Is Datalog$^{\pm}$ powerful enough for nondeterministic reasoning? In other words, can the existing languages enriched with disjunction in rule heads?*

Weakly-acyclic and guarded Datalog$^{\pm}$ (and their extensions) can be extended with disjunction, without sacrificing the decidability of query answering. In fact, the impact of disjunction on the complexity of query answering under guarded-based Datalog$^{\pm}$ languages has been recently investigated in (Gottlob et al. 2012b; Bourhis, Morak, and Pieris 2013). Unfortunately, the above decidability result does not hold for sticky Datalog$^{\pm}$. A recent (still unpublished) result shows that the combination of stickiness with disjunction leads to undecidability. This result gives rise to the question whether a restricted, yet meaningful fragment of sticky Datalog$^{\pm}$ can be isolated, which can be safely extended with disjunction. This will be the subject of future research.

• *What other developments have been made related to Datalog$^{\pm}$?*

1. Query rewriting has been extensively studied. Several algorithms have been designed with the aim of reducing the problem of query answering under Datalog$^{\pm}$ languages into evaluation of database queries (Gottlob, Orsi, and Pieris 2011; Orsi and Pieris 2011; Gottlob and Schwentick 2012; Gottlob et al. 2014a; Gottlob, Manna, and Pieris 2014a).

2. In (Gottlob et al. 2013), a probabilistic extension of Datalog$^{\pm}$ that is based on Markov logic networks as underlying probabilistic semantics is presented. It may, e.g., be used in data extraction from the Web.

3. In (Lukasiewicz, Martinez, and Simari 2012), a general framework for inconsistency management in Datalog$^{\pm}$ ontologies based on incision functions from belief revision is developed. It may, e.g., be used for handling inconsistencies in the Semantic Web.

4. Towards personalized semantic search (e.g., in social networks), in (Lukasiewicz, Martinez, and Simari 2013) an approach to preference-based query answering in ontologies, combining Datalog$^{\pm}$ with preference management as in relational databases, is presented.

5. Query answering under *finite* models has been also investigated. More precisely, query answering for most of the languages, without egds, mentioned above is *finitely controllable*. This means that the answers to a query remain exactly the same when, instead of considering arbitrary models, we restrict our attention to finite models. Finite controllability of query answering under (weakly-)guarded Datalog$^{\pm}$ was shown in (Bárány, Gottlob, and Otto 2010) (extending the work by (Rosati 2011)), under sticky Datalog$^{\pm}$ in (Gogacz and Marcinkowski 2013), and under tame Datalog$^{\pm}$ in (Gottlob, Manna, and Pieris 2014b).

• *Are there any other modeling features that will be added to Datalog$^{\pm}$?*

We are planning to investigate more expressive equality assertions and transitivity. Let us give some more details:

1. Though existing languages are powerful enough to express equality assertions, as said, this can only be done as long as the non-conflicting condition is satisfied. However, in real-life examples, new intensional knowledge may be inferred from the equality rules, which implies that the non-conflicting condition is violated. Hence, more general conditions, beyond the non-conflicting one, are needed.

2. Transitivity is partially captured by some Datalog$^{\pm}$ languages, e.g., weakly-sticky Datalog$^{\pm}$, as long as the necessary join operations are performed on finitely many values. However, this is not powerful enough for expressing some natural transitivity axioms. We only know that guarded-based Datalog$^{\pm}$ languages cannot be safely extended with transitivity since this leads to undecidability (Gottlob, Pieris, and Tendera 2013). Thus, the enrichment of Datalog$^{\pm}$ with this feature will be the subject of future substantial research.

• *Are there any implemented systems for reasoning over Datalog$^{\pm}$ languages?*

We know of three such systems: (1) Nyaya (Virgilio et al. 2012) is a system able to treat the first-order rewritable fragments of Datalog$^{\pm}$, that is, linear and sticky Datalog$^{\pm}$.

