Action, Time and Space in Description Logics

Dissertation

zur Erlangung des akademischen Grades
Doktor rerum naturalium (Dr. rer. nat.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dipl.-Inform. Maja Miličić
geboren am 9. Mai 1978 in Belgrad

Gutachter:

Prof. Dr.-Ing. Franz Baader
Technische Universität Dresden

Prof. Dr. rer. nat. Bernhard Nebel
Albert-Ludwigs-Universität Freiburg

Prof. Dr. rer. nat. habil. Michael Thielscher
Technische Universität Dresden

Tag der Verteidigung:
19. Juni 2008

Dresden, im September 2008

Acknowledgements

I am deeply indebted to Carsten Lutz for his friendly supervision and for many excellent ideas he provided. Without his guidance this thesis would not exist. I am greatly thankful to Franz Baader for providing a relaxed working environment and for giving me a financial support for the whole period of my PhD studies. I would also like to thank Frank Wolter and Ulrike Sattler for many great ideas and their support.

A big thanks goes to my colleagues Hongkai Liu, Conrad Drescher and Barış Sertkaya for reading parts of the preliminary version of this work and giving me their invaluable comments. Moreover, I would like to thank all members of the Chair for Automata Theory for their friendship and support. Most of all I want to thank my dear officemate Barış for sharing with me all ups and downs during the last four years, as well as our wonderful secretary Kerstin Achtruth for her caring attitude and warm support. Many thanks to my dear former flatmates Jelka and Yvonne for their encouragement, friendship, and care. Thanks to all my friends from Belgrade and Dresden, scattered around the world, for staying close despite the physical distance.

I want to thank my dear parents and brothers for their love, encouragement and continuous and unconditional support. I know that there is no one more proud and happy about this dissertation than my parents, and I dedicate this work to them. Finally, I want to thank my beloved Sebastian for suffering together through my PhD time, for proof-reading parts of this work and improving the language, and for his love which has kept me going.

This PhD thesis has been written with the financial support of the DFG graduate programme 334 "Specification of discrete processes and systems of processes by operational models and logics", the DFG project TH 541/14 and the EU project "Thinking ONtologiES" (TONES).

Contents

Chapter 1

Introduction

In the 1960s and 1970s it was recognized that knowledge representation (KR) and reasoning is the main part of any intelligent system. Nowadays, research on KR and reasoning plays an essential role in the area of artificial intelligence (AI). Various KR formalisms have been developed for different purposes. The central place in this thesis will have *description logics*, a prominent class of logic-based KR formalisms suited for the representation of and *decidable* reasoning about static knowledge; and *action formalisms*, formalisms based on classical logic dedicated to the modelling of and reasoning about dynamic systems. The largest part of this thesis is dedicated to integrating these two fields, by designing action formalisms underlined by description logics, with the main goal of ensuring *decidability* of standard reasoning problems about actions. A smaller part of the thesis is related to extensions of description logics with concrete datatypes, most importantly with those allowing to refer to the notions of space and time. We start by introducing description logics in Section 1.1 and action and planning formalisms in Section 1.2. Novel contributions on combining description logics with action formalisms are described in Section 1.3, while in Section 1.4 novel results on description logics equipped with concrete datatypes are presented.

1.1 Description Logics

Description logics (DLs) are a family of logic-based knowledge representation formalisms designed to represent and reason about conceptual knowledge in a structured and semantically well-understood way [BCM$^+$03]. DLs evolved from early knowledge representation formalisms such as semantic networks [Qui68] and frames [Min75]. These used simple graphs and structured objects, respectively, to represent knowledge and many algorithms were developed to manipulate these data structures. However, the major drawback of these early KR systems was a lack of formal semantics. This gave rise to the creation of so-called *concept languages* which allow to define concepts as formulae over fragments of first-order logic. The name changed to "concept description logics", "terminological logics", and finally to *description logics*. The earliest DL system was KL-ONE [BS85], introduced by Ron Brachman in 1985. Although it was later shown that reasoning about KL-ONE knowledge bases is undecidable [SS89], this early DL system laid the foundations of syntax, semantics, and reasoning problems for modern description logics. Most modern DLs can be viewed as well-behaved fragments of first-order logic that sacrifice some of the expressivity of first-order logic in order to regain decidability of reasoning.

The basic notions in DLs are *concepts* (unary predicates) and *roles* (binary relations). A specific DL is mainly characterized by the set of *constructors* it provides to build more complex concepts and roles out of atomic ones. The smallest propositionally closed DL, $\mathcal{ALC}$ [SSS91], is a notational variant of the multi-modal logic K_ω [Sch91] and provides the following constructors: negation ($\neg$), conjunction ($\sqcap$), disjunction ($\sqcup$), and existential ($\exists$) and universal ($\forall$) restriction. As a simple example of an $\mathcal{ALC}$-concept, we might describe the concept of a mother as follows:

$$\mathsf{Female} \sqcap \mathsf{Human} \sqcap \exists \mathsf{has\_child}.\mathsf{Human}.$$

More expressive DLs are obtained from $\mathcal{ALC}$ by adding *inverse roles*, *nominals* (constants), *number restrictions*, *transitive roles*, *role hierarchies*, etc. Among DLs that are not propositionally closed, an important role has the description logic $\mathcal{EL}$ which provides conjunction, existential restriction, and the top ($\top$) constructor.

Description logic *knowledge bases* usually consist of a terminological part, called a *TBox*, and an assertional part, called an *ABox*. TBoxes come in several flavours, most importantly *acyclic TBoxes* and *general TBoxes*. Acyclic TBoxes consist of concept definitions of the form $A \doteq C$, defining the concept name A as a complex concept C. TBoxes are called acyclic if and only if the definition of no concept refers directly or indirectly to itself. General TBoxes allow for *general concept inclusions (GCIs)* of the form $C \sqsubseteq D$, where C and D are (possibly) complex concepts, stating that C implies D. While acyclic TBoxes just introduce (possibly exponentially more succinct) representations for complex concepts, general TBoxes are able to capture complex domain constraints. ABoxes are used to give incomplete descriptions of the world, giving *assertions* about individuals, either by assigning individuals to concepts or by establishing binary relations between individuals via roles. We illustrate description logic knowledge bases with one example.

Example 1.1.1. *Let* Human *and* Female *be atomic concepts and let* $\mathsf{has\_child}$ *be a role. Then* $\mathsf{has\_child}^-$ *denotes the inverse of* $\mathsf{has\_child}$*, i.e. the has-parent relation. The TBox* $\mathcal{T}$

$$\begin{aligned}
\mathsf{Parent} &\doteq \mathsf{Human} \sqcap \exists \mathsf{has_child}.\mathsf{Human} \\
\mathsf{Mother} &\doteq \mathsf{Female} \sqcap \mathsf{Parent} \\
\mathsf{Father} &\doteq \neg\mathsf{Female} \sqcap \mathsf{Parent} \\
\mathsf{Human} &\sqsubseteq (= 1\ \mathsf{has_child}^-.\mathsf{Mother}) \sqcap (= 1\ \mathsf{has_child}^-.\mathsf{Father}) \\
\mathsf{Human} &\sqsubseteq \forall \mathsf{has_child}.\mathsf{Human}
\end{aligned}$$

defines concepts of parent, mother and father and requires that every human has exactly one mother and one father, as well as that humans have only human children. Moreover, the ABox

$$\mathcal{A} := \{(\mathsf{Female} \sqcap \mathsf{Human})(\mathsf{anna})\ ,\ \mathsf{has\_child}(\mathsf{anna}, \mathsf{mia})\ ,\ \mathsf{Human}(\mathsf{mia})\}$$

states that anna *is a female human with a human child* mia*.*

Description logics have a *model-theoretic* semantics, hence the meaning of DL concepts is given by means of *interpretations*. An interpretation is a tuple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a *domain*, and $\cdot^\mathcal{I}$ is an *interpretation function*. The interpretation function maps atomic concepts to subsets of the domain, roles to binary relations over the domain, and individuals to domain elements. The semantics of complex concepts is defined inductively based on

the constructors used in the concept description. Interpretations can be viewed as *complete* world descriptions. An interpretation $\mathcal{I}$ is said to be a *model* of a TBox $\mathcal{T}$ if and only if the left-hand side and the right-hand side of every concept definition in $\mathcal{T}$ is interpreted identically, and the extension of C is contained in the extension of D for every GCI $C \sqsubseteq D$ in $\mathcal{T}$. Similarly, $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$ if and only if for every assertion $C(a)$ in $\mathcal{A}$, the interpretation of a is contained in the interpretation of C, and for each $r(a, b)$ in $\mathcal{A}$, the tuple of interpretations of a and b is contained in the interpretation of r. The semantics of ABoxes is *open world semantics*, which means that the absence of information about an individual is not interpreted as negative information but only indicates a lack of knowledge.

Reasoning services on knowledge bases enable us to deduce implicit consequences from the explicitly represented knowledge. The standard reasoning problems considered in DLs are *satisfiability* and *subsumption* of concepts. A concept C is said to be satisfiable w.r.t. a TBox $\mathcal{T}$ if and only if there exists a model of $\mathcal{T}$ in which C is interpreted as a non-empty set. C is subsumed by a concept D w.r.t. $\mathcal{T}$ if and only if C is more specific than D in the sense that, w.r.t. every model of $\mathcal{T}$, the interpretation of C is a subset of that of D. To *classify* a TBox $\mathcal{T}$ means to compute all subsumption relationships between concepts occurring in $\mathcal{T}$. Further standard reasoning problems are *knowledge base consistency* and the *instance problem*[1]. An ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$ if and only if they have a common model. An individual a is an instance of a concept C w.r.t. $\mathcal{A}$ and $\mathcal{T}$ if and only if in all joint models of $\mathcal{A}$ and $\mathcal{T}$, the interpretation of a is contained in the interpretation of C. In Example 1.1.1, the ABox $\mathcal{A}$ is consistent w.r.t. the TBox $\mathcal{T}$, the concept $\mathsf{Mother} \sqcap \mathsf{Father}$ is not satisfiable w.r.t. $\mathcal{T}$, Mother is subsumed by Female w.r.t. $\mathcal{T}$, and the individual anna is an instance of Mother w.r.t. $\mathcal{A}$ and $\mathcal{T}$.

Unsurprisingly, the computational complexity of the above reasoning tasks increases with the expressivity of the underlying DL. $\mathcal{ALC}$-concept satisfiability is a PSPACE-complete problem [SSS91], and various decidable extensions of $\mathcal{ALC}$ are PSPACE-, EXPTIME-, and NEXPTIME-complete. While on one side, a substantial part of research on DLs was dedicated to pushing decidability borders by designing algorithms for very *expressive* DLs such as $\mathcal{SHOIQ}$ [HS05], there has also been a significant amount of research on so-called *lightweight* DLs such as $\mathcal{EL}$ and its carefully designed extensions [BBL05] and *DL-lite* [CGL$^+$05], in which reasoning is tractable.

Due to their formally defined semantics, good computational properties, and expressivity far beyond propositional logic, DLs are increasingly used as *ontology* languages. Ontologies provide a common vocabulary for a domain of interest together with descriptions of the meanings of terms built from the vocabulary and relationships between them. In the context of description logics, ontology is a synonym for a knowledge base. DL ontologies are successfully used in the *life sciences* and play a key role in the *Semantic Web*. We shall now briefly describe the role ontologies play in these two domains.

Ontologies in life sciences

Life sciences are natural sciences dedicated to the study of living organisms, including medicine, biology, biomedicine, and biochemistry. Given the complexity of the human anatomy, physiological processes in single cells, or the distribution and causes of diseases studied in epidemiology, the need for representing and classifying knowledge in a systematic way arose

[1] Throughout the present thesis, we will refer to the instance problem as the *consequence* problem: $C(a)$ is a consequence of a knowledge base iff a is an instance of C w.r.t. the same knowledge base.

naturally. Nowadays, DLs found a very successful application as ontology languages in the life sciences [Spa01, BC07, SAW+07, BSL06, GZB06]. An interesting characteristic of ontologies in this domain is that they usually comprise a very large number of concepts, but are defined over relatively inexpressive DLs such as $\mathcal{EL}$. This holds for one of the most prominent life science ontologies, the general-purpose medical ontology SNOMED [Spa01], that is accepted as a standard of the International Health Terminology Standards Development Organization. Further examples are the National Cancer Institute's ontology [SdCH+06], the medical ontology GALEN [RH97], and the GeneOntology [The00].

The Semantic Web

The Semantic Web [BLHL01, BHS03] is an evolving extension of the World Wide Web in which web content can be expressed in a format readable by software agents, thus permitting them to find, share, and integrate information more easily. This is achieved by a semantic markup, while the meaning of terms used in the markup is defined in ontologies. In 2004, the World Wide Web Consortium (W3C) recommended the DL-based formalism OWL[2] [HPSvH03] as a standard ontology web language. OWL is underlined by the expressive description logic $\mathcal{SHOIQ}$, but unlike the usual DL syntax it uses an RDF[3]/XML syntax. The fact that a crucial component of the Semantic Web is supposed to be based on DLs is a strong motivation for the ongoing DL research and opens a potentially large field for future applications.

DL reasoning services provide reasoning support for ontology engineering, such as their creation, usage, maintenance, integration, etc. Besides the standard ones, further DL reasoning services such as *answering queries* or *finding explanations* for inconsistencies were inspired by needs of ontology engineers and users. These services are provided by state-of-the-art DL reasoners such as FaCT++ [TH06], Pellet [SPG+07], RacerPro [HM01b], and CEL [BLS06]. The reasoners FaCT++, RacerPro and Pellet support expressive description logics and their cores are successful implementations of the tableau algorithm for $\mathcal{SHOIQ}$, using special techniques to avoid non-determinism and ensure termination. The CEL reasoner implements tractable reasoning in lightweight description logics such as $\mathcal{EL}$. Many of these reasoners are integrated with modern DL ontology editors such as Protégé [HTR06].

The main research goals in the area of description logics may be seen as, on the one hand, providing a toolbox of decidable logics such that, for a given application, one can select a DL with adequate expressivity; and on the other hand, introducing and investigating new reasoning services needed in various applications. For a comprehensive introduction to description logics and its applications we refer the reader to [BCM+03].

1.2 Action and Planning Formalisms

In AI, the notion of an *action* plays a central role in two different research communities, one dealing with planning and the other with reasoning about action. In both communities, an action denotes an entity whose execution (by some agent) causes changes to the world.

The research on reasoning about action has mainly been driven by the so-called *frame problem* since it was described by McCarthy and Hayes in [MH69]. The frame problem is the

[2] Web Ontology Language, see also http://www.w3.org/TR/owl-ref/
[3] Resource Description Framework, see also http://www.w3.org/RDF

challenge of representing the effects of action without explicitly having to represent a large number of intuitively obvious non-effects. Intuitively, this means that actions affect only a tiny part of the world. By the law of inertia, everything that is not affected by an action stays the same. Another problem well-known in the reasoning-about-action community is the *ramification problem*. This problem is concerned with indirect consequences of an action. It might be phrased as the challenge to represent the implicit effects of an action or how to control secondary and tertiary action effects which appear due to general *state (domain) constraints*.

In the present section, we briefly introduce well-known action formalisms such as the Situation Calculus and the Fluent Calculus, as well as their solutions to the frame and the ramification problem. In the area of reasoning about actions, the most prominent reasoning problems are *projection*, the problem of determining what holds after executing an action, and *executability*, the problem of whether an action is applicable. Although *planning*, the problem of checking whether there exists a sequence of actions that achieves a certain goal, can be formulated in action formalisms, it has been independently, and more exhaustively studied within the area of planning formalisms. We will also present the most important planning formalisms and planners.

Action Formalisms

The oldest action formalism, the *Situation Calculus (SitCalc)*, is a language specifically designed for representing dynamically changing worlds. It was first introduced by McCarthy in 1963 [McC63] and elaborated in [MH69]. Since then, many different versions of SitCalc were proposed, while the standard version is considered to be the one introduced by Reiter in [Rei01]. Formally, the Situation Calculus is a three-sorted second-order theory, with the three sorts being actions, situations, and objects. Situations are sequences of actions, built from the constant symbol $\mathbf{s_0}$ representing the initial situation, and the binary function symbol do: $do(a, s)$ denotes the situation obtained by executing action a in the situation s. Properties whose truth depends on the situation are represented by predicates that have one additional parameter denoting the situation. For example, "likes(b, c, s)" would be read as "b likes c in situation s". Such situation dependent predicates (and functions) are called *fluents*. In its most common form, the Situation Calculus is restricted to *deterministic* effects of actions. *Action pre-conditions* are expressed by means of the predicate $Poss$. Axioms of the form $Poss(A(\mathbf{x}), s) \equiv \Pi_A(\mathbf{x}, s)$ express that it is possible to execute action $A(\mathbf{x})$ in situation s iff $\Pi_A(\mathbf{x}, s)$ holds. It is required that $\Pi_A(\mathbf{x}, s)$ is a first order formula *uniform* in s, i.e., that it does not depend on any other situation term than s. In early versions of the Situation Calculus [LPR88, Rei91], action effects are specified in terms of *positive* and *negative effect* axioms. For each relational fluent F, first order formulae $\gamma_F^+(\mathbf{x}, a, s)$ and $\gamma_F^-(\mathbf{x}, a, s)$ are assumed to characterize all the conditions under which action a causes F to become true (or false) in the successor situation. The frame problem is solved by compiling positive and negative effects into fluent-centred *successor state axioms*:

$$F(\mathbf{x}, do(a, s)) \equiv (F(\mathbf{x}, s) \vee \gamma_F^+(\mathbf{x}, a, s)) \wedge \neg\gamma_F^-(\mathbf{x}, a, s)$$

These axioms ensure that the value of the fluent F differs in the situations s and $do(a, s)$ iff it is enforced by action effects. In the standard version of SitCalc [Rei01], no particular syntactic form of successor state axioms is assumed; they are simply of the form $F(\mathbf{x}, do(a, s)) \equiv \Phi_F(\mathbf{x}, a, s)$ where Φ_F is uniform in s. Standard domain specification in the Situation Calculus

is comprised out of the following axioms: (i) (second order) foundational axioms for situations; (ii) the initial database (initial situation axioms describing $\mathbf{s_0}$); (iii) unique name axioms for actions; (iv) action pre-condition axioms; and (v) successor state axioms.

Projection (the problem of finding consequences of action application) and action executability are solved in the Situation Calculus mainly by a method called *regression*. Regression is the iterative process of expressing situations of the form $do(a, s)$ in terms of a formula containing the action a and the situation s, but not the situation $do(a, s)$, which leads to obtaining an equivalent formula that contains only the initial situation $\mathbf{s_0}$. Alternatively to regression, consequences can be proved by its dual, *progression*, which denotes the computation of updates of the initial database [LR97, VL07]. However, progression of arbitrary formulae is not always first-order definable in SitCalc.

The second important action formalism, the *Fluent Calculus* [Thi05a, Thi05b], has its roots in the logic programming formalism of [HS90]. It was developed to solve not only the *representational*, but also the *inferential frame problem*. In contrast to the former problem which is concerned only with specifying all non-effects of actions, the latter is the problem of actually inferring these non-effects.

The Fluent Calculus extends SitCalc with the predefined sort for *states* and its subsort *fluent*. SitCalc fluents are reified which means that fluents are represented as logic terms, rather than predicates. Hence second-order quantifiers are not needed in order to quantify over fluents, which ensures that, unlike in SitCalc, progression in the Fluent Calculus is first-order definable. A fluent is, for example, the term $on(box, table)$, where on is a binary function. States represent facts that hold in a situation and they are obtained by concatenating fluents with a binary function symbol $\circ$. This function is assumed to be associative, commutative, and idempotent, ensuring that states can be viewed as sets of fluents. Central to the axiomatization technique of the Fluent Calculus is a function $State(s)$ which relates a situation s to the state of the world in that situation. It can be expressed that the box is on the table in the situation s with the formula $Holds(on(box, table), State(s))$, where the macro $Holds(f, z)$ is defined as $Holds(f, z) \equiv (\exists z')z = f \circ z'$. The inferential frame problem is solved by action-centred *state update axioms* that are designed in such a way that only one such axiom suffices to infer the entire change caused by the action in question, contrasting SitCalc successor state axioms which update single fluents. The state update axioms assert that the situation after the execution of an action is identical to the one before but for the fluents changed by the action. This is achieved by expressions of the form:

$$State(do(A(\mathbf{x}), s)) = State(s) - \vartheta^- + \vartheta^+$$

where finite states ϑ^+ and ϑ^- are the positive and negative effects of the action $A(\mathbf{x})$ in situation s if a condition $\Delta(s)$ is satisfied. The operation "+" simply denotes $\circ$-concatenation, while $z - f = z'$ holds if and only if $(z' = z \vee z' \circ f = z') \wedge \neg Holds(f, z')$, for a fluent f and can easily be recursively extended to subtraction of finite states.

Domain specifications in the Fluent Calculus are comprised of similar axioms as in SitCalc, with the difference that foundational axioms define properties of the $\circ$ operation and states (instead of situations); and that successor state axioms are replaced with state update axioms. It is shown in [Thi99, ST06] that standard domain axomatizations in the Situation Calculus and the Fluent Calculus are mutually translatable in case of deterministic actions.