In fact, the given set of rules and query are compiled into an SQL query, which is then evaluated over the extensional database; (2) DLV$^{\exists}$ (Leone et al. 2012) implements a bottom-up evaluation strategy for shy Datalog$^{\pm}$ inside the well-known Answer Set Programming (ASP) system DLV (Leone et al. 2006); and (3) Alaska (König et al. 2012), similarly to Nyaya, is able to treat the first-order rewritable fragments of Datalog$^{\pm}$, and it is based on SQL-rewritings. The above are interesting prototypes which show that a complete system, which is able to effectively answer queries over Datalog$^{\pm}$ ontologies, is realistic. The implementation of such a system will be the subject of future research.

Acknowledgements. Work funded by ERC Grant 246858 (DIADEM), and EPSRC grant EP/J008346/1 (PrOQAW).

References

Arenas, M.; Gottlob, G.; and Pieris, A. 2014. Expressive languages for querying the semantic web. In *Proc. of PODS*. To appear.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, 364–369.

Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; and Salvat, E. 2011. On rules with existential variables: Walking the decidability line. *Artif. Intell.* 175(9-10):1620–1654.

Bárány, V.; Gottlob, G.; and Otto, M. 2010. Querying the guarded fragment. In *Proc. of LICS*, 1–10.

Bourhis, P.; Morak, M.; and Pieris, A. 2013. The impact of disjunction on query answering under guarded-based existential rules. In *Proc. of IJCAI*.

Calì, A., and Kifer, M. 2006. Containment of conjunctive object meta-queries. In *Proc. of VLDB*, 942–952.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.* 48:115–174.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.* 14:57–83.

Calì, A.; Gottlob, G.; and Pieris, A. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.* 193:87–128.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning* 39(3):385–429.

Dantsin, E.; Eiter, T.; Georg, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv.* 33(3):374–425.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: Semantics and query answering. *Theor. Comput. Sci.* 336(1):89–124.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of ICLP/SLP*, 1070–1080.

Gogacz, T., and Marcinkowski, J. 2013. Converging to the chase - a tool for finite controllability. In *Proc. of LICS*, 540–549.

Gottlob, G., and Schwentick, T. 2012. Rewriting ontological queries into small nonrecursive datalog programs. In *Proc. of KR*.

Gottlob, G.; Hernich, A.; Kupke, C.; and Lukasiewicz, T. 2012a. Equality-friendly well-founded semantics and applications to description logics. In *Proc. of AAAI*, 757–764.

Gottlob, G.; Manna, M.; Morak, M.; and Pieris, A. 2012b. On the complexity of ontological reasoning under disjunctive existential rules. In *Proc. MFCS*, 1–18.

Gottlob, G.; Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2013. Query answering under probabilistic uncertainty in Datalog+/– ontologies. *Ann. Math. Artif. Intell.* 69(1):37–72.

Gottlob, G.; Kikot, S.; Kontchakov, R.; Podolskii, V.; Schwentick, T.; and Zakharyaschev, M. 2014a. The price of query rewriting in ontology-based data access. *Artif. Intell.* To appear.

Gottlob, G.; Hernich, A.; Kupke, C.; and Lukasiewicz, T. 2014b. Stable model semantics for guarded existential rules and description logics. In *Proc. of KR*.

Gottlob, G.; Manna, M.; and Pieris, A. 2013. Combining decidability paradigms for existential rules. *TPLP* 13(4-5):877–892.

Gottlob, G.; Manna, M.; and Pieris, A. 2014a. Polynomial combined rewritings for existential rules. In *Proc. of KR*.

Gottlob, G.; Manna, M.; and Pieris, A. 2014b. Querying hybrid fragments of existential rules. Forthcoming.

Gottlob, G.; Orsi, G.; and Pieris, A. 2011. Ontological queries: Rewriting and optimization. In *Proc. of ICDE*, 2–13.

Gottlob, G.; Orsi, G.; and Pieris, A. 2014. Database techniques for consistency checking of re-engineered UML class diagrams. Under review.

Gottlob, G.; Pieris, A.; and Tendera, L. 2013. Querying the guarded fragment with transitivity. In *Proc. of ICALP*, 287–298.

Gottlob, G.; Rudolph, S.; and Simkus, M. 2014. Expressiveness of guarded existential rule languages. In *Proc. of PODS*. To appear.

Grau, B. C.; Horrocks, I.; Krötzsch, M.; Kupke, C.; Magka, D.; Motik, B.; and Wang, Z. 2013. Acyclicity notions for existential rules and their application to query answering in ontologies. *J. Artif. Intell. Res.* 47:741–808.

Hernich, A.; Kupke, C.; Lukasiewicz, T.; and Gottlob, G. 2013. Well-founded semantics for extended Datalog and ontological reasoning. In *Proc. of PODS*, 225–236.

König, M.; Leclère, M.; Mugnier, M.-L.; and Thomazo, M. 2012. A sound and complete backward chaining algorithm for existential rules. In *Proc. of RR*, 122–138.

Krötzsch, M., and Rudolph, S. 2011. Extending decidable existential rules by joining acyclicity and guardedness. In *Proc. of IJCAI*.

Leone, N.; Pfeifer, G.; Faber, W.; Eiter, T.; Gottlob, G.; Perri, S.; and Scarcello, F. 2006. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.* 7(3):499–562.

Leone, N.; Manna, M.; Terracina, G.; and Veltri, P. 2012. Efficiently computable Datalog$^{\exists}$ programs. In *Proc. of KR*.

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2012. Inconsistency handling in Datalog+/– ontologies. In *Proc. of ECAI*.

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2013. Preference-based query answering in Datalog+/– ontologies. In *Proc. of IJCAI*, 1017–1023.

Marnette, B. 2009. Generalized schema-mappings: from termination to tractability. In *Proc. of PODS*, 13–22.

Orsi, G., and Pieris, A. 2011. Optimizing query answering under ontological constraints. *PVLDB* 4(11):1004–1015.

Rosati, R. 2011. On the finite controllability of conjunctive query answering in databases under open-world assumption. *J. Comput. Syst. Sci.* 77(3):572–594.

Virgilio, R. D.; Orsi, G.; Tanca, L.; and Torlone, R. 2012. Nyaya: A system supporting the uniform management of large sets of semantic data. In *Proc. of ICDE*, 1309–1312.

Index

Abductive Reasoning Approach to the Belief-Bias Effect, An, 653
Abstraction Technique for the Verification of Multi, Agent Systems against ATL Specifications, An, 428
Action Theories over Generalized Databases with Equality Constraints (Extended Abstract), 650
Aggregative Deontic Detachment for Normative Reasoning, 646
Amir, Eyal, 638
Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory, 101
Answer Set Programming and Logic Programming, 2
Answering Instance Queries Relaxed by Concept Similarity, 248
Appropriate Causal Models and Stability of Causation, 198
Area Chairs and Program Committee, xii
Argumentation, 42
ASP Encodings of Acyclicity Properties, 634
Asuncion, Vernon, 2
Atif, Jamal, 614
Aucher, Guillaume, 579
Automated Reasoning and Computation, 82
Awad, Edmond, 594
Axiomatizing Rationality, 178
Axioms .2 and .4 as Interaction Axioms, 579

Baader, Franz, xix, 678
Baaz, Matthias, xvii
Bagheri Hariri, Babak, 458
Baier, Jorge A., 388
Baral, Chitta, xv
Bartholomew, Michael, 583
Bauters, Kim, 602
Beek, Wouter, 587
Belardinelli, Francesco, 398
Belief Change and Base Dependence, 151
Belief Change and Semiorders, 160
Belief Change Operations: A Short History of Nearly Everything, Told in Dynamic Logic of Propositional Assignments, 141
Belief Revision and Nonmonotonicity, 121
Belief Revision in the Propositional Closure of a Qualitative Algebra, 622
Belle, Vaishak, 438, 590
Benferhat, Salem, 538
Bhatt, Mehul, 508
Bienvenu, Meghyn, 218
Bjorndahl, Adam, 178
Bloch, Isabelle, 614
Bochman, Alexander, 188
Booth, Richard, 594
Borgwardt, Stefan, 228
Botoeva, Elena, 238
Brown, Gavin, 661
Buchman, David, 548