In the versions of the Situation Calculus and Fluent Calculus that additionally provide *state constraints*, i.e., axioms describing general constraints on the domain, solutions to the *ramification problem* had to be worked out as well. In [LR94, Lif90, Lin95, McI00] it has been

suggested to use the circumscription formalism in order to address this problem. In particular, the approaches of [Lin95, McI00] rely on an acyclic *causality* relation between fluents and employ prioritized circumscription [Lif94] – in this special case, the result of circumscription is first-order definable. An advantage of the causality relation mentioned above is that it can be used to eliminate non-intuitive results of action application. A similar approach was followed in [Thi97], where causal relationships between fluents define conditions under which indirect effects are expected. Successor states are obtained by first computing direct effects of an action and then applying a chain of casual relationships in order to compute indirect effects as well. We would like to emphasize that the ramification problem is considered to be a very hard problem in the reasoning-about-action community, and to our best knowledge there is no generally accepted solution to this problem. There exist even some extreme opinions [Sch90, Rei01], arguing that the ramification problem need not be taken seriously, and that an axiomatizer should take ramifications into account when writing effect axioms. However, in domains with complex chains of causal influences that often occur in engineering systems (see [McI00]), it is far from easy to pre-compute all ramifications.

GOLOG and *FLUX*, languages for designing and programming intelligent agents, are underpinned by the Situation Calculus and the Fluent Calculus, respectively. GOLOG (alGOl in LOGic) [LRL+97] is a high-level language implemented in Prolog that combines control structures from classical programming with reasoning about actions. At its core, GOLOG uses the SitCalc action theory to define the meaning of its primitive actions and the initial situation. Based on these primitives, complex actions can be formed using constructs from imperative programming languages such as: sequence, non-deterministic choice of programs or parameters, non-deterministic iteration, if-then-else, and while. FLUX (FLUent eXecutor) [Thi05a] is a constraint logic programming method based on the Fluent Calculus. Unlike GOLOG, where the description of the current state is implicitly given via the initial state and the actions that have been performed to reach that state, FLUX has an explicit state representation as its fundamental concept. While GOLOG reasoning is based on regression, FLUX uses the progression principle and scales better to the control of agents over extended periods. Both FLUX and GOLOG perform sound but incomplete reasoning w.r.t. underlying theories.

Planning formalisms

An important prerequisite of intelligent systems as envisioned by AI research is the ability to reach complex goals without explicitly being told all the necessary steps. Consequently, planning has been seen as an important research area in AI since its very beginning. Informally, *planning* is the problem of finding a sequence of actions (called *plan*) that leads to a world state where a certain goal is satisfied. The associated decision problem is the plan existence problem (PLANEX), i.e. given a description of the initial situation, the available actions and the goal, the problem of whether such a sequence of actions exists.

The Situation Calculus has occasionally served as a theoretical device for the study of planning in AI, but except for a short period in the early 1970s, it has never been taken seriously for the implementations of planning systems. A part of the problem was that no solution to the frame problem existed for a while, and this led to the creation of the *STRIPS* formalism dedicated exclusively to planning in 1971. Although solutions to the frame problem in the Situation Calculus and the Fluent Calculus have been worked out in the meantime, and, in principle, GOLOG and FLUX support planning, the creation of STRIPS and first

efficient planners supporting STRIPS led the research on planning to take a new course, independent of the one taken by the reasoning-about-action community.

The STRIPS language [FN71] describes operators (parameterized actions which contain variables in place of individuals) in terms of their pre-conditions and effects. The effects of an operator are given by so-called add- and delete-lists, which are sets of positive literals (fluents). For example, moving an object x from the top of the object y to the table is described via the the operator $moveToTable(x, y)$:

$$\begin{array}{rcl} \text{pre} & : & clear(x) \wedge on(x, y) \\ \text{add} & : & clear(y), onTable(x) \\ \text{delete} & : & on(x, y) \end{array}$$

STRIPS suffered for some time from the lack of a formal semantics for its operators. The first attempt to provide a logic-based semantics for STRIPS was by Lifschitz [Lif86]. Lifschitz proposed a notion of *sound* operators, for which it is possible to introduce transition semantics in a well-defined manner. In the approach of Pednault [Ped89], STRIPS operators are mappings from first-order structures to first-order structures, where the addition and deletion of tuples are applied to the relations of the structure.

In basic (propositional) STRIPS, the initial state and goal state are completely described as conjunctions of positive literals under CWA. In this case, the state obtained by applying a (grounded) operator is simply obtained by subtracting fluents contained in the delete list from the current state, and adding fluents contained in the add-list. It was shown in [Byl94] that the plan existence problem is PSPACE-complete in basic STRIPS. Intuitively, the reason is that one may construct planning problems that can be solved only with exponentially long sequences of actions: think of a binary counter C of length n. The initial state describes that all bits of C are 0, the goal state requires that all bits of C are set to 1, and only a single operator *inc* is available that increments C by 1. The complexity of more expressive variants of STRIPS is extensively investigated in [ENS95].

ADL [Ped89, Ped94] relaxed some of the restrictions of the STRIPS language by supporting equality, conditional effects, and typing; as well as allowing disjunction, negation, and quantifiers in operator pre-conditions. The Problem Domain Description Language (PDDL) [FL03] was introduced as a computer-parsable, standardized syntax for representing STRIPS, ADL, and other languages. PDDL is now the standard language for planning competitions. For the study of comparative expressiveness of propositional planning formalisms ranging from basic STRIPS to propositional ADL, based on the notion of *compilability* see [Neb00]. Note that the generally accepted semantics of ADL is based on Lifschitz' approach [Lif86], and that world states are simply sets of propositional literals, while execution of operators which may result in the states not representable as sets of literals is forbidden.

There are several well-known planners supporting basic STRIPS. Most of these search for plans of a bounded size and use different heuristics for an efficient state-space search. Note that the complexity of STRIPS planning drops to NP if only plans of polynomial length in the size of the planning problem are considered. The best known STRIPS planners are GRAPHPLAN [BF97] based on processing the planning graph by using a backward search to extract a plan, SATplan [KS96] using planning-as-satisfiability approach [KS92] and BLACKBOX [KS98] combining ideas from the previous two. Recently, planners supporting ADL and other expressive fragments of PDDL beyond STRIPS have been developed. Among them is the FF (FAST-FORWARD) planner [HN01] which implements a successful state-search ap-

proach, combining forward and local search, and using a simplified planning graph heuristic for pruning the branches of the search tree that cannot provide a plan.

There has been significantly less work on so-called *conformant* planning, i.e. planning in case only incomplete world descriptions are available (open world assumption) and without any sensing capabilities during plan execution. The word "conformant" describes the requirement that the same plan leads to goal satisfaction, independently of the initial situation. Conformant planning can be transformed into a search problem in the space of *belief states*, where belief states represent sets of possible world states. It has been shown that the conformant plan existence problem is ExpSpace-complete in the propositional case [HJ99, Rin04], thus being much harder than basic propositional STRIPS planning. It follows from [Tur02] that, in the propositional case, finding conformant plans of polynomial size is Σ_2^P-complete. Among conformant planners, well-known representatives are: Conformant GraphPlan [SW98], SAT-based conformant planners [FG00, CGT03, PG06], the model checking approach [CR00], and Conformant FF [HB06]. Besides adapting search heuristics from their classic counterparts to the conformant case, conformant planners face the new challenge of compactly representing belief states whose size may be exponential in the size of the input.

Recently, there have been attempts to bridge the gap between action and planning formalisms. It has been shown in [CL06, CELN07] how to define the semantics of ADL in terms of progression in the Situation Calculus. Moreover, it has been suggested to integrate GOLOG with an ADL planner such that planning problems can be solved by a modern planner during the execution of a GOLOG program. Such attempts might enable a convergence of the reasoning-about-action and planning communities which have been pursuing different paths ever since the introduction of STRIPS.

1.3 Integrating DLs with Action Formalisms

1.3.1 Motivation

Action theories such as the Situation Calculus (SitCalc) and the Fluent Calculus [Rei01, Thi05b] are formulated in first- or higher-order logic and do not permit decidable reasoning. For reasoning about actions in practical applications, such theories are thus not directly suited. There are two obvious remedies to this problem. The first one is to accept undecidability and replace reasoning by programming – this route is taken by the inventors of action-oriented programming languages, such as GOLOG [LRL+97] and FLUX [Thi05a]. The second approach is to identify fragments of action theories such as SitCalc that are sufficiently expressive to be useful in applications, but nevertheless admit decidable reasoning. For example, a simple fragment of that kind is obtained by allowing only propositional logic for describing the state of the world and pre- and post-conditions of actions, as in propositional STRIPS [Byl94]. This implies that one has to choose between a very expressive but undecidable action formalism, and a decidable but only propositional one.

Motivated by this dichotomy among available action formalisms, we propose to investigate the middle ground and try to find a compromise between the two extremes. Investigating a fragment of SitCalc based on description logics appears to be a very natural choice, since DLs are known to have the properties we are looking for: they provide expressivity much beyond the propositional case while still maintaining decidability of reasoning. By "based on description logics", we mean that ABox assertions are used to describe world states and pre-

and post-conditions of actions, while TBoxes serve as state constraints. From the perspective of the DL community, our work can be seen as "importing" reasoning problems about actions into the DL world and relating them to standard DL reasoning problems. Our results, which are going to be described in more detail in the following sections, show that important reasoning problems, such as projection, executability, and the plan existence problem, are decidable in DL-based action formalisms.

Although our work has mainly a theoretical motivation, applications of DLs as ontology languages support our belief that description logic actions can find useful practical applications in the future. In medical applications, ABoxes can be used to describe states of the world, such as patient data in the medical domain. In this context, actions can naturally be used to represent the diagnostic and therapeutic steps taken during the treatment of a patient. Executability and projection could help to determine whether a certain therapy is applicable to the patient or not, as well as whether it has contra-indications.

Research related to the Semantic Web is not limited to the development of a semantic markup language for static web pages. In addition, there is ongoing research on annotating web services which allow their users to perform various actions, like buying a book or opening a bank account. The OWL-S initiative [MBM+07] uses OWL to develop an ontology of services, covering different aspects of web services. However, a faithful representation of the dynamic behaviour of processes (the changes they cause to the world) is beyond the scope of a static ontology language like OWL. In order to formalize the dynamic aspects of web services and to describe their composition, different approaches are used: translating OWL-S process models into the Situation Calculus/GOLOG [MSZ01, MS02] or Petri Nets [NM02]. Still, none of the existing approaches enables sound and complete reasoning about services, neither was any of them widely accepted and successful.

Without pretending that the formalism introduced in the present work can capture the full functionality of web services, we argue that it lays a theoretical foundation for future work on decidable reasoning about semantic web services. Our composite DL actions can be viewed as sequences of simple world altering web services. In this context, projection, executability, and planning are very important reasoning tasks as they support, e.g., web service discovery which is needed for automatic service execution.

1.3.2 Contributions and Results

We propose action formalisms in which the state of the world and the pre- and post-conditions of actions can be described using DL concepts. The proposal is generic in the sense that our framework can be instantiated with many standard DLs. We use a STRIPS-like syntax for actions, where atomic actions are defined as triples consisting of *pre-conditions*, *occlusions*, and *post-conditions*. Pre-conditions are simply represented by DL ABoxes, and they describe conditions under which the action is applicable. Conditional post-conditions are specified by expressions of the form ψ/φ, where ψ and φ are ABox assertions, describing that if ψ holds before executing the action then φ should hold after. Composite actions are defined to be sequences of atomic actions.

Naturally, DL ABoxes are used to give (incomplete) world state descriptions and TBoxes to capture state constraints. We develop two action formalisms, $\mathfrak{A}_1$ and $\mathfrak{A}_2$. Formalism $\mathfrak{A}_1$ supports only atomic concepts in action post-conditions and acyclic TBoxes while formalism $\mathfrak{A}_2$ extends $\mathfrak{A}_1$ by supporting complex concepts in post-conditions and handling ramifications introduced by general TBoxes. Occlusions play a different role in $\mathfrak{A}_1$ and $\mathfrak{A}_2$: they are

only a source of limited non-determinism in the former, but have an important function in fine-tuning the ramifications in the latter. Composite actions are defined to be sequences of atomic actions. When defining the semantics of actions, we assume that states of the world correspond to interpretations. Thus, the semantics of actions is defined by means of a transition relation on interpretations, similar to Pednault's STRIPS semantics [Ped94]. We have investigated the reasoning problems projection and executability in $\mathfrak{A}_1$ and $\mathfrak{A}_2$, as well as planning in $\mathfrak{A}_1$. The remainder of this subsection gives an account of the obtained novel results.

Atomic Concepts in Action Post-Conditions

It is ensured that changes induced by $\mathfrak{A}_1$-actions may occur only at an atomic level by forbidding complex concepts in the right-hand sides of post-conditions ψ/φ. Hence, φ has to be of the form $A(a)$, $\neg A(a)$, $r(a,b)$, or $\neg r(a,b)$, where A is an atomic primitive concept. This means that A is not defined in the acyclic TBox $\mathcal{T}$ that describes the background knowledge. The semantics of $\mathfrak{A}_1$-actions is such that the changes induced by action execution are minimized. More precisely, an action α transforms an interpretation $\mathcal{I}$ into $\mathcal{I}'$ if and only if the following conditions hold: $\mathcal{I}$ and $\mathcal{I}'$ (i) are models of $\mathcal{T}$, (ii) satisfy the conditional post-conditions of α, (iii) have the same domain and interpret all individuals in the same way, and (iv) coincide on the interpretation of all primitive concepts and roles in those parts of the domain not affected by the post-conditions and not specified by occlusions.

We illustrate the components of $\mathfrak{A}_1$ with an example from the medical domain.

Example 1.3.1. *Let the background knowledge about the types of drugs and their chemical ingredients be represented by the following TBox $\mathcal{T}$:*

$$\begin{aligned} \mathsf{Drug} &\doteq \mathsf{Antibiotic} \sqcup \mathsf{ChemotherapyDrug} \sqcup \ldots \\ \mathsf{PBA} &\doteq \mathsf{Antibiotic} \sqcap \exists \mathsf{basedOn.Penicillin} \\ \mathsf{MBD} &\doteq \mathsf{ChemotherapyDrug} \sqcap \exists \mathsf{basedOn.Methotrexate} \end{aligned}$$

Moreover, let the initial world state be described with the ABox $\mathcal{A} = \mathcal{A}_{\mathsf{drugs}} \cup \mathcal{A}_{\mathsf{patient}}$. The first component of $\mathcal{A}$, the ABox $\mathcal{A}_{\mathsf{drugs}}$, contains the knowledge about specific drugs and their compatibility:

$$\mathcal{A}_{\mathsf{drugs}} := \{\mathsf{PBA}(\mathsf{amoxil}), \mathsf{MBD}(\mathsf{trexall}), \neg\mathsf{compatible}(\mathsf{amoxil}, \mathsf{trexall}), \ldots\}$$

The second ABox $\mathcal{A}_{\mathsf{patient}}$ contains patient data, including patient's symptoms and the list of admissible and requested drugs for the chosen patient. Initially, all those drugs the patient is not allergic to may be described as admissible.

$$\begin{aligned} \mathcal{A}_{\mathsf{patient}} := \{&\mathsf{Patient}(\mathsf{p}), \exists\mathsf{hasSymptom.HighBloodPressure}(\mathsf{p}), \ldots, \\ &\mathsf{admissible}(\mathsf{p}, \mathsf{amoxil}), \mathsf{admissible}(\mathsf{p}, \mathsf{trexall}), \ldots, \\ &\mathsf{requested}(\mathsf{p}, \mathsf{amoxil}), \mathsf{requested}(\mathsf{p}, \mathsf{trexall}), \ldots\} \end{aligned}$$

The parameterized action $\mathsf{administer}(p, x)$ *describes the administration of a drug x to a patient p. Conditional post-conditions of the form* $\neg\mathsf{compatible}(x, y)/\neg\mathsf{admissible}(p, y)$ *ensure that, if a drug y is not compatible with the administered drug x, then y is not admissible anymore*

for the patient p.

$$\begin{aligned}\text{pre}: \quad & \{\mathsf{Patient}(p), \mathsf{Drug}(x), \mathsf{requested}(p,x), \mathsf{admissible}(p,x)\}\\ \text{post}: \quad & \{\neg\mathsf{requested}(p,x), \mathsf{administered}(p,x),\\ & \neg\mathsf{compatible}(x,\mathsf{amoxil})/\neg\mathsf{admissible}(p,\mathsf{amoxil}),\\ & \neg\mathsf{compatible}(x,\mathsf{trexall})/\neg\mathsf{admissible}(p,\mathsf{trexall}),\dots\}\end{aligned}$$

If the sequence of actions $\mathsf{administer}(\mathsf{p},\mathsf{amoxil});\mathsf{administer}(\mathsf{p},\mathsf{trexall})$ *is executable, this means that the corresponding treatment is applicable to the patient* p. *If it is executable, projection may be used to check whether the assertion* $(\exists\mathsf{administered}.\exists\mathsf{basedOn}.\mathsf{Penicillin})(\mathsf{p})$ *is a consequence of applying this sequence of actions.*

We show that instantiations of $\mathfrak{A}_1$ with expressive DLs can be viewed as fragments of the Situation Calculus and thus inherit SitCalc's well-established solution of the frame problem [Rei91, Rei01].

Concerning reasoning, we focus on the basic tasks of executability and projection, which are mutually polynomially reducible in our framework. For a large number of standard description logics $\mathcal{L}$ that contain the basic DL $\mathcal{ALC}$, we establish a close connection between projection in $\mathfrak{A}_1$ instantiated with $\mathcal{L}$, and standard DL reasoning tasks in a moderate extension of $\mathcal{L}$. More precisely, by using a method similar to regression in SitCalc, we show that projection in $\mathcal{L}$ can be polynomially reduced to ABox consequence in $\mathcal{LO}$, the extension of $\mathcal{L}$ with nominals, i.e., singleton concepts. This reduction allows us to prove decidability and upper complexity bounds for executability and projection in the action formalism $\mathfrak{A}_1$ instantiated with DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ (i.e., the extension of $\mathcal{ALC}$ with number restrictions, inverse roles and nominals). Since standard reasoning in $\mathcal{ALCQIO}$ is supported by standard DL reasoners such as FaCT++, RacerPro, or Pellet, reasoning about action can be delegated to these.

Thus, we give a positive answer to the question whether there exists a decidable compromise between propositional and FO action theories. To pinpoint the exact computational complexity of our formalism, we show that, in a certain sense, the reduction mentioned above can be reversed: standard DL reasoning in $\mathcal{LO}$ can polynomially be reduced to projection in $\mathcal{L}$. In particular, this means that the additional computational complexity (sometimes) caused by the introduction of nominals cannot be avoided. By combining the two reductions, we obtain tight complexity bounds for projection in DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$, where the complexity ranges from PSpace-complete to co-NExpTime-complete.

Further complexity results for the lightweight DL $\mathcal{EL}$ shed even more light on the nature of the projection problem. Although standard reasoning problems such as subsumption and ABox consequence are tractable in $\mathcal{EL}$ (compared to PSpace-complete in $\mathcal{ALC}$), projection in $\mathcal{EL}$ is PSpace-hard and thus as hard as projection in $\mathcal{ALC}$. If we disallow TBoxes, the complexity of projection in $\mathcal{EL}$ drops to co-NP-complete, and is thus still not tractable. The additional complexity comes from the fact that, by exploiting negative assertions in action post-conditions and existential restrictions in the ABox, disjunction can be simulated even by means of simple unconditional post-conditions.

We also consider several natural extensions of the formalism $\mathfrak{A}_1$ and point out some of the problems encountered with these extensions. While the semantics of actions can easily be adapted to account for ramifications introduced by role hierarchies, transitive roles introduce a serious non-determinism that cannot be trivially resolved. Similarly, allowing cyclic concept definitions in the background TBox is shown to cause semantic problems.

Complex Concepts in Action Post-Conditions

The restrictions adopted in the action formalism $\mathfrak{A}_1$, such as allowing only acyclic TBoxes and atomic primitive concepts in action post-conditions, defuse the TBox ramification problem. This problem inevitably occurs, however, in case of the more powerful general TBoxes. Since general TBoxes do not allow for a distinction between defined and primitive concept names, admitting general TBoxes as state constraints means admitting complex concepts in action post-conditions as well. Hence, we have to reconsider the frame problem, as the straightforward action semantics suited for atomic changes in $\mathfrak{A}_1$ cannot be applied in case of complex concepts in post-conditions.

We discuss how attempts to *automatically* solve the frame problem and general TBox ramification problem, e.g., by adopting a Winslett-style PMA semantics [Win88], lead to semantic and computational problems: counter-intuitive results and undecidability of reasoning are the consequence of adopting such a semantics. Since there appears to be no general automated solution to the general TBox ramification problem other than resorting to very inexpressive DLs [GLPR06], we propose to leave it to the designer of an action description to fine-tune the ramifications of the action. In SitCalc and the Fluent Calculus, the ramification problem is addressed similarly: the designer of an action description can control the ramifications of the action by specifying causal relationships between predicates [Lin95, Thi97]. While causality appears to be a satisfactory approach for addressing the ramification problem induced by Boolean state constraints, it seems not powerful enough for attacking the ramifications introduced by general TBoxes which usually involve complex quantification patterns. We therefore advocate a different approach: when describing an action, the user can specify the predicates that may change through the execution of the action, as well as those that may not change. In the action formalism $\mathfrak{A}_2$ that supports complex post-conditions, in order to allow an adequate fine-tuning of ramifications, we admit complex statements in action occlusions about the change of predicates. These can be of the form "the concept name A can change from positive to negative only at the individual a, and from negative to positive only where the complex concept C was satisfied before the action was executed".

Example 1.3.2. *Assume that the TBox $\mathcal{T}$ from Example 1.3.1 is extended with the following GCI:*

$$\exists \mathsf{administered}.\exists \mathsf{basedOn}.\mathsf{Penicillin} \sqsubseteq \forall \mathsf{admissible}.\neg(\exists \mathsf{basedOn}.\mathsf{Methotrexate})$$

It describes that, if a penicillin-based drug is administered, methotrexate-based drugs are not admissible. If an action post-condition contains the assertion $\mathsf{administered}(\mathsf{p}, \mathsf{amoxil})$*, then,* $\forall \mathsf{admissible}.\neg(\exists \mathsf{basedOn}.\mathsf{Methotrexate})$ *should also hold for* p *as an indirect effect of the action. Although this complex concept can be satisfied in many different ways, intuitively, we expect that only the interpretation of* admissible *might change, while* basedOn *and* Methotrexate *remain the same. We can tune the action ramifications by stating that, after action execution, only* admissible *may change from positive to negative, and only for the pairs whose first component is* p*, and the second component satisfies* $\exists \mathsf{basedOn}.\mathsf{Methotrexate}$.

We show that, for many standard extensions of $\mathcal{ALC}$, the reasoning problems *executability* and *projection* in $\mathfrak{A}_2$ are decidable. We also pinpoint the exact computational complexity of these reasoning problems. As a rule of thumb, our results show that reasoning in $\mathfrak{A}_2$ instantiated with a description logic $\mathcal{L}$ is of the same complexity as ABox consequence w.r.t. general TBoxes in $\mathcal{L}$ extended with nominals. For fine-tuning the ramifications, consistency

of actions is an important property. We introduce two notions of consistency (weak and strong) and show that weak consistency is of the same complexity as deciding projection while strong consistency is undecidable even when the action formalism is instantiated with the basic DL $\mathcal{ALC}$.

Planning

We complete our research of action reasoning problems in the context of description logics by investigating planning in the action formalism $\mathfrak{A}_1$. A *planning task* is described as a tuple consisting of DL ABoxes $\mathcal{A}$ and Γ, (incompletely) describing the initial and goal world state, respectively; an acyclic TBox $\mathcal{T}$ defining concepts used in $\mathcal{A}$ and Γ; a set of operators Op (parameterized $\mathfrak{A}_1$-actions), and a set Ind of individuals that can be used to instantiate operators. Intuitively, the *plan existence problem* is the following: is there a plan (a sequence of actions, i.e., grounded operators) which "transforms" $\mathcal{A}$ into a state where Γ is satisfied?.

Using standard planning terminology, we may say that we investigate "DL conformant planning" as world states are *incompletely* described with ABoxes and we require that plans are *uniform*, i.e., that the same plan achieves the goal, independently of the model of the initial ABox (world state). We investigate the computational complexity of the plan existence problem for description logics between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ as well as in the lightweight $\mathcal{EL}$. Our results show that planning is harder with operators that contain conditional post-conditions than with those that have only unconditional ones.

If we allow only for actions with unconditional post-conditions, we show that, in these logics the plan existence problem is decidable and of the same computational complexity as projection. By using a compact representation of the possible states obtained by action application, we show that the search space is exponential in the size of the planning task. Hence the plan existence problem can be solved by a PSpace (graph reachability) algorithm with a "projection oracle". As a consequence, we obtain that, in logics between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$, the complexity of the plan existence problem ranges from PSpace-complete to co-NExpTime-complete.

If conditional post-conditions are allowed, it is not possible to represent states of the search space by means of accumulated action post-conditions. The reason is that different models of the initial ABox $\mathcal{A}$ may cause different post-conditions to be triggered. Using combinatorial methods, we show that the plan existence problem in this case is in 2-ExpSpace. Since the only known lower complexity bound is inherited from propositional logic (ExpSpace-hard), we leave the exact computational complexity of "conditional" PLANEX as an open problem.

Finally, we show that hardness results for propositional planning carry over to planning in the lightweight description logic $\mathcal{EL}$.

1.3.3 Related Work

Early Results

In the early papers from the 1990s on representing actions in description logics [DL96, HKNP92, AF98], actions and plans were represented statically as description logic concepts. The emphasis was on classification-based reasoning, i.e., computing subsumption relations between actions and plans.

Motivated by the need to represent plan-like knowledge in the domain of telephone switching software, Litman and Devanbu [DL96] developed the Clasp system designed to represent

and reason about large collections of plan descriptions. In their system, the notions of subsumption and classification are extended from concepts to plans and actions. Actions are represented as concepts written in the early DL system CLASSIC by means of (essentially propositional) pre- and post-conditions. Plans are built out of actions by using constructors corresponding to regular expressions which allow to express disjunction, sequences, and iterations of actions. Subsumption algorithms for plans integrate the work on finite automata with concept subsumption algorithms. CLASP also provides a limited support of reasoning about action, based on the propositional STRIPS semantics.

In the RAT system [HKNP92] by Heinsohn et al., built on top of the early DL system $\mathcal{KRIS}$, a similar approach was followed, while increasing the expressiveness of action pre- and post-conditions and limiting plans to linear sequences of actions. Plans and actions are treated as concepts, and their instances are treated as DL individuals. Moreover, like in [DL96], several notions of subsumption between actions and plans are considered. One of them, called abstraction-subsumption means that an action α_1 subsumes an action α_2 if and only if the same holds for the pre- and post-conditions of α_1 and α_2. The meaning of actions is specified via syntax-based updates of ABoxes. RAT provides the service of simulated execution of plans, thus checking the feasibility of plans, i.e., checking whether all the intermediate states obtained by the plan execution are consistent.

In [AF98], Artale and Franconi proposed a temporal description logic whose concepts can be used to describe actions and plans. The proposed temporal DL is a hybrid between Allen's interval algebra and DLs such as $\mathcal{ALC}$ with feature agreements. Action concepts describe what is true before and after executing the action, while plans are constructed by using temporal Allen relations to relate actions and world states. Moreover, decision procedures for concept subsumption in the proposed temporal DLs can be used to decide subsumption between actions and plans. However, no satisfactory solution to the frame problem is presented.

In contrast to the previously described papers, the focus of De Giacomo et al. in [GINR96] was on representing and reasoning about actions. In this proposal based on epistemic description logics, a distinction is made between two kinds of description logic roles: standard static roles and functional action roles. Concept inclusions employing a knowledge operator are used to specify pre- and post-conditions of actions. This work was extended in [GINR97] in order to model sensing actions. However, the frame problem is handled properly only for sensing actions and neglected for the world altering ones. The authors extended the DL system CLASSIC in order to handle epistemic operators and even developed a mobile robot with planning support.

Recent Results

Recent results on actions in description logics were inspired by the formalisms and results presented in this thesis, or came as their follow-up.

In [LLMW06c], Liu et al. considered the problem of updating ABoxes when the state changes. It was assumed that changes are described at an atomic level like in $\mathfrak{A}_1$, i.e., in terms of *updates* consisting of possibly negated ABox assertions that involve only atomic concepts and roles. For an ABox $\mathcal{A}$ and an update $\mathcal{U}$, the updated ABox $\mathcal{A} * \mathcal{U}$ has exactly the models $\mathcal{I}'$ obtained by updating all models $\mathcal{I}$ of $\mathcal{A}$ with $\mathcal{U}$. Thus, the updated ABox $\mathcal{A} * \mathcal{U}$ has exactly the same consequences as those of applying $\mathcal{U}$ in $\mathcal{A}$. Obviously, updating ABoxes enables an explicit representation of updated states and thus follows the *progression* approach

to deciding projection, an alternative to the regression-based approach considered in this work. It turns out that DLs have to include nominals and the "@" constructor known from hybrid logic (or, equivalently, admit Boolean ABoxes) for updated ABoxes to be expressible. Algorithms to compute updated ABoxes in DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ were devised. Moreover, it was shown that an exponential blowup in the size of the whole input (original ABox together with update information) cannot be avoided unless every PTIME problem is LOGTIME-parallelizable. An exponential blowup in the size of the original ABox can be avoided by introducing abbreviations for complex concepts in an acyclic TBox.

In [GLPR06], De Giacomo et al. investigated ABox updates in the description logic *DL-Lite* in the presence of concept inclusions, under PMA semantics. *DL-lite* is a family of lightweight DLs. The *DL-Lite* version considered in [GLPR06] allows for unqualified existential restrictions ($\exists R$), concept negation, inverse and functional roles; concept inclusions $C \sqsubseteq D$ may contain negation only on the right hand-side. The advantage of *DL-Lite* is that basic reasoning problems are computationally tractable. It is shown that updates of *DL-Lite* ABoxes can be computed in polynomial time, and that they are of polynomial size in the size of the original knowledge base. However, in order to express updates, the authors allowed parameterized ABox assertions. Intuitively, such an assertion $C(z)$ describes that C holds "somewhere" in the domain. Without parameterized assertions, ABox updates are not expressible in *DL-Lite*, just like in case of more expressive DLs. In order to overcome this problem, the authors in [GLPR07] propose to compute maximal approximations of ABox updates in *DL-lite*. This means to compute *DL-lite* ABoxes whose set of models is a minimal set containing interpretations obtained by updating models of the original ABox. It is shown in [GLPR07] that maximal approximated ABox updates exist in *DL-lite* and that the algorithm from [GLPR06] can be adapted to compute approximated updates in polynomial time. Based on a (radical) functional view of ontologies, first ideas on how to define actions and GOLOG-like programs over *DL-Lite* ontologies were presented in [CGLR07].

In [GS07], Gu and Soutchanski considered a decidable fragment of SitCalc based on C^2, a two-variable fragment of first order logic with counting quantifiers. It is assumed that changes may occur only at an atomic level, like in $\mathfrak{A}_1$. The logic C^2 is more expressive than the description logics we consider in the present thesis. Its expressiveness corresponds to that of the description logic $\mathcal{ALCQIO}$ extended with role inclusions and Boolean operators on roles. [GS07] follows the axiomatizing tradition of the Situation Calculus, while the initial theory, pre-condition, and successor state axioms are written in C^2. Moreover, decidability of projection and executability are proved by showing that the regression approach to projection from [Rei01] works within the decidable two-variable fragment, by employing a careful substitution of variables.

In [DT07], Drescher and Thielscher investigated a decidable fragment of the Fluent Calculus based on description logics. In this fragment, ABoxes are used as state descriptions. The semantics for ABox updates is captured by Fluent Calculus state update axioms. This lays theoretical foundations for a practical action programming language built on top of description logic reasoners.

1.4 Description Logics with Concrete Domains

1.4.1 Concrete Domains

Overview

In order to use description logics in an application, it is crucial to identify a DL that is sufficiently expressive to represent the relevant notions of the application domain, but for which reasoning is still decidable. For several relevant applications of DLs, there is a need for DLs that can represent and reason about information of a more "concrete" nature, such as weights, amounts, durations, and spatial extensions.

As an example, consider an ontology designed to run an on-line business in the Semantic Web. In order to define the notion of a valid credit card in such an ontology, we need a means of storing the expiry year of the credit card and of checking whether it is greater than the current year. In ontologies for the life sciences, numbers and relations are of elementary importance for the representation of measurements, clinical findings, patients' vital parameters, or classifications based on those. In geo-spatial ontologies, it is highly desirable that the relations such as spatially_contains and spatially_overlaps are no standard roles, but "concrete" spatial relations, where spatial reasoning applies to.

The standard way of integrating numbers and other datatypes (spatial, temporal, etc.) into description logics is to divide the set of logical objects into two disjoint sets, one containing *abstract* objects and the other containing *concrete* objects. Real numbers and spatial regions would fall into the second set. Abstract objects can be related to concrete ones via functional *concrete features*, such as has_magnitude or has_area. Relations between concrete objects are described by a set of domain-specific predicates. The pair consisting of a set of concrete objects and predicates with a fixed extension over this set is called a *concrete domain* [BH91]. Concrete domains can be, for example:

- *Numerical:* The set of natural numbers $\mathbb{N}$ with the following predicates: unary $\geq_0$, binary $=$ and $>$.
- *Spatial:* A set of regions in the real plane with RCC8 [RCC92] spatial relations as predicates. RCC8 relations between regions are: eq (equal), dc (disconnected), ec (externally connected), po (partially overlap), as well as tpp (tangential proper part), ntpp (non-tangential proper part) and their inverses tppi and ntppi.
- *Temporal:* A set of real intervals with temporal relations of Allen's interval algebra [All83] as predicates. Allen relations between intervals are $=$ (equality), b (before), a (after), as well as: m (meet), o (overlap), d (during), s (starts), f (finishes) and their inverse relations mi, oi, di, si, fi.

A sequence of *abstract features*, i.e. functional roles, followed by a single concrete feature ($f_1 \cdots f_k g$) is called a *path*. By means of paths one can refer not only to concrete properties of an object, but also to concrete properties of other objects related to that object. For example, we may talk about the age of one's mother by using the sequence of features (mother age). It is also common to allow paths of the form rg, where r is a standard role.
The integration of concrete domains into description logics is usually achieved by adding a *concrete domain concept constructor* of the form $\exists u_1, \ldots, u_n.P$ (and $\forall u_1, \ldots, u_n.P$), where $u_1, \ldots, u_n$ are paths, and P is an n-ary predicate from the concrete domain. The intended interpretation is that u_i-values are related by the predicate P. For example, such an integration of the basic DL $\mathcal{ALC}$ with a concrete domain $\mathcal{D}$ is denoted by $\mathcal{ALC}(\mathcal{D})$. Returning to

the motivating example above, the notion of a a valid credit card might then be defined as:

$$\mathsf{CreditCard} \sqcap \exists \mathsf{expiry\_year}, \mathsf{current\_year}. \geq .$$

Relevant Existing Work

Concrete domains in the context of DLs were first introduced in [BH91], where the first tableau algorithm for deciding consistency of $\mathcal{ALC}(\mathcal{D})$-ABoxes was developed. The algorithm is independent of a particular concrete domain $\mathcal{D}$. Instead, it is required that $\mathcal{D}$ is admissible – which in particular means that deciding satisfiability of predicate conjunctions in $\mathcal{D}$ is decidable. Moreover, it is shown that two and more admissible concrete domains can be combined into one, retaining admissibility. It is proved in [Lut02b] that reasoning in $\mathcal{ALC}(\mathcal{D})$ is PSPACE-complete if reasoning with the concrete domain $\mathcal{D}$ is in PSPACE.

Various extensions of $\mathcal{ALC}(\mathcal{D})$ were treated in [HLM98, HLM99, Lut04b, Lut02b], including extensions with acyclic concept definitions, feature agreements, and a role-forming concrete domain operator. Even seemingly harmless extensions were shown to lead to NEXPTIME-complete reasoning and some of them even to undecidability.

It is shown in [Lut04b] that combining concrete domains with general TBoxes easily leads to undecidability. For example, the basic DL $\mathcal{ALC}$ extended with general TBoxes and a rather inexpressive concrete domain based on the natural numbers together with equality and incrementation predicates is undecidable, see also the survey paper [Lut03]. In view of this discouraging result, it was a natural question whether there are any useful concrete domains that can be combined with general TBoxes in a decidable DL. A positive answer to this question has been given in [Lut04a] and [Lut02a], where two such well-behaved concrete domains are identified: a temporal one based on the Allen relations for interval-based temporal reasoning, and a numerical one based on the real numbers equipped with various unary and binary predicates, such as "$\leq$", "$>_5$", and "$\neq$". Using an automata-based approach, it has been shown in [Lut04a, Lut02a] that reasoning in the DLs $\mathcal{ALC}$ and $\mathcal{SHIQ}$ extended with these concrete domains and general TBoxes is decidable and, more precisely, EXPTIME-complete.

Despite a number of theoretical results on concrete domains, only very limited concrete domains have found their way into actual implementations of DL reasoners. One reason for this may be the lack of practicable algorithms. One exception is the reasoning procedure presented in [HS01], designed for $\mathcal{SHOQ}(\mathcal{D})$, an expressive DL with a numerical concrete domain underlying the web ontology language OWL [BvHH+04]. However, paths appearing in the definition of concepts are restricted to concrete features, and only unary predicates (called *datatypes*) are permitted. The same restrictions apply to datatypes in the current version of OWL. The state-of-the-art OWL DL reasoners FaCT++ [TH06], PELLET [SPG+07], and RACERPRO [HM01b, HMW01] support simple numerical datatypes and strings within the above limitations. Nevertheless, the W3C OWL 1.1 working group has expressed interest in an extension to more powerful concrete domain predicates.

1.4.2 Contributions and Results

For several relevant applications of DLs such as the Semantic Web, life sciences and reasoning about ER and UML diagrams, there is a need for DLs that include, among others, both *concrete domains* and *general TBoxes* [BHS03, CLN98, Lut02c].

The purpose of our work is to advance the knowledge about decidable DLs with both concrete domains and general TBoxes. Our contribution is two-fold: first, instead of focusing on particular concrete domains as in previous work [Lut04a, Lut02a], we identify a *general* property of concrete domains, called ω-admissibility, that is sufficient for proving decidability of DLs equipped with concrete domains and general TBoxes. For the definition of ω-admissibility, we concentrate on a particular family of concrete domains: *constraint systems*. Roughly, a constraint system is a concrete domain that only has binary predicates interpreted as jointly exhaustive and pairwise disjoint (JEPD) relations. Intuitively, a constraint system is ω-admissible if the satisfiability problem for finite networks (i.e., sets of constraints) is decidable, and if finite and countably infinite unions of satisfiable networks are also satisfiable. We show that two useful constraint systems that are ω-admissible: a temporal one based on the total ordering on real numbers and the Allen relations [All83], and a spatial one based on the real plane and the RCC8 relations [EF91, Ben97, RCC92]. The proof of ω-admissibility turns out to be relatively straightforward in the Allen case, but is somewhat cumbersome for RCC8. We believe that there are many other useful constraint systems that can be shown to be ω-admissible.

The second part of our contribution is to develop a *tableau algorithm* for DLs with both general TBoxes and concrete domains. Since the existing decidability proofs for these logics use a theoretical automata-based approach [Lut04a, Lut02a], our tableau algorithm comes as the first implementable reasoning procedure for DLs with general TBoxes and concrete domains. Informally, tableau algorithms decide whether a concept is satisfiable by trying to construct a model for it. We employ such an algorithm to establish a general decidability result for the concept satisfiability and subsumption problem in $\mathcal{ALC}$ equipped with general TBoxes and any ω-admissible concrete domain. In particular, we obtain that satisfiability and subsumption are decidable w.r.t. general $\mathcal{ALC}$ TBoxes extended by the Allen relations as first established in [Lut04a]. As a new result, we prove decidability of analogous decision problems w.r.t. general $\mathcal{ALC}$ TBoxes extended by the RCC8 relations as a concrete domain. Due to the interleaving influences of concrete domains and general concept inclusions, we had to invent a sophisticated *blocking* technique in order to prevent the explicit creation of an infinite number of domain elements, and thus ensure termination of the algorithm. In contrast to existing tableau algorithms [HMW01, HS01], we do not impose any restrictions on the concrete domain constructor. As state-of-the-art DL reasoners such as FaCT++ [TH06] and RacerPro [HM01b] are based on tableau algorithms similar to the one described in this chapter, we view our algorithm as a first step towards a practical implementation of description logics with (ω-admissible) concrete domains and general TBoxes. In particular, we identify an expressive fragment of our logic that should be easily integrated into existing DL reasoners.

1.5 Structure of the Thesis

- In Chapter 2 we introduce the basics of description logics: their syntax, semantics, and standard reasoning problems. Moreover, we present formal preliminaries of the Situation Calculus.

- In Chapter 3 we introduce the generic DL action formalism $\mathfrak{A}_1$ that admits atomic concepts in the post-conditions of actions and a background knowledge in the form of acyclic TBoxes. We also define the semantics of actions as well as the standard reason-

ing tasks executability and projection. We show that the formalism $\mathfrak{A}_1$ can be viewed as a fragment of the Situation Calculus and thus inherits SitCalc's solution to the frame problem. In this chapter we focus on instantiations of $\mathfrak{A}_1$ with propositionally closed description logics $\mathcal{L}$. We cover the case of $\mathcal{ALC}$ and its extensions with (all possible combinations of) number restrictions, inverse roles, and nominals. We show that projection in $\mathfrak{A}_1$ instantiated with $\mathcal{L}$ can be polynomially reduced to ABox consequence in $\mathcal{LO}$, the extension of $\mathcal{L}$ with nominals. Finally, we show that the additional computational complexity (sometimes) caused by the introduction of nominals cannot be avoided, since standard DL reasoning in $\mathcal{LO}$ can polynomially be reduced to projection in $\mathcal{L}$.

- In Chapter 4, we consider instantiations of $\mathfrak{A}_1$ with lightweight description logics as well as several extensions of $\mathfrak{A}_1$. We show that deciding projection in $\mathfrak{A}_1$ instantiated with $\mathcal{EL}$ is PSPACE-hard, and thus not easier than for $\mathcal{ALC}$. Moreover, projection for $\mathcal{EL}$ without TBoxes is shown to be co-NP-complete. Considering possible extensions, we show that the semantics of $\mathfrak{A}_1$ can easily be extended to account for ramifications introduced by role inclusions. Moreover, we show that the same holds for the reduction of projection to ABox consequence from the previous chapter. However, we show that extensions involving transitive roles or acyclic TBoxes introduce semantic problems. Finally, allowing complex concepts in post-conditions of $\mathfrak{A}_1$-actions and adopting PMA semantics is shown to lead to computational problems.