Cabalar, Pedro, 598
Calvanese, Diego, 218, 458
Canonical Logic Programs Are Succinctly Incomparable with Propositional Formulas, 665
Causality and Rationality, 178
Certain Answers as Objects and Knowledge, 328
Cerutti, Federico, 42
Characteristics of Multiple Viewpoints in Abstract Argumentation, 72
Charalambidis, Angelos, 12
Choi, Arthur, 558
Cohn, Tony, xix
Computing Narratives of Cognitive User Experience for Building Design Analysis: KR for Industry Scale Computer-Aided Architecture Design, 508
Concept Dissimilarity with Triangle Inequality, 614
Conference Committee, xi
Constructive Negation in Extensional Higher-Order Logic Programming, 12
Coste-Marquis, Sylvie, 52

Darwiche, Adnan, 111, 558
Datalog+/–: Questions and Answers, 682
David Poole's Specificity Revised, 168
Decidable Gödel Description Logics without the Finitely-Valued Model Property, 228
Decidable Reasoning in a Fragment of the Epistemic Situation Calculus, 468
De Clercq, Sofie, 602
De Cock, Martine, 602
De Giacomo, Giuseppe, xv
de Haan, Ronald, 82
Delgrande, James P., 151, 606
Denecker, Marc, 22
Description Logics, 218
Deutsch, Alin, 458
Diagnostic Problem Solving via Planning with Ontic and Epistemic Goals, 388
Diakidoy, Irene-Anna, 610
Diéguez, Martín, 598
Dietz, Emmanuelle-Anna, 653
Dimopoulos, Yannis, 669
Distel, Felix, 228, 614
Doutre, Sylvie, 62
Duckham, Matt, 618
Dufour-Lussier, Valmi, 622
Dunne, Paul E., 72
Dvorák, Wolfgang, 72

Dynamic Causal Calculus, 188
Dynamic Logic Framework for Abstract Argumentation, A, 62
Dyrkolbotn, Sjur Kristoffer, 626

Ecke, Andreas, 248
Egalitarian Belief Merging, On, 121
Eiter, Thomas, xv, xvii
Everaere, Patricia, 121
Ewin, Christopher, 448
Exact Learning of Lightweight Description Logic Ontologies, 298

Finite Model Reasoning in Horn Description Logics, 288
Fionda, Valeria, 630
First-Order Default Logic Revisited, 673
First-Order Semantics for Golog and ConGolog under a Second-Order Induction Axiom for Situations, A, 478
Forgetting in Action, 498
Formalization of Programs in First-Order Logic with a Discrete Linear Order, A, 338
Full Papers, 1

Gebser, Martin, 634
Generalized Multi-Context Systems, 368
Giacomin, Massimiliano, 42
Girju, Roxana, 638
Girlea, Codruta, 638
Gottlob, Georg, xix, 258, 268, 682
Grossi, Davide, 131
Gutierrez, Claudio, 630
Gutierrez, Julian, 408
Gutiérrez-Basulto, Víctor, 278

Halpern, Joseph Y., 178, 198
Harrenstein, Paul, 408
Harrison, Amelia, 32
Hermann, Alice, 622
Hernich, André, 258
Herzig, Andreas, 62, 141
Heuristic Guided Optimization for Propositional Planning, 669
Hölldobler, Steffen, 653
How to Argue for Anything: Enforcing Arbitrary Sets of Labellings Using AFs, 626
How to Progress Beliefs in Continuous Domains, 438
Huang, Xiaowei, 418

Ianovski, Egor, 208
Ibañez-García, Yazmín, 288
Interval Methods for Judgment Aggregation in Argumentation, 594
Invited Talks, xix, 677

Janhunen, Tomi, 634
Jung, Jean Christoph, 278
Justified Beliefs by Justified Arguments, 131

Kakas, Antonis, 610
Kazemi, Seyed Mehran, 548
Kersting, Kristian, 548
Kisa, Doga, 558
Knowledge Maps of Web Graphs, 630
Knowledge Representation and Reasoning, 328
Konev, Boris, 298, 318
Konieczny, Sébastien, 52, 121
Kontchakov, Roman, 238
Krötzsch, Markus, 308
Kupke, Clemens, 258

Lakemeyer, Gerhard, 468, 590
Le Ber, Florence, 622
Lee, Joohyung, 583
Levesque, Hector J., 438, 468, 498
Li, Sanjiang, 618
Libkin, Leonid, 328
Lieber, Jean, 622
Lifschitz, Vladimir, 32
Lightweight Description Logics and Branching Time: A Troublesome Marriage, 278
Lin, Fangzhen, 338, 478
Linear Programs for Measuring Inconsistency in Probabilistic Logics, 568
Linsbichler, Thomas, 72
Liu, Weiming, 618
Logic Programs with Ordered Disjunction: First-Order Semantics and Expressiveness, 2
Lomuscio, Alessio, xxii, 428, 488
Long, Zhiguo, 618
Ludwig, Michel, 318
Lukasiewicz, Thomas, 258, 682
Lutz, Carsten, 288, 298

Mailly, Jean-Guy, 52
Manna, Marco, 268
Marquis, Pierre, 52, 121
Marrella, Andrea, 518
McIlraith, Sheila, xix, 388
Mecella, Massimo, 518
Meert, Wannes, 111
Michael, Loizos, 348, 610
Michaliszyn, Jakub, 428, 488
Miller, Rob, 610
Minimal Change in AGM Revision for Non-Classical Logics, 657
Model Checking Unbounded Artifact-Centric Systems, 488
Mombourquette, Brent, 388
Montali, Marco, 458
Morgenstern, Leora, 642
Moss, Lawrence S., xxii

Natarajan, Sriraam, 548
Nested Regular Path Queries in Description Logics, 218
Nominal Schemas in Description Logics: Complexities Clarified, 308
Nowé, Ann, 602

OBDDs for CNFs of Bounded Treewidth, On, 92
Ong, Luke, 208
Ontology-Based Monitoring of Dynamic Systems, 678
Ortiz, Magdalena, 218
Ovchinnikova, Ekaterina, xxii
Oveisi, Mehrdad, 151
Ozaki, Ana, 298

Pagnucco, Maurice, 498
Parameterized Complexity of Reasoning Problems Beyond NP, The, 82

Parent, Xavier, 646
Pass, Rafael, 178
Patrizi, Fabio, 650
Pearce, Adrian R., 448
Pelletier, Francis Jeffry, 151
Peñaloza, Rafael, 228, 248
Peppas, Pavlos, 160
Pereira, Luís Moniz, 653
Perrussel, Laurent, 62
Pieris, Andreas, 268, 682
Pirró, Giuseppe, 630
Planning, Strategies, and Diagnosis, 388
Polynomial Combined Rewritings for Existential Rules, 268
Poole, David, 548
Popowich, Fred, 151
Potyka, Nico, 568
Practical Uniform Interpolation and Forgetting for *ALC* TBoxes with Applications to Logical Difference, 318
Predicting Performance of OWL Reasoners: Locally or Globally?, 661
Preface, xv
Probabilistic Sentential Decision Diagrams, 558
Progression of Knowledge in Multiagent Systems, On the, 590
Psychology-Inspired Approach to Automated Narrative Text Comprehension, A, 610

Qualitative Spatial Representation and Reasoning in Angry Birds: The Extended Rectangle Algebra, 378
Query Inseparability for Description Logic Knowledge Bases, 238