- In Chapter 5 we introduce the action formalism $\mathfrak{A}_2$ that allows complex concepts in post-conditions of actions and supports general TBoxes as state constraints. Since $\mathfrak{A}_2$-actions contain complex occlusion patterns for fine-tuning the ramifications, consistency of actions is introduced as an important reasoning task besides executability and projection. We show that the formalism $\mathfrak{A}_2$ generalizes $\mathfrak{A}_1$. Moreover, we show that, for many standard propositionally closed DLs, the reasoning problems executability and projection in $\mathfrak{A}_2$ are decidable. For DLs contained in $\mathcal{ALCIO}$, we use a type-elimination method to show that projection in $\mathfrak{A}_2$ is EXPTIME-complete. For the DLs $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$, we show how to reduce projection to ABox consequence in $\mathcal{ALCQIO}$ extended with Boolean operations on roles, thus proving that projection for these logics is co-NEXPTIME-complete. We show that the weak kind of action consistency can be reduced to projection, while strong consistency is undecidable already for $\mathcal{ALC}$.

- In Chapter 6 we investigate planning in $\mathfrak{A}_1$. We introduce the notion of a planning task and define the plan existence problem. We show that the plan existence problem is decidable for actions described in fragments of $\mathcal{ALCQIO}$. More precisely, we show that its computational complexity coincides with the one of projection for DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ where operators contain only unconditional post-conditions. If we allow for conditional post-conditions the plan existence problem is shown to be in 2-EXPSPACE. Finally, we show that hardness results for propositional planning carry over to planning in the lightweight description logic $\mathcal{EL}$.

- Chapter 7 is dedicated to description logics with concrete domains. We introduce a special type of concrete domains called constraint systems and define ω-admissibility – the property of constraint systems which ensures that they can be combined with general TBoxes in a decidable DL. We show that some useful concrete domains, such

as a spatial one based on RCC8 and a temporal one based on Allen's relations, are ω-admissible. Moreover, we introduce the description logic $\mathcal{ALC}(\mathcal{C})$ that incorporates constraint systems and general TBoxes. The tableau algorithm for deciding satisfiability in $\mathcal{ALC}(\mathcal{C})$ equipped with an ω-admissible constraint system is developed. Finally, we discuss the feasibility of our algorithm and identify a fragment for which the tableau algorithm is implementable in a particularly straightforward way.

- In Chapter 8 we give a summary of our results and discuss possible future extensions of our work.

Most of the results presented in this work have already been published. The first results on integrating expressive description logics with action formalisms from Chapter 3 were published in [BLM$^+$05b, BLM$^+$05c, BLM$^+$05a]. The results on projection in $\mathcal{EL}$ from Chapter 4 were published in [LLM08]. The results on reasoning about actions in the presence of general TBoxes from Chapter 5 were published in [LLMW06b, LLMW06a], while the results on planning presented in Chapter 6 appeared in [Mil07]. Finally, the results on DLs with concrete domains and general TBoxes from Chapter 7 were published in [LM05a, LM05b, LM07]. The results on ABox updates in expressive DLs mentioned in the "related work" Section 1.3.3 and published in [LLMW06c] are going to become a part of the PhD dissertation of Hongkai Liu.

Chapter 2

Formal Preliminaries

In Section 2.1 we give basics of description logics that are of interest for the present work, while in Section 2.2 we give an introduction to the Situation Calculus.

Throughout this work, we will use $\#S$ or $|S|$ to denote the cardinality of a set S, while we use 2^S to denote the power set of S, i.e. $2^S = \{T \mid T \subseteq S\}$.

2.1 Description Logics

In description logics, concepts are inductively defined with the help of a set of constructors which determine the expressive power of the specific DL. We introduce the constructors available in $\mathcal{ALCQIO}$ and explain how its fragments discussed in this work can be obtained by omitting constructors.

Definition 2.1.1 ($\mathcal{ALCQIO}$ Syntax). Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be disjoint and countably infinite sets of *concept names*, *role names*, and *individual names*. A *role* is either a role name or the inverse r^- of a role name r. The set of $\mathcal{ALCQIO}$-concepts is the smallest set satisfying the following properties:

- each concept name $A \in \mathsf{N_C}$ is a concept;
- if C and D are concepts, r is a role, a an individual name, and n a natural number, then the following are also concepts:

$$\begin{array}{rl} \{a\} & (\textit{nominal}) \\ \neg C & (\textit{negation}) \\ C \sqcap D & (\textit{conjunction}) \\ C \sqcup D & (\textit{disjunction}) \\ \exists r.C & (\textit{existential restriction}) \\ \forall r.C & (\textit{universal restriction}) \\ (\geqslant n\ r\ C) & (\textit{at-most number restriction}) \\ (\leqslant n\ r\ C) & (\textit{at-least number restriction}) \end{array}$$

△

It is convenient to introduce some abbreviations. As usual, we use the Boolean standard abbreviations $\rightarrow$ and $\leftrightarrow$. We use $(= n\ r\ C)$ to abbreviate the conjunction of $(\geqslant n\ r\ C)$ and

Constructor	$\mathcal{EL}$	$\mathcal{EL}^{(\neg)}$	$\mathcal{ELU}$	$\mathcal{ALC}$
$\sqcap$, $\exists$, $\top$	•	•	•	•
$\neg A$		•		•
$\sqcup$			•	•
$\forall$, $\neg$				•

Figure 2.1: $\mathcal{ALC}$ and its "$\mathcal{EL}$"-like sublanguages.

Sym.	Constr.	$\mathcal{ALC}$	$\mathcal{ALCO}$	$\mathcal{ALCQ}$	$\mathcal{ALCI}$	$\mathcal{ALCQO}$	$\mathcal{ALCIO}$	$\mathcal{ALCQI}$	$\mathcal{ALCQIO}$
$\mathcal{Q}$	$(\leqslant n\ r\ C)$ $(\geqslant n\ r\ C)$			•		•		•	•
$\mathcal{I}$	r^-				•		•	•	•
$\mathcal{O}$	$\{a\}$		•			•	•		•

Figure 2.2: Description logics between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$.

$(\leqslant n\ r\ C)$. Additionally, we use $\top$ (*top*) for a propositional tautology and $\bot$ (*bottom*) for $\neg\top$.

The DL that allows only for negation ($\neg$), conjunction ($\sqcap$), disjunction ($\sqcup$), and universal ($\forall$) and existential ($\exists$) restriction is called $\mathcal{ALC}$. The DL $\mathcal{ALC}$ is the smallest propositionally closed description logic and usually considered to be a "prototypical" DL. In our work, we will also consider several description logics that are less expressive than $\mathcal{ALC}$. The DL that allows only for conjunction ($\sqcap$), existential restriction ($\exists$), and top ($\top$) is called $\mathcal{EL}$, while $\mathcal{EL}^{(\neg)}$ and $\mathcal{ELU}$ denote its extensions with atomic negation ($\neg A$) – i.e., negation that may occur only in front of concept names, and disjunction ($\sqcup$), respectively. The features of the mentioned languages are summarized in Figure 2.1.

Considering extensions of $\mathcal{ALC}$, the availability of additional constructors is indicated by concatenation of a corresponding letter: $\mathcal{Q}$ stands for number restrictions; $\mathcal{I}$ stands for inverse roles, and $\mathcal{O}$ for nominals. This explains the name $\mathcal{ALCQIO}$, and also allows us to refer to its sublanguages as indicated in Figure 2.2.

If a language $\mathcal{L}_1$ is a sublanguage of a language $\mathcal{L}$ (including $\mathcal{L}$), and $\mathcal{L}$ is a sublanguage of a language $\mathcal{L}_2$ (including $\mathcal{L}_2$), we say that $\mathcal{L}$ is *between* $\mathcal{L}_1$ and $\mathcal{L}_2$. Moreover, for a DL $\mathcal{L}$, by $\mathcal{L}$-concepts we denote concepts that are built by using only constructors provided by $\mathcal{L}$.

In order to refer to the computational complexity of reasoning tasks later on, we need to define the *size* of concepts. Intuitively, the size of a concept C is the number of characters needed to write it down.

Definition 2.1.2 (Concept Size). The size of concepts over $\mathsf{N_I}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ is inductively defined as follows:

- $|\top| = |\bot| = |A| = |\{a\}| := 1$, for all $A \in \mathsf{N_C}$ and $a \in \mathsf{N_I}$;
- $|C \sqcap D| = |C \sqcup D| := |C| + |D| + 1$; $|\neg C| = |\forall r.C| = |\exists r.C| := |C| + 1$, for all concepts C and D and roles r;
- $|(\geqslant n\ r\ C)| = |(\leqslant n\ r\ C)| = [\log_2(max\{2, n\})] + |C| + 1$, for all natural numbers n, roles r, and concepts C.

△

Note that, usually, binary coding is assumed for number restrictions.

Throughout this thesis, we will use the notion of a *subconcept* of a concept. This notion is introduced in the next definition.

Definition 2.1.3 (Subconcepts). The set of subconcepts $\mathsf{sub}(E)$ of a concept E is inductively defined as follows:

- $\mathsf{sub}(E) := \{E\}$, if E is $\top$, $\bot$, a concept name A, or a nominal $\{a\}$;
- $\mathsf{sub}(E) := \{E\} \cup \mathsf{sub}(C) \cup \mathsf{sub}(D)$, if E is of the form $C \sqcap D$ or $C \sqcup D$;
- $\mathsf{sub}(E) := \{E\} \cup \mathsf{sub}(C)$, if E is of the form $\neg C$, $\forall r.C$, $\exists r.C$, $(\geqslant n\ r\ C)$, or $(\leqslant n\ r\ C)$.

△

The semantics of concepts is defined w.r.t. a *model-theoretic semantics*, meaning that concepts are interpreted as subsets of the interpretation domain.

Definition 2.1.4 ($\mathcal{ALCQIO}$ Semantics). An interpretation $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is a mapping that assigns

- to each concept name A, a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$;
- to each individual name a, an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$;
- to each role name r, a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

The interpretation of inverse roles and complex concepts is then defined as follows:

$$\begin{aligned}
(r^-)^{\mathcal{I}} &= \{(e,d) \mid (d,e) \in r^{\mathcal{I}}\} \\
(\{a\})^{\mathcal{I}} &= \{a^{\mathcal{I}}\} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists r.C)^{\mathcal{I}} &= \{d \mid \text{there exists } e \text{ such that } (d,e) \in r^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\} \\
(\forall r.C)^{\mathcal{I}} &= \{d \mid \text{for all } e \text{ such that } (d,e) \in r^{\mathcal{I}} \text{ it holds that } e \in C^{\mathcal{I}}\} \\
(\leqslant n\ r\ C)^{\mathcal{I}} &= \{d \mid \#\{e \in C^{\mathcal{I}} \mid (d,e) \in r^{\mathcal{I}}\} \leq n\} \\
(\geqslant n\ r\ C)^{\mathcal{I}} &= \{d \mid \#\{e \in C^{\mathcal{I}} \mid (d,e) \in r^{\mathcal{I}}\} \geq n\}
\end{aligned}$$

An interpretation $\mathcal{I}$ is called a *model* of a concept C if $C^{\mathcal{I}} \neq \emptyset$ △

Note that the existential restriction $\exists r.C$ is an abbreviation for $(\geqslant 1\ r\ C)$, while the universal restriction $\forall r.C$ abbreviates $(\leqslant 0\ r\ \neg C)$. For this reason, in sublanguages of $\mathcal{ALCQIO}$ that provide number restrictions, we will not treat existential and universal restrictions explicitly.

Although not standard in description logics, we will often require interpretations to respect the *unique name assumption (UNA)* on individual names, since it is natural in reasoning about actions.

Definition 2.1.5 (Unique Name Assumption (UNA)). An interpretation $\mathcal{I}$ respects *unique name assumption (UNA)* on individual names iff for all $a, b \in \mathsf{N_I}$, $a^{\mathcal{I}} = b^{\mathcal{I}}$ implies $a = b$. △

Note that, since the set of individual names $\mathsf{N_I}$ is countably infinite, interpretations that respect UNA on individual names, strictly speaking, also have to have *infinite* domains. In the coming chapters, in order to be able to work with *finite* domains, we may assume that interpretations interpret only individual names from a finite set of individuals Ind, and respect UNA on Ind.

Besides interpretations, we will use the notion of a *frame*. Intuitively, frames are interpretations that interpret only role names.

Definition 2.1.6 (Frame). A *frame* is a pair $(\Delta^{\mathcal{F}}, \cdot^{\mathcal{F}})$, where $\Delta^{\mathcal{F}}$ is a non-empty set and $\cdot^{\mathcal{F}}$ maps each role name r to a binary relation $r^{\mathcal{F}} \subseteq \Delta^{\mathcal{F}} \times \Delta^{\mathcal{F}}$. An interpretation $\mathcal{I}$ is said to be *based on* $\mathcal{F}$ if $\Delta^{\mathcal{F}} = \Delta^{\mathcal{I}}$ and $\cdot^{\mathcal{F}}$ and $\cdot^{\mathcal{I}}$ agree on the interpretation of all role names. A frame $\mathcal{F}$ *validates* a concept C if and only if $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ based on $\mathcal{F}$. △

In description logics, *TBoxes* are used to capture the background knowledge about the world. They come in several flavours: the most simple, *acyclic TBoxes*, introduce defined concept names as *abbreviations* for complex concepts, while the powerful *general TBoxes* introduce constraints of the form "for all domain elements where C holds, D holds as well."

Definition 2.1.7 (Concept Definition, Acyclic TBox). A *concept definition* is of the form $A \doteq C$, where A is a concept name and C a concept. A *TBox* is a finite set of concept definitions with unique left-hand sides. We say that a concept name A *directly uses* a concept name B w.r.t. $\mathcal{T}$ if there is a concept definition $A \doteq C \in \mathcal{T}$ with B occurring in C. Let *uses* be the transitive closure of directly uses. Then a TBox $\mathcal{T}$ is *acyclic* if no concept name uses itself w.r.t. $\mathcal{T}$. Otherwise, it is called *cyclic*.

An interpretation $\mathcal{I}$ *satisfies* a concept definition $A \doteq C$ (written $\mathcal{I} \models A \doteq C$) if $A^{\mathcal{I}} = C^{\mathcal{I}}$. $\mathcal{I}$ is called a *model* of a TBox $\mathcal{T}$, written $\mathcal{I} \models \mathcal{T}$, if $\mathcal{I}$ satisfies all concept definitions in $\mathcal{T}$. △

We call a concept name A *defined in* a TBox $\mathcal{T}$ if A occurs on the left-hand side of a concept definition in $\mathcal{T}$, and *primitive in* $\mathcal{T}$ otherwise. Note that, in the case of acyclic TBoxes, interpretations of primitive concept names and role names *uniquely* determine interpretations of defined concept names, which is *not* the case for cyclic TBoxes.

Note that acyclic TBoxes do not introduce additional expressiveness, as they can be compiled away by a process called *unfolding* – exhaustive replacement of defined concept names by their definitions. However, acyclic TBoxes enable an exponentially more succinct representation of concepts. Consider the following acyclic TBox $\mathcal{T}$:

$$\begin{aligned} A_0 &\doteq \exists r.A_1 \sqcap \exists s.A_1 \\ A_1 &\doteq \exists r.A_2 \sqcap \exists s.A_2 \\ &\vdots \\ A_n &\doteq \exists r.P \sqcap \exists s.P \end{aligned}$$

It is not difficult to see that when unfolding A_0 w.r.t. $\mathcal{T}$, we obtain a concept whose size is exponential in n.

Definition 2.1.8 (GCI, General TBox). A *general concept inclusion axiom (GCI)* is an expression of the form $C \sqsubseteq D$, where C and D are concepts. An expression $C \doteq D$ is an abbreviation for two GCIs $C \sqsubseteq D$ and $D \sqsubseteq C$. A *general TBox* $\mathcal{T}$ is a finite set of GCIs.

An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We denote satisfaction of a GCI $C \sqsubseteq D$ by an interpretation $\mathcal{I}$ with $\mathcal{I} \models C \sqsubseteq D$. An interpretation $\mathcal{I}$ is called a *model* of a general TBox $\mathcal{T}$, written $\mathcal{I} \models \mathcal{T}$, if $\mathcal{I}$ satisfies all GCIs in $\mathcal{T}$. △

It is obvious that general TBoxes generalize acyclic and cyclic ones. General TBoxes are highly desirable in knowledge representation, as they can capture complex constraints and dependencies in the application domain. However, they usually increase complexity of reasoning, and may introduce semantic and computational problems, as discussed in the coming chapters.

In DLs, a complete description of the state of the world corresponds to an interpretation. However, it is realistic to assume that only an incomplete knowledge is available. This incomplete knowledge about the current state of the world is represented in an ABox.

Definition 2.1.9 (ABox). A *concept assertion* is of the form $C(a)$, and a *role assertion* is of the form $r(a, b)$ or $\neg r(a, b)$, where $a, b \in \mathsf{N_I}$, C is a concept, and r a role. An *ABox assertion* (or just *assertion*) is a concept assertion or a role assertion. An *ABox* is a finite set of assertions. An interpretation $\mathcal{I}$ *satisfies* an assertion

$$\begin{array}{lll} C(a) & \text{iff} & a^{\mathcal{I}} \in C^{\mathcal{I}}; \\ r(a,b) & \text{iff} & (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}; \\ \neg r(a,b) & \text{iff} & (a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}. \end{array}$$

If φ is an assertion, then we write $\mathcal{I} \models \varphi$ iff $\mathcal{I}$ satisfies φ. An interpretation $\mathcal{I}$ is called a *model* of an ABox $\mathcal{A}$, written $\mathcal{I} \models \mathcal{A}$, if $\mathcal{I}$ satisfies all assertions in $\mathcal{A}$. △

To improve readability, we will sometimes write the assertion $C(a)$ in the form $a : C$. Negated role assertions are usually not considered in DL, but they are very useful as pre- and post-conditions in action definitions. As described below, reasoning with such assertions can easily be reduced to reasoning without them if the DL under consideration allows for universal restriction and atomic negation.

Although they will not be in the main focus of this work, we will also shortly consider *role boxes*.

Definition 2.1.10 (Role inclusion, Role box). A *role inclusion* is an expression of the form $r \sqsubseteq s$, where r and s are (possibly inverse) roles. A *role box* is a finite set of role inclusions.

An interpretation $\mathcal{I}$ *satisfies* a role inclusion $r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is called a *model* of a role box $\mathcal{R}$, written $\mathcal{I} \models \mathcal{R}$, if $\mathcal{I}$ satisfies all role inclusions in $\mathcal{R}$. △

Role boxes are also called *role hierarchies* in the literature, and description logics that provide for role boxes usually contain the letter $\mathcal{H}$ in their name. In this work, we will consider the description logic $\mathcal{ALCQIOH}$, which extends $\mathcal{ALCQIO}$ with role boxes.

Throughout this work we will use abbreviations such as $\mathcal{I} \models \mathcal{T}, \mathcal{A}$ for $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$; or $\mathcal{I} \models \mathcal{T}, \mathcal{R}$ for $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{R}$. In the next definition, we define the size of TBoxes, ABoxes and role boxes.

Definition 2.1.11 (Size of TBoxes, ABoxes and role boxes). The size of a (general) TBox $\mathcal{T}$ is defined as follows:

$$|\mathcal{T}| = \sum_{C \doteq D \in \mathcal{T}} (|C| + |D|) + \sum_{C \sqsubseteq D \in \mathcal{T}} (|C| + |D|)$$

The size of an ABox $\mathcal{A}$ is defined as $|\mathcal{A}| = \sum_{\varphi \in \mathcal{A}} |\varphi|$, where $|\varphi| = 2$ if $\varphi = r(a,b)$ or $\varphi = \neg r(a,b)$ and $|\varphi| = |C|$ if $\varphi = C(a)$.

The size of a role box $\mathcal{R}$ is defined as the cardinality of set $\mathcal{R}$. △

Various reasoning problems are considered for DLs. For the purpose of this work, it is sufficient to introduce only four of them: concept satisfiability, concept subsumption, ABox consistency, and ABox consequence.

Definition 2.1.12 (DL Reasoning Problems). Let C and D be a concepts, $\mathcal{A}$ an ABox, and $\mathcal{T}$ a (general) TBox. Then

- C is *satisfiable* w.r.t. the TBox $\mathcal{T}$ iff there exists an interpretation $\mathcal{I}$ that is a model of both C and $\mathcal{T}$;
- C is *subsumed* by D w.r.t. the TBox $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) iff for all models $\mathcal{I}$ of $\mathcal{T}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$;
- $\mathcal{A}$ is *consistent* w.r.t. the TBox $\mathcal{T}$ iff there exists an interpretation $\mathcal{I}$ that is a model of both $\mathcal{T}$ and $\mathcal{A}$.
- an assertion φ is a *consequence* of an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ (written $\mathcal{T}, \mathcal{A} \models \varphi$) if every model of $\mathcal{A}$ and $\mathcal{T}$ satisfies φ.

△

If the TBox $\mathcal{T}$ is empty in the afore defined reasoning problems, we simply drop the phrase "w.r.t. $\mathcal{T}$" instead of writing w.r.t. $\emptyset$. Similarly, we may define concept satisfiability and subsumption, as well as ABox consistency and consequence *w.r.t. a role box* $\mathcal{R}$, by requiring additionally that interpretations $\mathcal{I}$ are models of $\mathcal{R}$. Moreover, we will refer to ABox consistency and ABox consequence *under UNA*, to denote ABox consistency and consequence where only models $\mathcal{I}$ of the ABox respecting UNA on individual names are taken into account.

ABox consequence will play an important role in this work. As it is a slightly unusual DL reasoning problem, we briefly show that ABox consequence with negated role assertions, i.e. assertions of the form $\neg r(a,b)$, can be polynomially reduced to ABox consistency without negated role assertions, and vice versa. From these reductions, it follows that ABox consistency and ABox consequence are of the same complexity. We proceed in two steps:

Firstly, we reduce ABox consequence (under UNA) with negated role assertions to ABox consistency (under UNA) with negated role assertions, and vice versa. For an assertion φ, let $\neg\varphi = \neg C(a)$ if $\varphi = C(a)$, $\neg\varphi = \neg r(a,b)$ if $\varphi = r(a,b)$, and $\neg\varphi = r(a,b)$ if $\varphi = \neg r(a,b)$. Then φ is a consequence (under UNA) of $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff $\mathcal{A} \cup \{\neg\varphi\}$ is inconsistent (under UNA) w.r.t. $\mathcal{T}$. Conversely, an ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$ iff $\bot$ is not a consequence of $\mathcal{A}$ w.r.t. $\mathcal{T}$.

Second, since ABoxes with negated role assertions are slightly unusual, we show how to reduce ABox consistency with negated role assertions to ABox consistency without such assertions. Given an ABox $\mathcal{A}$, introduce a concept name X_a not used in $\mathcal{A}$ for each individual name a used in $\mathcal{A}$. Then replace each assertion $\neg r(a,b)$ with the two assertions $(\forall r.\neg X_b)(a)$ and $X_b(b)$. Clearly, the resulting ABox $\mathcal{A}'$ is consistent iff the original one is, and $\mathcal{A}'$ is of size linear in the size of $\mathcal{A}$.

Finally, ABox consistency under UNA can be reduced to ABox consistency without UNA in the following way: Given an ABox $\mathcal{A}$, and a TBox $\mathcal{T}$, let $a_1, \ldots, a_n$ be all different individual

names that appear in $\mathcal{A}$ and $\mathcal{T}$. We introduce $k = [\log_2(n+1)]$ concepts $X_1, \ldots, X_k$ not used in $\mathcal{A}$ and $\mathcal{T}$, and add assertion $X_i(a_j)$ if i-th bit of (the binary representation of) j is 1, and $\neg X_i(a_j)$ if i-th bit of j is 0. In other words, for each a_j, $X_1, \ldots, X_k$ represent the bits $1, \ldots, k$ of j. Thus, in models $\mathcal{I}$ of the resulting $\mathcal{A}'$, $a_j \neq a_l$ implies that $a_j^{\mathcal{I}} \neq a_l^{\mathcal{I}}$. Moreover, $\mathcal{A}'$ is of size polynomial in the size of $\mathcal{A}$.

Throughout the present work, we consider arbitrary, but fixed, sets of concept names $\mathsf{N_C}$, role names $\mathsf{N_R}$, and individual names $\mathsf{N_I}$. Moreover, concept names are usually denoted by uppercase letters, e.g. $A, B, \ldots$, while lowercase letters are used for roles, e.g. $r, s, \ldots$, and individuals, e.g. $a, b, \ldots$.

For a given DL $\mathcal{L}$, we speak of an $\mathcal{L}$-(general) TBox $\mathcal{T}$ iff only $\mathcal{L}$-concepts appear in concept definitions and GCIs in $\mathcal{T}$. Similarly, we speak of an $\mathcal{L}$-ABox $\mathcal{A}$ iff only $\mathcal{L}$-concepts appear in $\mathcal{A}$.

2.2 Situation Calculus

In order to establish a formal relation between our DL-based action formalisms and the Situation Calculus, we give a short introduction to this well-known standard action formalism. The *Situation Calculus (SitCalc)* is a language specifically designed for the representation of dynamically changing worlds, first introduced by McCarthy in 1963 [McC63]. Since then, many different versions of SitCalc were proposed. In this work, we relate to the standard version introduced by Reiter in [Rei01].

Formally, the language $\mathcal{L}_{sitcalc}$ is a three-sorted second-order theory with equality, with the three sorts being *actions*, *situations*, and *objects* for everything else depending on the domain of application. Besides the standard logical symbols, infinitely many constants of sort *object*, infinitely many variable symbols of each sort and infinitely many predicate variables of all arities, the alphabet of $\mathcal{L}_{sitcalc}$ includes: [1]

- two function symbols of sort *situation*:
  1. A constant $\mathbf{s_0}$, denoting the initial situation.
  2. A binary function symbol $do : action \times situation \rightarrow situation$. The intended interpretation is that situations are finite sequence of actions, and $do(a, s)$ denotes the sequence formed by adding action a to the sequence s.
- A binary predicate symbol $\sqsubset$: $situation \times situation$, defining an ordering relation on situations. Since the intended interpretation of situations is as finite action sequences, $s \sqsubset s'$ means that s is a proper subsequence of s'. Moreover, $s \sqsubseteq s'$ abbreviates $s \sqsubset s' \vee s = s'$.
- A binary predicate symbol $Poss : action \times situation$. The intended interpretation of $Poss$ is that it is possible to apply action a in situation s.
- For each $n \geq 0$, a finite or countably infinite number of function symbols of sort $(action \cup object)^n \rightarrow action$. They are called *action functions*, and are used to denote actions like $move(x, y)$.

[1] Note that in [Rei01], $\mathcal{L}_{sitcalc}$ also includes functions of sort $(action \cup object)^n \rightarrow object$ and $(action \cup object)^n \times situation \rightarrow object$ (so-called functional fluents). We leave them out as DLs are function-free, and thus a "function-free" fragment of SitCalc is sufficient to embed our DL action formalism into SitCalc; for details see Section 3.1.3.

- For each $n \geq 0$, countably infinitely many predicate symbols with arity n, and sorts $(action \cup object)^n$. They are used to denote situation independent relations.
- For each $n \geq 0$, a finite or countably infinite number of predicate symbols with arity $n+1$, and sorts $(action \cup object)^n \times situation$. They are called *fluents*, and are used to denote properties whose truth depends on the situation. For example, "likes(b, c, s)" would be read as "b likes c in situation s".

The main purpose of the Situation Calculus is to provide a framework for axiomatizing (i) in which situations an action can be applied and (ii) the effect that actions have on situations. The former is achieved through so-called *action pre-condition axioms* while the effects of actions are described using so-called *successor state axioms*.

In order to define these axioms, we need to introduce the notion of a *uniform formula*. A formula of $\mathcal{L}_{sitcalc}$ is said to be *uniform* in term σ of sort situation iff it does not mention the predicates *Poss* or $\sqsubset$, it does not quantify over variables of sort situation, it does not mention equality on situations, and whenever it mentions a term of sort situation in the situation argument position of a fluent, then that term is σ.

A *basic action theory* $\mathcal{D}$ is defined in [Rei01] as

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{\mathbf{s_0}} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una},$$

where: [2]

- Σ is the set of the four *foundational axioms* for situations,

$$do(a_1, s_1) = do(a_2, s_2) \rightarrow a_1 = a_2 \wedge s_1 = s_2$$
$$\forall P.\big(P(\mathbf{s_0}) \wedge \forall a, s.(P(s) \rightarrow P(do(a, s))) \rightarrow \forall s.P(s)\big)$$
$$\neg s \sqsubset \mathbf{s_0}$$
$$s \sqsubset do(a, s') \leftrightarrow s \sqsubseteq s'$$

- $\mathcal{D}_{ss}$ is a set of *successor state axioms*, i.e. sentences of the form

$$F(x_1, \ldots, x_n, do(a, s)) = \Phi_F(x_1, \ldots, x_n, a, s)$$

  where F is an $(n+1)$-ary relational fluent, and $\Phi_F(x_1, \ldots, x_n, a, s)$ is a formula uniform in s, all of whose free variables are among $a, s, x_1, \ldots, x_n$.

- $\mathcal{D}_{ap}$ is the set of *action pre-condition axioms*, i.e. sentences of the form

$$Poss(A(x_1, \ldots, x_n), s) = \Pi_A(x_1, \ldots, x_n, s)$$

  where A is an n-ary action function symbol, and $\Pi_A(x_1, \ldots, x_n, s)$ is a formula uniform in s, all of whose free variables are among $s, x_1, \ldots, x_n$.

- $\mathcal{D}_{una}$ is the set of *unique name axioms* for actions;

- $\mathcal{D}_{\mathbf{s_0}}$ is the description of the initial situation. It is a set of first-order sentences that are uniform in $\mathbf{s_0}$, so that $\mathbf{s_0}$ is the only term of sort situation mentioned by the sentences of $\mathcal{D}_{\mathbf{s_0}}$.

[2] Here, all free variables are assumed to be universally quantified.

The uniformity requirement on Φ_F in successor state axioms guarantees that the truth value of F in the successor situation $do(a, s)$ is determined by the current situation s. Similarly, the uniformity requirement on Π_A in pre-condition axioms ensures that the pre-conditions for the executability of the action A are determined entirely by the current situation s.

Successor state axioms of a special syntactic form are called *context-free*. These are of the form

$$F(x_1, \ldots, x_n, do(a, s)) \equiv (\gamma_F^+(x_1, \ldots, x_n, a) \vee F(x_1, \ldots, x_n, s)) \wedge \neg\gamma_F^-(x_1, \ldots, x_n, a),$$

where both $\gamma_F^+(x_1, \ldots, x_n, a)$ and $\gamma_F^-(x_1, \ldots, x_n, a)$ are independent of the situation s. Intuitively, a successor state axiom for a fluent F is context-free iff the truth value of F in the next situation $do(a, s)$ depends on the truth value of F in the current situation s, but is independent of the truth values of any other fluents in s.

Note that a basic action theory $\mathcal{D}$ as defined in [Rei01] does not include *state constraints*. In more expressive variants of SitCalc [LR94, Lin95], state constraints describe global properties that must hold in all situations. In general, they are of the form

$$\forall s. \forall x_1 \ldots \forall x_n. \Theta(x_1, \ldots, x_n, s)$$

where $\Theta(x_1, \ldots, x_n, s)$ is uniform in s.

Let $a_1, \ldots, a_n$ be action constants. The reasoning tasks executability and projection are defined as follows:

Executability. We introduce the following abbreviation:

$$executable(s) := \forall a, s' : \big((do(a, s') \sqsubseteq s) \rightarrow Poss(a, s')\big)$$

In the Situation Calculus, the sequence of actions $a_1, \ldots, a_n$ is *executable* in a situation s w.r.t. the basic action theory $\mathcal{D}$ if $\mathcal{D} \models executable(do(a_n, do(a_{n-1}, \cdots, do(a_1, s) \cdots)$.

Projection. Let $\varphi(s)$ be a Situation Calculus formula with one free situation-typed variable s. Then φ is a *consequence* of applying the sequence of actions $a_1, \ldots, a_n$ in the initial situation $\mathbf{s_0}$ described by $\mathcal{D}_{\mathbf{s_0}}$ w.r.t. the basic action theory $\mathcal{D}$ iff we have that $\mathcal{D} \models \varphi(do(a_n, do(a_{n-1}, \cdots, do(a_1, \mathbf{s_0}) \cdots)$.

Consequences in the Situation Calculus are often proved by a mechanism called *regression*. Regression is based on expressing a formula containing the situation $do(a, s)$ in terms of a formula containing the action a and the situation s, but not the situation $do(a, s)$. By iterating this procedure, one can end up with an equivalent formula (called *regressed formula*) containing only the initial situation $\mathbf{s_0}$. Proving consequences is usually simpler from this formula than from the original one, as one does not have to consider all axioms from the basic action theory $\mathcal{D}$, but only those related to the initial situation $\mathbf{s_0}$, i.e. axioms from $\mathcal{D}_{\mathbf{s_0}} \cup \mathcal{D}_{una}$. Naturally, axioms from $\Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap}$ are applied in order to compute the regressed formula. Regression is usually easier to apply if successor state axioms are context-free [Rei01].

Planning is normally not conceptualized within the Situation Calculus, but within planning formalisms, such as *STRIPS* [FN71]. However, it is possible to define the notion of a *plan* is SitCalc. Let $G(s)$ be a SitCalc formula with one free situation-typed variable s, and let σ be variable-free situation term. Then σ is a *plan* for G (relative to a basic action theory $\mathcal{D}$) iff $\mathcal{D} \models executable(\sigma) \wedge G(\sigma)$. One way to prove existence of a plan is to show that $\mathcal{D} \models \exists s.(executable(s) \wedge G(s))$.

Chapter 3

Action Formalism $\mathfrak{A}_1$: Simple Post-Conditions and Acyclic TBoxes

In this chapter we introduce the generic action formalism $\mathfrak{A}_1$ that corresponds to a fragment of the standard Situation Calculus. The corresponding fragment of SitCalc is obtained by restricting the logic for describing pre- and post-conditions as well as the state of the world to a certain decidable description logic $\mathcal{L}$.

Using a STRIPS-like syntax, atomic actions in $\mathfrak{A}_1$ are described as triples consisting of pre-conditions, occlusions (specifying atomic assertions that may change freely), and conditional post-conditions. ABox assertions are used to describe all three components, as well as states of the world. For example, by means of ABox assertions we may state that in the current world state, Dirk holds an electricity contract. Moreover, the pre-condition of obtaining a bank account is that the customer holds a proof of the address. The conditional post-conditions of this action state that, if the customer holds an employer's letter, the bank account comes with a credit card, and otherwise without it. In $\mathfrak{A}_1$, acyclic TBoxes are used to give concept definitions. For example, the proof of the address can be defined to be either an electricity contract or a lease. By disallowing defined or complex concepts in post-conditions of $\mathfrak{A}_1$-actions, we ensure that changes may occur only at an atomic level.

Semantics of actions is given via transition relation on DL interpretations. This transition relation ensures that extensions of primitive concept names and role names before and after action execution differ only if stated so by action post-conditions or occlusions. Possible action ramifications are related to extensions of TBox defined concept names: they may change after action execution, although this is not explicitly specified by action post-conditions.

Although $\mathfrak{A}_1$ is a generic formalism that can be instantiated with any decidable description logic, in this chapter we focus on its instantiations with propositionally closed fragments of $\mathcal{ALCQIO}$. The reason for choosing $\mathcal{ALCQIO}$ is that it forms the core of OWL DL, the description logic variant of OWL.

The rest of this chapter is organized as follows. In Section 3.1.1 we define syntax and semantics of $\mathfrak{A}_1$-actions. In Section 3.1.2 we define the standard reasoning problems executability and projection for $\mathfrak{A}_1$-actions. Executability is the problem of whether action pre-conditions are satisfied, i.e. whether the action is applicable in the current state, while projection is the problem of determining consequences of actions. Moreover, we show that

these two reasoning problems are mutually polynomially reducible in $\mathfrak{A}_1$. In Section 3.1.3, we show that the formalism $\mathfrak{A}_1$ instantiated with $\mathcal{ALCQIO}$ can be viewed as a fragment of the Situation Calculus and thus inherits SitCalc's solution to the frame problem. We perform a detailed investigation on how the choice of a description logic influences the complexity of reasoning in $\mathfrak{A}_1$. In Section 3.2.1 we show that, for description logics $\mathcal{L}$ between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$, projection in $\mathfrak{A}_1$ instantiated with $\mathcal{L}$ can be polynomially reduced to ABox consequence in $\mathcal{LO}$, the extension of $\mathcal{L}$ with nominals. Since standard reasoning in $\mathcal{ALCQIO}$ is supported by the efficient DL reasoners, this means that reasoning about action can be passed on to the standard DL reasoners. In Section 3.2.2, we show that the additional computational complexity (sometimes) caused by the introduction of nominals cannot be avoided, since standard DL reasoning in $\mathcal{LO}$ can polynomially be reduced to projection in $\mathcal{L}$. By combining the two reductions, we obtain tight complexity bounds for projection in $\mathfrak{A}_1$ for DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$, where the complexity ranges from PSPACE-complete to co-NEXPTIME-complete.

3.1 The Formalism

3.1.1 Action Descriptions

The framework for reasoning about actions proposed in this section is not restricted to a particular description logic, but can be instantiated with any description logic that seems appropriate for the application domain at hand. For simplicity, we concentrate on *ground actions*, i.e., actions where the input parameters have already been instantiated by individual names. *Parameterized* actions (operators), which contain variables in place of individual names, should be viewed as a compact representation of all its ground instances: an operator simply represents the set of all ground actions obtained from the parameterized action by replacing variables with individual names.

The handling of such parameterized actions takes place "outside" of our formalism and is not discussed in detail in the current chapter. We may safely restrict ourselves to ground actions since all the reasoning tasks considered in this chapter presuppose that parameterized actions have already been instantiated. For other tasks, such as planning, we will work directly with parameterized actions; see Chapter 6 for more details.

Definition 3.1.1 ($\mathfrak{A}_1$-Action). Let $\mathcal{T}$ be an acyclic TBox. An *atomic* $\mathfrak{A}_1$*-action* for $\mathcal{T}$ is a triple $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ which consists of

- a finite set pre of ABox assertions, the *pre-conditions*;
- a finite set occ of *occlusions* of the form $A(a)$ or $r(a, b)$, with A a primitive concept name in $\mathcal{T}$, r a role name, and $a, b \in \mathsf{N_I}$;
- a finite set post of *conditional post-conditions* of the form φ/ψ, where φ is an ABox assertion and ψ is a *primitive literal for* $\mathcal{T}$, i.e., an ABox assertion $A(a)$, $\neg A(a)$, $r(a, b)$, or $\neg r(a, b)$ with A a primitive concept name in $\mathcal{T}$ and r a role name.

A *composite* $\mathfrak{A}_1$*-action* for $\mathcal{T}$ is a finite sequence $\alpha_1, \ldots, \alpha_k$ of atomic actions for $\mathcal{T}$. △

We call post-conditions of the form $\top(a)/\psi$ *unconditional* and write just ψ instead. Moreover, an action of the form $\mathcal{U} = (\emptyset, \emptyset, \{\top(a)/\psi_1, \ldots, \top(a)/\psi_n\})$ is called *update (for a TBox* $\mathcal{T}$*)* and we write only $\mathcal{U} = \{\psi_1, \ldots, \psi_n\}$ instead.

Intuitively, the pre-conditions specify under which conditions the action is applicable. The conditional post-condition φ/ψ says that, if φ is true before executing the action, then ψ should be true afterwards. By the law of inertia, only those facts that are forced to change by the post-conditions should be changed by applying the action. However, it is well-known that enforcing this minimization of change strictly is sometimes too restrictive [Lif90, San94]. The rôle of occlusions is to indicate those primitive literals that can change arbitrarily.

In order to refer to the computational complexity of reasoning tasks later on, we need to define the size of $\mathfrak{A}_1$-actions:

Definition 3.1.2 (Size of $\mathfrak{A}_1$-Actions). The size of an atomic $\mathfrak{A}_1$-action $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ is defined with $|\alpha| = |\mathsf{pre}| + |\mathsf{occ}| + |\mathsf{post}|$, where $|\mathsf{pre}|$ and $|\mathsf{occ}|$ are sizes of the ABoxes pre and occ, and $|\mathsf{post}| = \sum_{\varphi/\psi \in \mathsf{post}}(|\varphi| + |\psi|)$. The size of a composite $\mathfrak{A}_1$-action $\alpha_1, \ldots, \alpha_n$ is defined with $|\alpha_1, \ldots, \alpha_k| = |\alpha_1| + \cdots |\alpha_k|$. △

Throughout this and the next chapter, only $\mathfrak{A}_1$-actions will be considered, and we just write "action" instead of "$\mathfrak{A}_1$-action".

Example 3.1.3. *To illustrate the definition of actions, consider the actions of opening a bank account and applying for child benefit in the UK. Suppose the pre-condition of opening a bank account is that the customer a is eligible for a bank account in the UK and holds a proof of address. Moreover, suppose that, if a letter from the employer is available, then the bank account comes with a credit card, otherwise not. This can be formalized by the following action α_1, for which the set of occlusions is empty:*

$$\begin{aligned}\mathsf{pre}: \ & \{\mathsf{Eligible\_bank}(a), \exists \mathsf{holds}.\mathsf{Proof\_address}(a)\}\\ \mathsf{post}: \ & \{\mathsf{holds}(a,b),\\ & \quad \exists \mathsf{holds}.\mathsf{Letter}(a)/\mathsf{B\_acc\_credit}(b),\\ & \quad \neg\exists \mathsf{holds}.\mathsf{Letter}(a)/\mathsf{B\_acc\_no\_credit}(b)\}\end{aligned}$$

Suppose that one can apply for child benefit in the UK if one has a child and a bank account. The action α_2 that offers this application then looks as follows, where again the set of occlusions is empty:

$$\begin{aligned}\mathsf{pre}: \ & \{\mathsf{parent\_of}(a,c), \exists \mathsf{holds}.\mathsf{B\_acc}(a)\}\\ \mathsf{post}: \ & \{\mathsf{receives\_c\_benef\_for}(a,c)\}\end{aligned}$$

The meaning of the concepts used in α_1 and α_2 are defined in the following acyclic TBox $\mathcal{T}$:

$$\begin{aligned}\mathsf{Eligible\_bank} &\doteq \exists \mathsf{permanent\_resident}.\{\mathsf{UK}\}\\ \mathsf{Proof\_address} &\doteq \mathsf{Electricity\_contract} \sqcup \mathsf{Lease}\\ \mathsf{B\_acc} &\doteq \mathsf{B\_acc\_credit} \sqcup \mathsf{B\_acc\_no\_credit}\end{aligned}$$

When defining the semantics of actions, we assume that states of the world correspond to interpretations. Thus, the semantics of actions can be defined by means of a transition relation on interpretations. Let $\mathcal{T}$ be an acyclic TBox, $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ an action for $\mathcal{T}$, and $\mathcal{I}$ an interpretation. For each primitive concept name A and role name r, set:

$$\begin{aligned}\alpha^{\mathcal{I}}_+(A) &:= \{b^{\mathcal{I}} \mid \varphi/A(b) \in \mathsf{post} \wedge \mathcal{I} \models \varphi\}\\ \alpha^{\mathcal{I}}_-(A) &:= \{b^{\mathcal{I}} \mid \varphi/\neg A(b) \in \mathsf{post} \wedge \mathcal{I} \models \varphi\}\\ D^{\mathcal{I}}_\alpha(A) &:= (\Delta^{\mathcal{I}} \setminus \{b^{\mathcal{I}} \mid A(b) \in \mathsf{occ}\}) \cup (\alpha^{\mathcal{I}}_+(A) \cup \alpha^{\mathcal{I}}_-(A))\\ \alpha^{\mathcal{I}}_+(r) &:= \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/r(a,b) \in \mathsf{post} \wedge \mathcal{I} \models \varphi\}\\ \alpha^{\mathcal{I}}_-(r) &:= \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/\neg r(a,b) \in \mathsf{post} \wedge \mathcal{I} \models \varphi\}\\ D^{\mathcal{I}}_\alpha(r) &:= ((\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid r(a,b) \in \mathsf{occ}\}) \cup (\alpha^{\mathcal{I}}_+(r) \cup \alpha^{\mathcal{I}}_-(r))\end{aligned}$$

The transition relation on interpretations should ensure that $\alpha^{\mathcal{I}}_+(A) \subseteq A^{\mathcal{J}}$ and $\alpha^{\mathcal{I}}_-(A) \cap A^{\mathcal{J}} = \emptyset$ if $\mathcal{J}$ is the result of applying α in $\mathcal{I}$. It should also ensure that nothing else changes, with the possible exception of the occluded literals. Intuitively, $D^{\mathcal{I}}_\alpha(A)$ and $D^{\mathcal{I}}_\alpha(r)$ describe those parts of the model that are *not* exempted from this restriction by the presence of an occlusion. Since we restrict our attention to acyclic TBoxes, for which the interpretation of defined concepts is uniquely determined by the interpretation of primitive concepts and role names, it is not necessary to consider defined concepts when defining the transition relation.