Rahwan, Iyad, 594
Rajaratnam, David, 498
Razgon, Igor, 92
Reasoning about Actions and Processes, 438
Reasoning about Equilibria in Game-Like Concurrent Systems, 408
Reasoning with Uncertain Inputs in Possibilistic Networks, 538
Redundant Topological Constraints (Extended Abstract), On, 618
Relational Logistic Regression, 548
Renz, Jochen, 378
Reports from the Field, 508
Representing and Reasoning about Time Travel Narratives: Foundational Concepts, 642
Revision of Argumentation Systems: Minimal Change of Arguments Statuses, On the, 52
Ribeiro, Marcio M., 657
Rintanen, Jussi, 634
Rondogiannis, Panos, 12
Rosati, Riccardo, xxii
Rough Set Semantics for Identity on the Web, 587
Rudolph, Sebastian, 308
Ryzhikov, Vladislav, 238

Sardina, Sebastian, 518
Satisfiability of Alternating-Time Temporal Epistemic Logic through Tableaux, 398
Sattler, Uli, 661
Sazonau, Viachaslau, 661
SCC Recursive Meta-Algorithm for Computing Preferred Labellings in Abstract Argumentation, An, 42
Schlobach, Stefan, 587
Schneider, Thomas, 278, 288
Schockaert, Steven, 602
Schüller, Peter, 358
Schultz, Carl, 508
Semantics of Gringo and Infinitary Propositional Formulas, The, 32
Shen, Yuping, 665
Short Papers, 578
Sideris, Andreas, 669
Simkus, Mantas, 218
Simultaneous Learning and Prediction, 348
Skolemization for Weighted First-Order Model Counting, 111
SmartPM: An Adaptive Process Management System through Situation Calculus, IndiGolog, and Classical Planning, 518
Sponsors, xxiv
Stable Model Semantics for Guarded Existential Rules and Description Logics, 258
Stable Models of Multi-Valued Formulas: Partial versus Total Functions, 583
State-Boundedness in Data-Aware Dynamic Systems, 458
Stolzenburg, Frieder, 168
Strass, Hannes, 101
Strong Equivalence of Non-Monotonic Temporal Theories, 598
Szeider, Stefan, 82

Tabia, Karim, 538
Tackling Winograd Schemas by Formalizing Relevance Theory in Knowledge Graphs, 358
Tasharrofi, Shahab, 368
Temporal Logic of Strategic Knowledge, A, 418
Ternovska, Eugenia, 368
[There Exists] ∃GuaranteeNash for Boolean Games Is NEXP-Hard, 208
Thielscher, Michael, 498
Thimm, Matthias, 528
Thosar, Madhura, 508
Toward a Knowledge Level Analysis of Forgetting, 606
Tracking Beliefs and Intentions in the Werewolf Game, 638
Transforming Situation Calculus Action Theories for Optimised Reasoning, 448
Turhan, Anni-Yasmin, 248
Tutorials, xxii
Tweety: A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and Knowledge Representation, 528

Uncertainty, 538
Using Answer Set Programming for Solving Boolean Games, 602

Vallati, Mauro, 42
Van den Broeck, Guy, 111, 558
Van Der Hoek, Wiebe, 131
van Der Meyden, Ron, 418
van der Torre, Leon, 646
van Harmelen, Frank, 587
Vassos, Stavros, 448, 650
Veith, Helmut, xvii

Vennekens, Joost, 22
Vienna Summer of Logic 2014, xvii

Wallner, Johannes Peter, 101
Wassermann, Renata, 657
Well-Founded Semantics Is the Principle of Inductive Definition, Revisited, The, 22
Williams, Mary-Anne, 160
Wirth, Claus-Peter, 168
Wolter, Frank, 238, 298
Woltran, Stefan, 72
Wooldridge, Michael, 408

Yang, Fangkai, 32

Zakharyaschev, Michael, 238
Zanella, Marina, 42
Zhang, Heng, 2
Zhang, Peng, 378
Zhang, Yan, 2
Zhao, Xishun, 665
Zhou, Yi, 673