Definition 3.1.4. Let $\mathcal{T}$ be an acyclic TBox, $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ an action for $\mathcal{T}$, and $\mathcal{I}, \mathcal{I}'$ models of $\mathcal{T}$ sharing the same domain and interpretation of all individual names and respecting UNA on individual names. We say that α *may transform* $\mathcal{I}$ *to* $\mathcal{I}'$ *w.r.t.* $\mathcal{T}$ ($\mathcal{I} \Rightarrow^{\mathcal{T}}_\alpha \mathcal{I}'$) iff, for each primitive concept A and role name r, we have

$$\begin{array}{rcl} \alpha^{\mathcal{I}}_+(A) \cap \alpha^{\mathcal{I}}_-(A) = \emptyset \quad \text{and} \quad \alpha^{\mathcal{I}}_+(r) \cap \alpha^{\mathcal{I}}_-(r) = \emptyset \\ A^{\mathcal{I}'} \cap D^{\mathcal{I}}_\alpha(A) & = & ((A^{\mathcal{I}} \cup \alpha^{\mathcal{I}}_+(A)) \setminus \alpha^{\mathcal{I}}_-(A)) \cap D^{\mathcal{I}}_\alpha(A) \\ r^{\mathcal{I}'} \cap D^{\mathcal{I}}_\alpha(r) & = & ((r^{\mathcal{I}} \cup \alpha^{\mathcal{I}}_+(r)) \setminus \alpha^{\mathcal{I}}_-(r)) \cap D^{\mathcal{I}}_\alpha(r). \end{array}$$

The composite action $\alpha_1, \ldots, \alpha_k$ *may transform* $\mathcal{I}$ to $\mathcal{I}'$ ($\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_k} \mathcal{I}'$) iff there are models $\mathcal{I}_0, \ldots, \mathcal{I}_k$ of $\mathcal{T}$ with $\mathcal{I} = \mathcal{I}_0$, $\mathcal{I}' = \mathcal{I}_k$, and $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_i$ for $1 \leq i \leq k$. △

One remark is in order. By definition of $\mathcal{I} \Rightarrow^{\mathcal{T}}_\alpha \mathcal{I}'$, interpretations $\mathcal{I}$ and $\mathcal{I}'$ are supposed to respect UNA on individual names. As already discussed in Section 2.1, since the set of individual names $\mathsf{N_I}$ is countably infinite, this implies that all interpretations $\mathcal{I}$ and $\mathcal{I}'$ related by $\Rightarrow^{\mathcal{T}}_\alpha$ have infinite domains. However, in the coming chapters, in order to be able to work with finite models, we may "relax" the requirement that interpretations interpret all individuals from $\mathsf{N_I}$, and assume that they interpret only individual names from a finite set Ind and respect UNA on this set.

Note that acyclic TBoxes in $\mathfrak{A}_1$ do not correspond to state constraints in the standard sense. By semantics of $\mathfrak{A}_1$-actions, TBoxes do not constrain the initial and successor interpretations, except for specifying how defined concept names depend on primitive concept and role names. In this light, acyclic TBoxes can be used to capture acyclic causality relations between concept names, similar to those from [Lin95, Thi97]. In $\mathfrak{A}_1$, only primitive concept names can appear in consequences of action post-conditions and hence their interpretations may change as a direct effect of action execution, while interpretations of defined concept names change as a ramification of action application.

Because of our restriction to acyclic TBoxes and primitive literals in the consequence part of post-conditions, actions without occlusions are *deterministic*, i.e., for any model $\mathcal{I}$ of $\mathcal{T}$ there exists at most one model $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow^{\mathcal{T}}_\alpha \mathcal{I}'$. First note that there are indeed cases where there is no successor model $\mathcal{I}'$. In this case, we say that the action is *inconsistent with* $\mathcal{I}$. It is easy to see that this is the case iff there are post-conditions $\varphi_1/\psi, \varphi_2/\neg\psi \in \mathsf{post}$ such that both φ_1 and φ_2 are satisfied in $\mathcal{I}$. Second, assume that α is consistent with $\mathcal{I}$. The fact that there is exactly one model $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow^{\mathcal{T}}_\alpha \mathcal{I}'$ is an easy consequence of the next lemma, whose proof we leave as an easy exercise.

Lemma 3.1.5. *Let $\mathcal{T}$ be an acyclic TBox, $\alpha = (\mathsf{pre}, \emptyset, \mathsf{post})$ an atomic action for $\mathcal{T}$, and $\mathcal{I} \Rightarrow^{\mathcal{T}}_\alpha \mathcal{I}'$ for models $\mathcal{I}, \mathcal{I}'$ of $\mathcal{T}$. If A is a primitive concept and r a role name, then*

$$A^{\mathcal{I}'} := \big(A^{\mathcal{I}} \cup \{b^{\mathcal{I}} \mid \varphi/A(b) \in \mathsf{post} \text{ and } \mathcal{I} \models \varphi\}\big) \setminus \{b^{\mathcal{I}} \mid \varphi/\neg A(b) \in \mathsf{post} \text{ and } \mathcal{I} \models \varphi\},$$

$$r^{\mathcal{I}'} := \big(r^{\mathcal{I}} \cup \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/r(a,b) \in \mathsf{post} \text{ and } \mathcal{I} \models \varphi\}\big) \setminus \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/\neg r(a,b) \in \mathsf{post} \text{ and } \mathcal{I} \models \varphi\}.$$

Since the interpretation of the defined concepts is uniquely determined by the interpretation of the primitive concepts and the role names, it follows that there cannot exist more than one $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow_\alpha^\mathcal{T} \mathcal{I}'$.

3.1.2 Reasoning about Actions

Assume that we want to apply a composite action $\alpha_1, \ldots, \alpha_k$ for the acyclic TBox $\mathcal{T}$. Usually, we do not have complete information about the world (i.e., the model $\mathcal{I}$ of $\mathcal{T}$ is not known completely). All we know are some facts about this world, i.e., we have an ABox $\mathcal{A}$, and all models of $\mathcal{A}$ together with $\mathcal{T}$ are considered to be possible states of the world.

Before trying to apply the action, we want to know whether it is indeed executable, i.e., whether all necessary pre-conditions are satisfied. If the action is executable, we may want to know whether applying it achieves the desired effect, i.e., whether an assertion that we want to make true really holds after executing the action. These problems are basic inference problems considered in the reasoning about action community, see e.g. [Rei01] (and Section 2.2). In our setting, they can formally be defined as follows:

Definition 3.1.6 (Reasoning Problems). Let $\mathcal{T}$ be an acyclic TBox, $\alpha_1, \ldots, \alpha_k$ an action for $\mathcal{T}$ with $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$, and $\mathcal{A}$ an ABox.

- *Executability:* $\alpha_1, \ldots, \alpha_k$ is *executable in $\mathcal{A}$ w.r.t. $\mathcal{T}$* iff the following condition is true for all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$:
  - $\mathcal{I} \models \mathsf{pre}_1$
  - for all i with $1 \leq i < k$ and all interpretations $\mathcal{I}'$ with $\mathcal{I} \Rightarrow_{\alpha_1,\ldots,\alpha_i}^\mathcal{T} \mathcal{I}'$, we have $\mathcal{I}' \models \mathsf{pre}_{i+1}$.
- *Projection:* an assertion φ is a *consequence of applying $\alpha_1, \ldots, \alpha_k$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$* (written $\mathcal{T}, \mathcal{A}^{\alpha_1,\ldots,\alpha_k} \models \varphi$ [1]) iff, for all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$, and all $\mathcal{I}'$ with $\mathcal{I} \Rightarrow_{\alpha_1,\ldots,\alpha_k}^\mathcal{T} \mathcal{I}'$, we have $\mathcal{I}' \models \varphi$.

  We may refer to the assertion φ as a *query*.

If $\mathcal{T}$ is empty, we simply drop the phrase "w.r.t. $\mathcal{T}$" instead of writing "w.r.t. the empty TBox $\emptyset$". △

Note that executability alone does not guarantee that we cannot get stuck while executing a composite action. It may also happen that the action to be applied is inconsistent with the current interpretation. This cannot happen if we additionally know that all actions α_i are *consistent with* $\mathcal{T}$ in the following sense: α_i is not inconsistent with any model $\mathcal{I}$ of $\mathcal{T}$. Summing up, to achieve an effect φ (an ABox assertion) starting from a world description $\mathcal{A}$ and given a TBox $\mathcal{T}$, we need an action $\alpha_1, \ldots, \alpha_k$ such that $\alpha_1, \ldots, \alpha_k$ is executable in $\mathcal{A}$ w.r.t $\mathcal{T}$, α_i is consistent with $\mathcal{T}$ for $1 \leq i \leq k$, and φ is a consequence of applying $\alpha_1, \ldots, \alpha_k$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$.

We do not view consistency with the considered acyclic TBox $\mathcal{T}$ as a reasoning task[2], but rather as a condition that we generally expect to be satisfied by all well-formed actions.

[1] Analogously, we will write $\mathcal{T}, \mathcal{A}^{\alpha_1,\ldots,\alpha_k} \not\models \varphi$ to denote that φ is *not* a consequence of applying $\alpha_1, \ldots, \alpha_k$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$.

[2] Note that in the action formalism $\mathfrak{A}_2$ (to be defined in Chapter 5), which is designed to handle general TBoxes, consistency becomes a real reasoning task; see Section 5.1.2.

Still, we should be able to decide whether an action is consistent with a TBox. This can be done by a reduction to standard DL reasoning: given the characterization of consistency *with a model* stated above Lemma 3.1.5, it is not difficult to see that an atomic action α with post-conditions post is consistent with a TBox $\mathcal{T}$ iff $\{\varphi_1/\psi, \varphi_2/\neg\psi\} \subseteq \mathsf{post}$ implies that the ABox $\{\varphi_1, \varphi_2\}$ is inconsistent w.r.t. $\mathcal{T}$.

In Example 3.1.3, both actions are consistent with $\mathcal{T}$. Given the ABox

$$\mathcal{A} = \{\mathsf{parent}(a, c), \mathsf{permanent\_resident}(a, \mathsf{UK}), \exists \mathsf{holds.Electricity\_contract}(a)\},$$

the composite action $\alpha = \alpha_1, \alpha_2$ is executable, and $\mathsf{receives\_c\_benef\_for}(a, c)$ is a consequence of applying α_1, α_2 in $\mathcal{A}$ w.r.t. $\mathcal{T}$. Note that the presence of the TBox is crucial for this result.

The main aim of this chapter is to show how the two reasoning tasks executability and projection can be decided in the description logics between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$, and how their complexity depends on the description logic used within our framework. There is one particularly simple case: for atomic actions α, deciding executability boils down to standard DL reasoning: α is executable in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff $\mathcal{T}, \mathcal{A} \models \varphi$ for all $\varphi \in \mathsf{pre}$. Executability for composite actions is less trivial, and the same holds for projection of both atomic and composite action. We show now that the two reasoning problems can be polynomially reduced to each other. This allows us to concentrate on projection when proving decidability and complexity results.

Lemma 3.1.7. *Executability and projection can be reduced in polynomial time to each other.*

Proof. Let $\alpha_1, \ldots, \alpha_k$ with $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ be a composite action for the acyclic TBox $\mathcal{T}$. This action is executable in the ABox $\mathcal{A}$ iff

(i) pre_1 is satisfied in every model of $\mathcal{A}$ and $\mathcal{T}$ and, for $1 \leq i < k$,

(ii) all assertions in pre_{i+1} are consequences of applying $\alpha_1, \ldots, \alpha_i$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$.

Condition (ii) is obviously a projection problem. Condition (i) can also be seen as a projection problem for the empty action $(\emptyset, \emptyset, \emptyset)$.

Conversely, assume that we want to know whether φ is a consequence of applying $\alpha_1, \ldots, \alpha_k$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$. We consider the composite action $\alpha'_1, \ldots, \alpha'_k, \alpha'$, where $\alpha'_i = (\emptyset, \mathsf{occ}_i, \mathsf{post}_i)$ for $1 \leq i \leq k$, and $\alpha' = (\{\varphi\}, \emptyset, \emptyset)$. Then φ is a consequence of applying $\alpha_1, \ldots, \alpha_k$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff $\alpha'_1, \ldots, \alpha'_k, \alpha'$ is executable. ❑

It is interesting to note that the reduction of projection of a composite action $\alpha = \alpha_1, \ldots, \alpha_k$ is to executability for actions of length $k + 1$. Indeed, we shall later see that, for some description logics, projection of atomic actions is computationally harder than executability of atomic actions.

3.1.3 Relation to Situation Calculus

We compare the action formalism $\mathfrak{A}_1$ with one of the most prominent (families of) formalisms for reasoning about actions, the *Situation Calculus* [Rei01]. The preliminaries of the Situation Calculus are given in Section 2.2. We show how to translate the components of our formalism, i.e., ABoxes, TBoxes, and actions, into the Situation Calculus. Based on this translation, we then establish a correspondence between the reasoning problems. This correspondence shows

that the consequences of an action application computed in our framework are identical to the consequences that would be computed in the Situation Calculus. In particular, this means that our solution of the frame problem is identical to Reiter's as initially proposed in [Rei91].

Since it was shown in [Thi99, ST06] that standard domain specifications in the Situation Calculus and the Fluent Calculus are mutually translatable, we conclude that our formalisms can also be embedded into the Fluent Calculus. This was also explicitly shown in [DT07].

Actions of the Situation Calculus correspond to our atomic actions, while situations can be viewed as first-order structures and roughly correspond to interpretations in our framework. In its most common form, as described in Section 2.2, the Situation Calculus is restricted to *deterministic* effects of actions. Therefore, in this section we restrict ourselves to deterministic actions, i.e., to actions without occlusions—c.f. Lemma 3.1.5.

The foundation for translating the action formalism $\mathfrak{A}_1$ into the Situation Calculus is provided by the standard translation of description logics into first order logic [BCM$^+$03, Bor96]. For our purposes, this translation needs to be slightly modified: in the standard translation, concept names correspond to unary predicates, concepts correspond to first-order formulae with one free variable, and role names correspond to binary predicates. In the Situation Calculus, all concept names and role names correspond to *fluents* since their extension depends on the actual situation. Thus, we need to extend the predicates corresponding to concept names and role names by one additional argument of type situation.

We now describe the modified translation for $\mathcal{ALCQIO}$-concepts. The translation is based on two recursive mappings $\pi_{x,s}(\cdot)$ and $\pi_{y,s}(\cdot)$, which translate $\mathcal{ALCQIO}$-concepts into a formula with one free object variable x or y and one free situation variable s. For each concept name A, we introduce a binary predicate of the same name with one place for objects and one for situations. For each role name r, we introduce a ternary predicate of the same name with two places for objects and one for situations. And for each individual name a, we introduce an object-typed constant $\mathbf{a}$. The mapping $\pi_{x,s}$ is defined as follows, and the mapping $\pi_{y,s}$ is defined like $\pi_{x,s}$ with the roles of x and y swapped:

$$\begin{aligned}
\pi_{x,s}(A) &= A(x,s), && \text{for concept names } A \in \mathsf{N_C}\\
\pi_{x,s}(\{a\}) &= (x = \mathbf{a}), && \text{for individuals } a \in \mathsf{N_I}\\
\pi_{x,s}(\neg D) &= \neg\pi_{x,s}(D),\\
\pi_{x,s}(C \sqcap D) &= \pi_{x,s}(C) \wedge \pi_{x,s}(D),\\
\pi_{x,s}(C \sqcup D) &= \pi_{x,s}(C) \vee \pi_{x,s}(D),\\
\pi_{x,s}(\bowtie n\, r\, C) &= \exists^{\bowtie n} y.r(x,y,s) \wedge \pi_{y,s}(C),\\
\pi_{x,s}(\bowtie n\, r^-\, C) &= \exists^{\bowtie n} y.r(y,x,s) \wedge \pi_{y,s}(C).
\end{aligned}$$

where $\bowtie \in \{\geq, \leq\}$.

The next step is to translate ABoxes into first-order logic. This is easily achieved using the mapping $\pi_{x,s}$ that we have just introduced. In the following, $\varphi[x/\mathbf{a}]$ denotes the result of replacing each free occurrence of x in φ with the object constant $\mathbf{a}$:

$$\pi_s(\mathcal{A}) = \bigwedge_{C(b)\in\mathcal{A}} \pi_{x,s}(C)[x/\mathbf{b}] \wedge \bigwedge_{r(b,c)\in\mathcal{A}} r(\mathbf{b},\mathbf{c},s) \wedge \bigwedge_{\neg r(\mathbf{b},\mathbf{c})\in\mathcal{A}} \neg r(\mathbf{b},\mathbf{c},s)$$

To translate an $\mathfrak{A}_1$-action into a description of an action in Situation Calculus form, we need to translate the pre-conditions into action pre-condition axioms and the post-conditions into

successor state axioms. We begin with the former. For each action $\alpha = (\mathsf{pre}, \emptyset, \mathsf{post})$, we introduce an action-typed constant $\mathbf{u}$ and define an action pre-condition axiom

$$\mathsf{Poss}(\mathbf{u}, s) \equiv \pi_s(\mathsf{pre}),$$

which specifies whether it is possible to carry out the action $\mathbf{u}$ in a situation s.[3]

To define successor state axioms, we fix a finite set of actions $\alpha_1, \ldots, \alpha_n$ with $\alpha_i = (\mathsf{pre}_i, \emptyset, \mathsf{post}_i)$ and associated action constant $\mathbf{u}_i$, for $1 \leq i \leq n$. Then, for each concept name A and each role name r, we introduce successor state axioms as follows, where u denotes an action-typed variable

$$A(x, \mathsf{do}(u, s)) \equiv \Phi_A(x, u, s) \quad \text{and} \quad r(x, y, \mathsf{do}(u, s)) \equiv \Phi_r(x, y, u, s)$$

with $\Phi_A(x, u, s)$ and $\Phi_r(x, y, u, s)$ defined as follows:

$$\begin{aligned}\Phi_A(x, u, s) := & \bigvee_{\{(\varphi,b,i) \mid \varphi/A(b) \in \mathsf{post}_i\}} \big(\pi_s(\{\varphi\}) \wedge x = \mathbf{b} \wedge u = \mathbf{u}_i\big) \vee \\ & A(x, s) \wedge \neg \bigvee_{\{(\varphi,b,i) \mid \varphi/\neg A(b) \in \mathsf{post}_i\}} \big(\pi_s(\{\varphi\}) \wedge x = \mathbf{b} \wedge u = \mathbf{u}_i\big)\end{aligned}$$

$$\begin{aligned}\Phi_r(x, y, u, s) := & \bigvee_{\{(\varphi,b,c,i) \mid \varphi/r(b,c) \in \mathsf{post}_i\}} \big(\pi_s(\{\varphi\}) \wedge x = \mathbf{b} \wedge y = \mathbf{c} \wedge u = \mathbf{u}_i\big) \vee \\ & r(x, y, s) \wedge \neg \bigvee_{\{(\varphi,b,c,i) \mid \varphi/\neg r(b,c) \in \mathsf{post}_i\}} \big(\pi_s(\{\varphi\}) \wedge x = \mathbf{b} \wedge y = \mathbf{c} \wedge u = \mathbf{u}_i\big)\end{aligned}$$

It is easily seen that the syntactic form of the formulas Φ_A and Φ_r is as required for successor state axioms in the Situation Calculus, i.e. they are *uniform* in the situation variable s. Reiter identifies a special form of successor state axioms, so-called *context-free* ones (c.f. Section 2.2), for which there exists a particularly simple algorithm for regression, the basic computational mechanism of the Situation Calculus. It is interesting to note that our successor state axioms are context-free iff only unconditional post-conditions are used in actions.

The only component of our formalism for reasoning about actions that we have not yet translated into a Situation Calculus form are TBoxes. Indeed, there is no need to translate them, which can be seen as follows. Since we assume TBoxes to by acyclic, we may completely eliminate TBoxes using a process called *unfolding*: first, exhaustively replace each defined concept name appearing on the right-hand side of a concept definition in the TBox $\mathcal{T}$ with its defining concept description as given by $A \doteq C \in \mathcal{T}$. Second, replace all defined concept names in the ABox and action descriptions by their defining concept descriptions and drop the TBox [BCM$^+$03]. This unfolding process preserves executability and consequences. For example, if $\mathcal{A}'$, α', and φ' are the result of unfolding a TBox $\mathcal{T}$ given the ABox $\mathcal{A}$, the atomic action α, and the assertion φ, then φ is a consequence of applying α in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff φ' is a consequence of applying α' in $\mathcal{A}'$. Note that the unfolding of TBoxes may lead to an exponential blowup in the size of ABox and actions. For our purposes, however, this is irrelevant: we only carry out the translation to compare reasoning in the two formalisms, and not to actually use it for practical reasoning.

[3] We use $\mathbf{u}$ for action constants and u for action variables instead of the more common $\mathbf{a}$ and a to avoid confusion with individual names and corresponding object constants.

Now that all components of our framework have been translated to counterparts in the Situation Calculus, we show how an ABox $\mathcal{A}$ and a set of atomic actions $\alpha_1, \ldots, \alpha_n$ can be translated into a *basic action theory* as defined in Section 2.2 (and [Rei01]), where all free variables are assumed to be universally quantified:

- Σ is the set of the four *foundational axioms* for situations, as defined in Section 2.2.
- $\mathcal{D}_{ss}$ is the set of *successor state axioms*, one for each concept and role name occurring in $\mathcal{A}$ as defined above,
- $\mathcal{D}_{ap}$ is the set of *action pre-condition axioms*, one for each action-typed constant $\mathbf{u}_1, \ldots, \mathbf{u}_n$ corresponding to the actions $\alpha_1, \ldots, \alpha_n$ as defined above,
- $\mathcal{D}_{una}$ is the set of *unique name axioms* for actions:

$$\bigwedge_{1 \leq i < j \leq n} \mathbf{u}_i \neq \mathbf{u}_j$$

- $\mathcal{D}_{\mathbf{s_0}} := \pi_{\mathbf{s_0}}(\mathcal{A})$ is the description of the initial situation.

We use $\mathcal{D}(\mathcal{A}, \alpha_1, \ldots, \alpha_n)$ to denote the basic action theory obtained from $\mathcal{A}$ and $\alpha_1, \ldots, \alpha_n$, i.e., $\Sigma \cup \mathcal{D}_{\mathbf{s_0}} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una}$.

Finally, we compare our framework with reasoning in the Situation Calculus. We formulate the main theorem of this section stating that reasoning in our formalism coincides with reasoning in the Situation Calculus.

Theorem 3.1.8. *Let $\mathcal{A}$ be an ABox, $\alpha = \alpha_1, \ldots, \alpha_n$ a composite action, and φ an assertion. Then*

1. *α is executable in $\mathcal{A}$ iff the sequence $\mathbf{u}_1, \ldots, \mathbf{u}_n$ is executable in $\mathbf{s_0}$ w.r.t. the basic action theory $\mathcal{D}(\mathcal{A}, \alpha_1, \ldots, \alpha_n)$.*
2. *φ is a consequence of applying α in $\mathcal{A}$ iff $\pi_s(\{\varphi\})$ is a consequence of applying the sequence $\mathbf{u}_1, \ldots, \mathbf{u}_n$ in $\mathbf{s_0}$ w.r.t. the basic action theory $\mathcal{D}(\mathcal{A}, \alpha_1, \ldots, \alpha_n)$.*

Thus, the framework for reasoning about actions presented in this chapter is fully compatible not only with ontology languages based on description logics, but also with the Situation Calculus.

We would like to note that there is a more explicit way for dealing with TBoxes than unfolding: TBoxes can be translated into state constraints of the Situation Calculus:

$$\pi_s(\mathcal{T}) \quad = \quad \bigwedge_{A \doteq C \in \mathcal{T}} \forall x . \pi_{x,s}(A) \leftrightarrow \pi_{x,s}(C).$$

To obtain an analogue of Theorem 3.1.8, we can then devise successor state axioms only for primitive concepts and role names, but not for defined concepts—although the latter are fluents. This corresponds to not minimizing defined concepts in Definition 3.1.4. Note that, if we admitted also cyclic TBoxes, then the unfolding approach could not be used any more and we would be forced to translate TBoxes into state constraints. This would pose semantic problems as discussed in more detail in Section 4.3, in Appendix B of [Rei01], and in [Lif90].

3.2 Deciding Executability and Projection

The purpose of this section is to develop reasoning procedures for the reasoning problems introduced in Section 3.1.2, and to analyze the computational complexity of executability and projection in the description logics between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$. Throughout this section, we assume that all actions are consistent with their TBox, and that TBoxes are acyclic.

By Lemma 3.1.7, we can restrict the attention to the projection problem. Basically, we solve this problem by an approach that is similar to the regression operation used in the Situation Calculus approach [Rei01]. The main idea is to reduce projection, which considers sequences of interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_k$ obtained by action application, to standard reasoning tasks for a single interpretation $\mathcal{I}$.

We show that the theory we obtain can again be expressed by a description logic TBox and ABox. This way, projection is reduced to ABox consequence in DL, from which we obtain decidability results and upper complexity bounds. Interestingly, when taking this approach, we cannot always stay within the DL we started with since we need to introduce nominals in the reduction. We prove lower complexity bounds for projection showing that the increase in complexity that is sometimes obtained by introducing nominals cannot be avoided.

The following results are proved in this section:

Theorem 3.2.1. *Executability and projection of composite $\mathfrak{A}_1$-actions w.r.t. acyclic TBoxes are*

1. PSPACE*-complete for $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCQ}$, and $\mathcal{ALCQO}$;*
2. EXPTIME*-complete for $\mathcal{ALCI}$ and $\mathcal{ALCIO}$;*
3. *co*-NEXPTIME*-complete for $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$.*

Points 1 and 3 hold regardless of whether numbers in number restrictions are coded in unary or binary. Thus, in all cases considered, the complexity of executability and projection for a description logic $\mathcal{L}$ coincides with the complexity of ABox consequence in $\mathcal{LO}$, the extension of $\mathcal{L}$ with nominals.

3.2.1 Reduction to DL Reasoning

We reduce projection in fragments $\mathcal{L}$ of $\mathcal{ALCQIO}$ to ABox consequence in the extension $\mathcal{LO}$ of $\mathcal{L}$ with nominals.

Theorem 3.2.2. *Let $\mathcal{L}$ be a DL from the set $\{\mathcal{ALC}, \mathcal{ALCI}, \mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQ}, \mathcal{ALCQO}, \mathcal{ALCQI}, \mathcal{ALCQIO}\}$. Then projection of composite actions formulated in $\mathcal{L}$ can be polynomially reduced to ABox consequence in $\mathcal{LO}$ w.r.t. acyclic TBoxes under UNA.*

Let $\mathcal{L}$ be one of the languages listed in Theorem 3.2.2, and let $\mathcal{A}$ be an ABox, $\alpha_1, \ldots, \alpha_n$ a composite action with $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$, $\mathcal{T}$ an acyclic TBox, and φ_0 an assertion, all formulated in $\mathcal{L}$. We are interested in deciding whether φ_0 is a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. Without loss of generality, we assume that φ_0 is of the form $A_0(a_0)$, for a concept name A_0:

1. Assertions $r(a, b)$ and $\neg r(a, b)$ can be replaced with $(\exists r.\{b\})(a)$ and $(\forall r.\neg\{b\})(a)$, respectively. This presupposes nominals, but nominals will be used in our reduction, anyway.

2. If $\varphi = C(a)$ with C not a concept name, we add a concept definition $A_0 \doteq C$ to the TBox $\mathcal{T}$, and then consider $\varphi = A_0(a)$.

In the following, we call $\mathcal{A}$, $\mathcal{T}$, $\alpha_1, \ldots, \alpha_n$, and φ_0 *the input*. We devise a reduction ABox $\mathcal{A}_{\mathsf{red}}$, an (acyclic) reduction TBox $\mathcal{T}_{\mathsf{red}}$, and a reduction assertion φ_{red} such that

φ_0 is a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff $\mathcal{A}_{\mathsf{red}}, \mathcal{T}_{\mathsf{red}} \models \varphi_{\mathsf{red}}$.

The main idea of the reduction is to define $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ such that each *single* model of them encodes a *sequence* of interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ obtained by applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}$ (and *all* such sequences are encoded by reduction models). To ensure this, we use the following intuitions:

- The reduction ABox states that (i) the "$\mathcal{I}_0$-part" of a reduction model $\mathcal{I}$ is a model of $\mathcal{A}$, and that (ii) the $\mathcal{I}_i$-part of $\mathcal{I}$ satisfies the post-conditions post_i, for $1 \leq i \leq n$.
- The reduction TBox states that each $\mathcal{I}_i$ part of $\mathcal{I}$ is a model of $\mathcal{T}$, for $i \leq n$.
- We need to describe the law of inertia, i.e., the fact that we want to minimize the changes that are performed when applying an action. This task is split among the reduction ABox and TBox.

To understand the splitting mentioned in the third item, it is important to distinguish two kinds of elements in interpretations: we call an element $d \in \Delta^{\mathcal{I}}$ *named* if $a^{\mathcal{I}} = d$ for some individual a used in the input, and *unnamed* otherwise. Intuitively, the minimization of changes on named elements can be described in a direct way through the ABox $\mathcal{A}_{\mathsf{red}}$, while the minimization of changes on unnamed elements is achieved through a suitable encoding of $\mathcal{T}$ in $\mathcal{T}_{\mathsf{red}}$. Indeed, minimizing changes on unnamed elements boils down to enforcing that changes in concept (non)membership and role (non)membership involving (at least) one unnamed domain element *never* occur: due to the restriction to primitive concept names in post-conditions, our actions are not expressive enough to enforce such changes.

In the reduction, we use the following concept names, role names, and individual names:

- The smallest set that contains all concepts appearing in the input and is closed under taking subconcepts is denoted with Sub. For every $C \in \mathsf{Sub}$ and every $i \leq n$, we introduce a concept name $T_C^{(i)}$. It will be ensured by the TBox $\mathcal{T}_{\mathsf{red}}$ that the concept name $T_C^{(i)}$ stands for the interpretation of C in the i-th interpretation.
- We use a concept name $A^{(i)}$ for every primitive concept name A used in the input and every $i \leq n$. Intuitively, $A^{(i)}$ represents the interpretation of the concept name A in the i-th interpretation, *but only with respect to the named domain elements*. Since concept membership of unnamed elements never changes, the "unnamed part" of the interpretation of the concept name A can always be found in $A^{(0)}$.
- We use a role name $r^{(i)}$ for every role name r used in the input and every $i \leq n$. Similarly to concept names, $r^{(i)}$ stands for the interpretation of r in the i-th interpretation, *but only records role relationships where both involved domain elements are named.*
- We use a concept name N that will be used to denote "named elements" of interpretations.

- The set of individual names used in the input is denoted with Ind. For every $a \in \mathsf{Ind}$, we introduce an auxiliary role name r_a.
- An auxiliary individual name $a_{\mathsf{help}} \notin \mathsf{Ind}$.

The reduction TBox $\mathcal{T}_{\mathsf{red}}$ consists of several components. The first component simply states that N denotes exactly the named domain elements:

$$\mathcal{T}_N := \Big\{ N \doteq \bigsqcup_{a \in \mathsf{Ind}} \{a\} \Big\}.$$

The second component $\mathcal{T}_{\mathsf{sub}}$ contains one concept definition for every $i \leq n$ and every concept $C \in \mathsf{Sub}$ that is not a defined concept name in $\mathcal{T}$. These concept definitions ensure that $T_C^{(i)}$ stands for the interpretation of C in the i-th interpretation as desired:

$$T_A^{(i)} \doteq (N \sqcap A^{(i)}) \sqcup (\neg N \sqcap A^{(0)}) \quad \text{if } A \text{ primitive in } \mathcal{T} \tag{a}$$

$$T_{\{a\}}^{(i)} \doteq \{a\} \tag{b}$$

$$T_{\neg C}^{(i)} \doteq \neg T_C^{(i)} \tag{c}$$

$$T_{C \sqcap D}^{(i)} \doteq T_C^{(i)} \sqcap T_D^{(i)} \tag{d}$$

$$T_{C \sqcup D}^{(i)} \doteq T_C^{(i)} \sqcup T_D^{(i)} \tag{e}$$

$$T_{(\geqslant m\ r\ C)}^{(i)} \doteq \Big(N \sqcap \bigsqcup_{0 \leq j \leq min\{m, |\mathsf{Ind}|\}} \big((\geqslant j\ r^{(i)}\ (N \sqcap T_C^{(i)})) \sqcap (\geqslant (m-j)\ r^{(0)}\ (\neg N \sqcap T_C^{(i)})) \big) \Big) \tag{f}$$
$$\sqcup \Big(\neg N \sqcap (\geqslant m\ r^{(0)}\ T_C^{(i)}) \Big)$$

$$T_{(\leqslant m\ r\ C)}^{(i)} \doteq \Big(N \sqcap \bigsqcup_{0 \leq j \leq min\{m, |\mathsf{Ind}|\}} \big((\leqslant j\ r^{(i)}\ (N \sqcap T_C^{(i)})) \sqcap (\leqslant (m-j)\ r^{(0)}\ (\neg N \sqcap T_C^{(i)})) \big) \Big) \tag{g}$$
$$\sqcup \Big(\neg N \sqcap (\leqslant m\ r^{(0)}\ T_C^{(i)}) \Big)$$

where $r^{-(i)}$ denotes $(r^{(i)})^-$ in the concept definitions for number restrictions. Line (a) reflects the fact that concept names $A^{(i)}$ only represent the extension of A in the i-th interpretation *for named domain elements*. To get $T_A^{(i)}$, the full extension of A in the i-th interpretation, we use $A^{(i)}$ for named elements and $A^{(0)}$ for unnamed ones. A similar splitting of role relationships into a named part and an unnamed part is reflected in the translation of number restrictions given in the last two lines.

Now we can assemble the reduction TBox $\mathcal{T}_{\mathsf{red}}$:

$$\mathcal{T}_{\mathsf{red}} \quad := \quad \mathcal{T}_{\mathsf{sub}} \cup \mathcal{T}_N \ \cup \{ T_A^{(i)} \doteq T_E^{(i)} \mid A \doteq E \in \mathcal{T},\ i \leq n \}$$

The last summand of $\mathcal{T}_{\mathsf{red}}$ ensures that all definitions from the input TBox $\mathcal{T}$ are satisfied by all interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$.

The reduction ABox $\mathcal{A}_{\mathsf{red}}$ also consists of several components. The first component ensures that, for each individual a occurring in the input, the auxiliary role r_a connects each individual (including a_{help}) with a, and only with a. This construction will simplify the definition of the other components of $\mathcal{A}_{\mathsf{red}}$:

$$\mathcal{A}_{\mathsf{aux}} := \big\{ a : \big(\exists r_b.\{b\} \sqcap \forall r_b.\{b\} \big) \mid a \in \mathsf{Ind} \cup \{a_{\mathsf{help}}\},\ b \in \mathsf{Ind} \big\}.$$

To continue, we first introduce the following abbreviations, for $i \leq n$:

$$\begin{aligned} \mathsf{p}_i(C(a)) &:= \forall r_a.T_C^{(i)} \\ \mathsf{p}_i(r(a,b)) &:= \forall r_a.\exists r^{(i)}.\{b\} \\ \mathsf{p}_i(\neg r(a,b)) &:= \forall r_a.\forall r^{(i)}.\neg\{b\}. \end{aligned}$$

The next component of $\mathcal{A}_{\mathsf{red}}$ formalizes satisfaction of the post-conditions. Note that its formulation relies on $\mathcal{A}_{\mathsf{aux}}$. For $1 \leq i \leq n$, we define

$$\mathcal{A}^{(i)}_{\mathsf{post}} := \big\{a_{\mathsf{help}} : \big(\mathsf{p}_{i-1}(\varphi) \rightarrow \mathsf{p}_i(\psi)\big) \mid \varphi/\psi \in \mathsf{post}_i\big\}.$$

The following component formalizes the minimization of changes on named elements. For $1 \leq i \leq n$ the ABox $\mathcal{A}^{(i)}_{\mathsf{min}}$ contains

1. the following assertions for every $a \in \mathsf{Ind}$ and every primitive concept name A with $A(a) \notin \mathsf{occ}_i$:

$$\begin{aligned} a &: \Big(\big(A^{(i-1)} \sqcap \bigsqcap_{\varphi/\neg A(a) \in \mathsf{post}_i} \neg\mathsf{p}_{i-1}(\varphi)\big) \rightarrow A^{(i)}\Big) \\ a &: \Big(\big(\neg A^{(i-1)} \sqcap \bigsqcap_{\varphi/A(a) \in \mathsf{post}_i} \neg\mathsf{p}_{i-1}(\varphi)\big) \rightarrow \neg A^{(i)}\Big); \end{aligned}$$

2. the following assertions for all $a, b \in \mathsf{Ind}$ and every role name r with $r(a,b) \notin \mathsf{occ}_i$:

$$\begin{aligned} a &: \Big(\big(\exists r^{(i-1)}.\{b\} \sqcap \bigsqcap_{\varphi/\neg r(a,b) \in \mathsf{post}_i} \neg\mathsf{p}_{i-1}(\varphi)\big) \rightarrow \exists r^{(i)}.\{b\}\Big) \\ a &: \Big(\big(\forall r^{(i-1)}.\neg\{b\} \sqcap \bigsqcap_{\varphi/r(a,b) \in \mathsf{post}_i} \neg\mathsf{p}_{i-1}(\varphi)\big) \rightarrow \forall r^{(i)}.\neg\{b\}\Big). \end{aligned}$$

The ABox $\mathcal{A}_{\mathsf{ini}}$ ensures that the first interpretation of the encoded sequence is a model of the input ABox $\mathcal{A}$:

$$\begin{aligned} \mathcal{A}_{\mathsf{ini}} := \; & \{T_C^{(0)}(a) \mid C(a) \in \mathcal{A}\} \cup \\ & \{r^{(0)}(a,b) \mid r(a,b) \in \mathcal{A}\} \cup \\ & \{\neg r^{(0)}(a,b) \mid \neg r(a,b) \in \mathcal{A}\}. \end{aligned}$$

We can now assemble $\mathcal{A}_{\mathsf{red}}$:

$$\begin{aligned} \mathcal{A}_{\mathsf{red}} := \; & \mathcal{A}_{\mathsf{ini}} \cup \mathcal{A}_{\mathsf{aux}} \cup \\ & \mathcal{A}^{(1)}_{\mathsf{post}} \cup \cdots \cup \mathcal{A}^{(n)}_{\mathsf{post}} \cup \\ & \mathcal{A}^{(1)}_{\mathsf{min}} \cup \cdots \cup \mathcal{A}^{(n)}_{\mathsf{min}}. \end{aligned}$$

Finally, the reduction assertion φ_{red} is defined as $T^{(n)}_{A_0}(a_0)$. Then we have the following.

Lemma 3.2.3. *φ is a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff $\mathcal{A}_{\mathsf{red}}, \mathcal{T}_{\mathsf{red}} \models \varphi_{\mathsf{red}}$ under UNA.*

Proof. We first introduce a few notions that we are going to use in the proof. With Con, we denote the set of concept names that appear in the input, with Prim concept names from the input which are primitive in $\mathcal{T}$, and with Rol the set of role names that appear in the input. Moreover, if $\mathcal{I}$ is an interpretation, we denote with $\mathsf{Ind}^{\mathcal{I}}$ the set $\{a^{\mathcal{I}} \mid a \in \mathsf{Ind}\}$. Finally, Assert will denote the set of assertions that appear in the input.

"⇒" We prove this direction by contraposition. Assume that $\mathcal{A}_{\mathsf{red}}, \mathcal{T}_{\mathsf{red}} \not\models \varphi_{\mathsf{red}}$. This means that there is an interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$, $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$, and $\mathcal{J} \not\models T^{(n)}_{A_0}(a_0)$. In order to show that $\varphi = A_0(a_0)$ is not a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$, we have to find interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ such that $\mathcal{I}_0 \models \mathcal{A}$, $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_i$ for $1 \leq i \leq n$, and $\mathcal{I}_n \not\models A_0(a_0)$.

Let us define interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$, based on $\mathcal{J}$, in the following way:

- $\Delta^{\mathcal{I}_i} := \Delta^{\mathcal{J}}$
- $a^{\mathcal{I}_i} := a^{\mathcal{J}}$ for $a \in \mathsf{Ind}$
- $A^{\mathcal{I}_i} := (T^{(i)}_A)^{\mathcal{J}}$ for $A \in \mathsf{Con}$
- $r^{\mathcal{I}_i} := (r^{(i)})^{\mathcal{J}} \cap (N^{\mathcal{J}} \times N^{\mathcal{J}}) \cup (r^{(0)})^{\mathcal{J}} \cap \left(\Delta^{\mathcal{J}} \times (\neg N)^{\mathcal{J}} \cup (\neg N)^{\mathcal{J}} \times \Delta^{\mathcal{J}}\right)$ for $r \in \mathsf{Rol}$

Claim 1. For $i \leq n$, the following holds:

(a) If $a \in \mathsf{Ind}$, then $a^{\mathcal{I}_i} \in A^{\mathcal{I}_i}$ iff $a^{\mathcal{J}} \in (A^{(i)})^{\mathcal{J}}$, for all $A \in \mathsf{Prim}$
If $x \notin \mathsf{Ind}^{\mathcal{J}}$, then $x \in A^{\mathcal{I}_i}$ iff $x \in (A^{(0)})^{\mathcal{J}}$, for all $A \in \mathsf{Prim}$

(b) If $a, b \in \mathsf{Ind}$ then, for all $r \in \mathsf{Rol}$:

$$(a^{\mathcal{I}_i}, b^{\mathcal{I}_i}) \in r^{\mathcal{I}_i} \text{ iff } (a^{\mathcal{J}}, b^{\mathcal{J}}) \in (r^{(i)})^{\mathcal{J}}$$

If $x \notin \mathsf{Ind}^{\mathcal{J}}$ or $y \notin \mathsf{Ind}^{\mathcal{J}}$ then, for all $r \in \mathsf{Rol}$:

$$(x, y) \in r^{\mathcal{I}_i} \text{ iff } (x, y) \in (r^{(0)})^{\mathcal{J}}$$

(c) $E^{\mathcal{I}_i} = (T^{(i)}_E)^{\mathcal{J}}$ for every $E \in \mathsf{Sub}$

(d) $\mathcal{I}_i \models \varphi$ iff $\mathcal{J} \models a : \mathsf{p}_i(\varphi)$ for all $\varphi \in \mathsf{Assert}$ and $a \in \mathsf{Ind} \cup \{a_{\mathsf{help}}\}$

Proof.

(a) follows from the fact that $A^{\mathcal{I}_i} = (T^{(i)}_A)^{\mathcal{J}} = ((A^{(i)})^{\mathcal{J}} \cap N^{\mathcal{J}}) \cup ((A^{(0)})^{\mathcal{J}} \cap (\neg N)^{\mathcal{J}})$ and $N^{\mathcal{J}} = \{a^{\mathcal{J}} \mid a \in \mathsf{Ind}\}$ (due to $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$ and the definition of $A^{\mathcal{I}_i}$).

(b) follows directly from the definition of $r^{\mathcal{I}_i}$.

(c) is proved by structural induction on E:

- $E = A$, where $A \in \mathsf{Con}$. We have that $A^{\mathcal{I}_i} = (T^{(i)}_A)^{\mathcal{J}}$ by definition of $\mathcal{I}_i$
- $E = \{a\}$, where $a \in \mathsf{Ind}$. We have that $\{a\}^{\mathcal{I}_i} = (T^{(i)}_{\{a\}})^{\mathcal{J}} = \{a\}^{\mathcal{I}_i}$ since $a^{\mathcal{I}_i} = a^{\mathcal{J}}$

– $E = \neg C$: $(\neg C)^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i} \setminus C^{\mathcal{I}_i} = \Delta^{\mathcal{J}} \setminus (T_C^{(i)})^{\mathcal{J}} = (\neg T_C^{(i)})^{\mathcal{J}} = (T_{\neg C}^{(i)})^{\mathcal{J}}$ holds since $C^{\mathcal{I}_i} = (T_C^{(i)})^{\mathcal{J}}$ by induction, and since $\mathcal{J}$ satisfies (c) of $\mathcal{T}_{\mathsf{sub}}$.

– $E = C \sqcap D$: $(C \sqcap D)^{\mathcal{I}_i} = C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i} = (T_C^{(i)})^{\mathcal{J}} \cap (T_D^{(i)})^{\mathcal{J}} = (T_C^{(i)} \sqcap T_D^{(i)})^{\mathcal{J}} = (T_{C \sqcap D}^{(i)})^{\mathcal{J}}$ holds since $C^{\mathcal{I}_i} = (T_C^{(i)})^{\mathcal{J}}$ and $D^{\mathcal{I}_i} = (T_D^{(i)})^{\mathcal{J}}$ by induction, and since $\mathcal{J}$ satisfies (d) of $\mathcal{T}_{\mathsf{sub}}$.

– $E = C \sqcup D$ is similar to the previous case.

– $E = (\geq m\ r\ C)$: since $\mathcal{J}$ satisfies (f) of $\mathcal{T}_{\mathsf{sub}}$, we have that $x \in (T_{(\geq m\ r\ C)}^{(i)})^{\mathcal{J}}$ iff one of the following holds:

$$x \in \left(N \sqcap \bigsqcup_{0 \leq j \leq \min\{m, |\mathsf{Ind}|\}} \left(\left(\geq j\ r^{(i)}\ (N \sqcap T_C^{(i)}) \right) \sqcap \left(\geq (m-j)\ r^{(0)}\ (\neg N \sqcap T_C^{(i)}) \right) \right) \right)^{\mathcal{J}}$$

$$\text{or} \quad x \in \left(\neg N \sqcap \left(\geq m\ r^{(0)}\ T_C^{(i)} \right) \right)^{\mathcal{J}}$$

Thus, we obtain that:

$$x \in N^{\mathcal{J}} \wedge \bigvee_{0 \leq j \leq \min\{m, |\mathsf{Ind}|\}} \Big(\begin{array}{l} \#\{y \mid (x,y) \in (r^{(i)})^{\mathcal{J}} \wedge y \in \left(N^{\mathcal{J}} \cap (T_C^{(i)})^{\mathcal{J}}\right)\} \geq j \wedge \\ \#\{y \mid (x,y) \in (r^{(0)})^{\mathcal{J}} \wedge y \in \left((\neg N)^{\mathcal{J}} \cap (T_C^{(i)})^{\mathcal{J}}\right)\} \\ \qquad\qquad \geq (m-j) \end{array} \Big)$$

$$\text{or} \quad x \in (\neg N)^{\mathcal{J}} \wedge \#\{y \mid (x,y) \in (r^{(0)})^{\mathcal{J}} \wedge y \in \left((\neg N)^{\mathcal{J}} \cap (T_C^{(i)})^{\mathcal{J}}\right)\} \geq m$$

By induction, we have that $C^{\mathcal{I}_i} = (T_C^{(i)})^{\mathcal{J}}$. Thus, using the definition of $r^{\mathcal{I}_i}$, we have that the above disjunction holds iff:

$$x \in N^{\mathcal{J}} \wedge \bigvee_{0 \leq j \leq \min\{m, |\mathsf{Ind}|\}} \Big(\begin{array}{l} \#\{y \mid (x,y) \in r^{\mathcal{I}_i} \wedge y \in \left(N^{\mathcal{J}} \cap C^{\mathcal{I}_i}\right)\} \geq j \wedge \\ \#\{y \mid (x,y) \in r^{\mathcal{I}_i} \wedge y \in \left((\neg N)^{\mathcal{J}} \cap C^{\mathcal{I}_i}\right)\} \geq (m-j) \end{array} \Big)$$

$$\text{or} \quad x \in (\neg N)^{\mathcal{J}} \wedge \#\{y \mid (x,y) \in r^{\mathcal{I}_i} \wedge y \in \left((\neg N)^{\mathcal{J}} \cap C^{\mathcal{I}_i}\right)\} \geq m$$

By the semantics, and since $|N^{\mathcal{J}}| = |\mathsf{Ind}|$, this is equivalent to:

$$x \in (\geq m\ r\ C)^{\mathcal{I}_i}.$$

– $E = (\leq m\ r\ C)$: similar to the previous case

(d) follows directly from **(b)**, **(c)**, and the fact that

$$\mathcal{J} \models \{a : (\exists r_b.\{b\} \sqcap \forall r_b.\{b\}) \mid a \in \mathsf{Ind} \cup \{a_{\mathsf{help}}\},\ b \in \mathsf{Ind}\}.$$

This finishes the proof of the claim. Next, we will show that $\mathcal{I}_0 \models \mathcal{A}$, $\mathcal{I}_{i-1} \Rightarrow_{\alpha_i}^{\mathcal{T}} \mathcal{I}_i$ for all $1 \leq i \leq n$, and $\mathcal{I}_n \not\models A_0(a_0)$:

- $\mathcal{I}_0 \models \mathcal{A}$: this follows immediately from Claim 1 ((b) and (c)) and $\mathcal{J} \models \mathcal{A}_{\mathsf{ini}}$.

- $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_i$ for all $1 \leq i \leq n$ is split into three sub-tasks showing that the conditions from Definition 3.1.4 are satisfied:

  (i) Let $A \in \mathsf{Prim}$. Then $\alpha_{i+}^{\mathcal{I}_{i-1}}(A) \cap \alpha_{i-}^{\mathcal{I}_{i-1}}(A) = \emptyset$ since α_i is consistent with $\mathcal{T}$. It remains to show that if $x \in D_{\alpha_i}^{\mathcal{I}_{i-1}}(A)$, then

  $$x \in A^{\mathcal{I}_i} \text{ iff } x \in (A^{\mathcal{I}_{i-1}} \cup \alpha_{i+}^{\mathcal{I}_{i-1}}(A)) \setminus \alpha_{i-}^{\mathcal{I}_{i-1}}(A) \qquad (*)$$

  We distinguish among the following cases for $x \in D_{\alpha_i}^{\mathcal{I}_{i-1}}(A)$:

  * $x \notin \mathsf{Ind}^{\mathcal{J}}$: Then Claim 1(a) implies that $x \in A^{\mathcal{I}_i}$ iff $x \in (A^{(0)})^{\mathcal{J}}$ iff $x \in A^{\mathcal{I}_{i-1}}$. Since $x \notin \alpha_{i+}^{\mathcal{I}_{i-1}}(A)) \cup \alpha_{i-}^{\mathcal{I}_{i-1}}(A))$, we have shown $(*)$.
  * $x = a^{\mathcal{J}}$, $a \in \mathsf{Ind}$. Then Claim 1(a),(d), together with $\mathcal{J} \models \mathcal{A}^{(i)}_{\mathsf{post}}$, $\mathcal{J} \models \mathcal{A}^{(i)}_{\mathsf{min}}$, gives us the following implications: $a^{\mathcal{J}} \in \alpha_{i+}^{\mathcal{I}_{i-1}}(A)$ implies $a^{\mathcal{J}} \in A^{\mathcal{I}_i}$, $a^{\mathcal{J}} \in \alpha_{i-}^{\mathcal{I}_{i-1}}(A)$ implies $a^{\mathcal{J}} \notin A^{\mathcal{I}_i}$. If $A(a) \notin \mathsf{occ}_i$, we have that: $a^{\mathcal{J}} \in A^{\mathcal{I}_{i-1}} \setminus \alpha_{i-}^{\mathcal{I}_{i-1}}(A)$ implies $a^{\mathcal{J}} \in A^{\mathcal{I}_i}$, and $a^{\mathcal{J}} \in A^{\mathcal{I}_i}$ implies $a^{\mathcal{J}} \in A^{\mathcal{I}_{i-1}} \cup \alpha_{i+}^{\mathcal{I}_{i-1}}(A)$. It is not difficult to see that $(*)$ is implied.

  Thus, we have shown that

  $$A^{\mathcal{I}_i} \cap D_{\alpha_i}^{\mathcal{I}_{i-1}}(A) = ((A^{\mathcal{I}_{i-1}} \cup \alpha_{i+}^{\mathcal{I}_{i-1}}(A)) \setminus \alpha_{i-}^{\mathcal{I}_{i-1}}(A)) \cap D_{\alpha_i}^{\mathcal{I}_{i-1}}(A).$$

  (ii) Let $r \in \mathsf{Rol}$. Then $\alpha_{i+}^{\mathcal{I}_{i-1}}(r) \cap \alpha_{i-}^{\mathcal{I}_{i-1}}(r) = \emptyset$ since α_i is consistent with $\mathcal{T}$. Showing $r^{\mathcal{I}_i} \cap D_{\alpha_i}^{\mathcal{I}_{i-1}}(r) = ((r^{\mathcal{I}_{i-1}} \cup \alpha_{i+}^{\mathcal{I}_{i-1}}(r)) \setminus \alpha_{i-}^{\mathcal{I}_{i-1}}(r)) \cap D_{\alpha_i}^{\mathcal{I}_{i-1}}(r)$ is analogous to (i).

  (iii) $\mathcal{I}_i \models \mathcal{T}$, $i \leq n$, is an immediate consequence of Claim 1(c): Let $A \doteq E$ be a concept definition in $\mathcal{T}$. We have that $T_A^{(i)} \doteq T_E^{(i)} \in \mathcal{T}_{\mathsf{red}}$ and, since $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$, obtain $(T_A^{(i)})^{\mathcal{J}} = (T_E^{(i)})^{\mathcal{J}}$. Claim 1(c) gives us that $A^{\mathcal{I}_i} = (T_A^{(i)})^{\mathcal{J}} = (T_E^{(i)})^{\mathcal{J}} = E^{\mathcal{I}_i}$.

- Finally, it is obvious that Claim 1(c) and $\mathcal{J} \not\models T_{A_0}^{(n)}(a_0)$ imply $\mathcal{I}_n \not\models A_0(a_0)$.

"$\Leftarrow$": For this direction, we also show the contrapositive. Assume that $\varphi = A_0(a_0)$ is not a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$. Then there are interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ such that $\mathcal{I}_0 \models \mathcal{A}$, $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_i$ for $1 \leq i \leq n$, and $\mathcal{I}_n \not\models A_0(a_0)$. We show that, then, $T_{A_0}^{(n)}(a_0)$ is not a consequence of $\mathcal{A}_{\mathsf{red}}$ w.r.t. $\mathcal{T}_{\mathsf{red}}$.

Claim 2 The interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ satisfy the following:

(a) For all $a \in \mathsf{Ind}$ and $A \in \mathsf{Prim}$ such that $A(a) \notin \mathsf{occ}_i$, the following holds:

if $\mathcal{I}_{i-1} \models A(a)$ and, for each $\varphi/\neg A(a) \in \mathsf{post}_i$, $\mathcal{I}_{i-1} \not\models \varphi$, then $\mathcal{I}_i \models A(a)$, and

if $\mathcal{I}_{i-1} \models \neg A(a)$ and, for each $\varphi/A(a) \in \mathsf{post}_i$, $\mathcal{I}_{i-1} \not\models \varphi$, then $\mathcal{I}_i \models \neg A(a)$.

For all $a, b \in \mathsf{Ind}$ and $r \in \mathsf{Rol}$ such that $r(a,b) \notin \mathsf{occ}_i$, the following holds:

if $\mathcal{I}_{i-1} \models r(a,b)$ and, for each $\varphi/\neg r(a,b) \in \mathsf{post}_i$, $\mathcal{I}_{i-1} \not\models \varphi$, then $\mathcal{I}_i \models r(a,b)$

if $\mathcal{I}_{i-1} \models \neg r(a,b)$ and, for each $\varphi/r(a,b) \in \mathsf{post}_i$, $\mathcal{I}_{i-1} \not\models \varphi$, then $\mathcal{I}_i \models \neg r(a,b)$.

(b) If $x \notin \mathsf{Ind}^{\mathcal{I}_0}$, then $x \in A^{\mathcal{I}_i}$ iff $x \in A^{\mathcal{I}_0}$, for all $A \in \mathsf{Prim}$

(c) If $x \notin \mathsf{Ind}^{\mathcal{I}_0}$ or $y \notin \mathsf{Ind}^{\mathcal{I}_0}$, then $(x, y) \in r^{\mathcal{I}_i}$ iff $(x, y) \in r^{\mathcal{I}_0}$, for all $r \in \mathsf{Rol}$

Proof.
(a) Let $A \in \mathsf{Prim}$. Since $A(a) \notin \mathsf{occ}_i$, we have that $a^{\mathcal{I}_{i-1}} \in D_{\alpha_i}^{\mathcal{I}_{i-1}}(A)$. Thus, since $\mathcal{I}_{i-1} \Rightarrow_{\alpha_i}^{\mathcal{T}} \mathcal{I}_i$, we have that $a^{\mathcal{I}_{i-1}} \in A^{\mathcal{I}_i}$ iff $a^{\mathcal{I}_{i-1}} \in ((A^{\mathcal{I}_{i-1}} \cup \alpha_{i+}^{\mathcal{I}_{i-1}}(A)) \setminus \alpha_{i-}^{\mathcal{I}_{i-1}}(A))$. Therefore, $\mathcal{I}_{i-1} \models A(a)$ and $a^{\mathcal{I}_{i-1}} \notin \alpha_{i-}^{\mathcal{I}_{i-1}}(A)$ imply $\mathcal{I}_i \models A(a)$.

Other cases, including those for $r \in \mathsf{Rol}$ are analogous.
(b) In a similar way to **(a)**, we can show that $x \in A^{\mathcal{I}_i}$ iff $x \in A^{\mathcal{I}_{i-1}}$ for all $x \notin \mathsf{Ind}^{\mathcal{I}_0}$ and $1 \leq i \leq n$. As an immediate consequence, we obtain **(b)**.
(c) Analogous to **(b)**.

This finishes the proof of Claim 2.

We define an interpretation $\mathcal{J}$ in the following way:

- $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}_0}$ $(= \Delta^{\mathcal{I}_1} = \cdots = \Delta^{\mathcal{I}_n})$
- $a^{\mathcal{J}} := a^{\mathcal{I}_0}$ $(= a^{\mathcal{I}_1} = \cdots = a^{\mathcal{I}_n})$ for $a \in \mathsf{Ind}$
- $a_{\mathsf{help}}^{\mathcal{J}} := d$, for an arbitrary $d \in \Delta^{\mathcal{J}}$
- $N^{\mathcal{J}} := \{a^{\mathcal{J}} \mid a \in \mathsf{Ind}\}$
- $r_b^{\mathcal{J}} := \{(a^{\mathcal{J}}, b^{\mathcal{J}}) \mid a \in \mathsf{Ind} \cup \{a_{\mathsf{help}}\}\}$, for all $b \in \mathsf{Ind}$
- $(A^{(i)})^{\mathcal{J}} := A^{\mathcal{I}_i}$ for $A \in \mathsf{Con}$ and $i \leq n$
- $(r^{(i)})^{\mathcal{J}} := r^{\mathcal{I}_i}$ for $r \in \mathsf{Rol}$ and $i \leq n$
- $(T_C^{(i)})^{\mathcal{J}} := C^{\mathcal{I}_i}$ for all $C \in \mathsf{Sub}$ and $i \leq n$

Please note that the definition of $\mathcal{J}$ implies that, for all $i \leq n$, $\varphi \in \mathsf{Assert}$, and $a \in \mathsf{Ind} \cup \{a_{\mathsf{help}}\}$, we have the following:

$$\mathcal{I}_i \models \varphi \quad \text{iff} \quad \mathcal{J} \models a : \mathsf{p}_i(\varphi) \tag{$**$}$$

We will show now that $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$, $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$, and $\mathcal{J} \not\models T_{A_0}(a_0)$.

(i) $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$:

- $\mathcal{J} \models \mathcal{A}_{\mathsf{ini}}$ follows directly from $\mathcal{I}_0 \models \mathcal{A}$ and the definition of $\mathcal{J}$.
- $\mathcal{J}$ models $\mathcal{A}_{\mathsf{aux}}$ by definition of $r_b^{\mathcal{J}}$.
- $\mathcal{J} \models \mathcal{A}_{\mathsf{post}}^{(i)}$, for each $1 \leq i \leq n$: by $(**)$, we obtain that:

  $$\text{If } \mathcal{I}_{i-1} \models \varphi \text{ implies } \mathcal{I}_i \models \psi, \text{ then } \mathcal{J} \models a_{\mathsf{help}} : (\mathsf{p}_{i-1}(\varphi) \rightarrow \mathsf{p}_i(\psi))$$

  for every $\varphi/\psi \in \mathsf{post}_i$. Since $\mathcal{I}_{i-1} \Rightarrow_{\alpha_i}^{\mathcal{T}} \mathcal{I}_i$, we have that $\mathcal{J}$ satisfies $\mathcal{A}_{\mathsf{post}}^{(i)}$.

- $\mathcal{J} \models \mathcal{A}^{(i)}_{\mathsf{min}}$, for each $1 \leq i \leq n$: by $(**)$ and definitions of $(A^{(i)})^{\mathcal{J}}$, we have that

  If $\mathcal{I}_{i-1} \models A(a)$ and, for all $\varphi/\neg A(a) \in \mathsf{post}_i$, $\mathcal{I}_{i-1} \not\models \varphi$ imply $\mathcal{I}_i \models A(a)$,
  then $\mathcal{J} \models a : \Big(\big(A^{(i-1)} \sqcap \bigsqcap_{\varphi/\neg A(a) \in \mathsf{post}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \rightarrow A^{(i)}\Big)$

  The symmetric case for $(\neg A)^{\mathcal{J}}$ and the cases for $(r^{(i)})^{\mathcal{J}}$ can be considered in a similar way. Thus, by Claim 2(a), we have that $\mathcal{J}$ satisfies $\mathcal{A}^{(i)}_{\mathsf{min}}$.

(ii) $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$:

- $\mathcal{J}$ satisfies the concept definition $N \doteq \bigsqcup_{a \in \mathsf{Ind}} \{a\}$ by definition of $N^{\mathcal{J}}$.

- $\mathcal{J}$ satisfies every concept definition $T^{(i)}_A \doteq T^{(i)}_C$, where $A \doteq C \in \mathcal{T}$ and $i \leq n$: by definition of $(T^{(i)}_A)^{\mathcal{J}}$ and since $\mathcal{I}_i \models \mathcal{T}$, we have that $(T^{(i)}_A)^{\mathcal{J}} = A^{\mathcal{I}_i} = C^{\mathcal{I}_i} = (T^{(i)}_C)^{\mathcal{J}}$.

- Finally, we will show that $\mathcal{J} \models \mathcal{T}_{\mathsf{sub}}$. By structural induction on $E \in \mathsf{Sub}$, we show that $\mathcal{J}$ satisfies every concept definition with $T^{(i)}_E$ on the left-hand side:

  - $E = A$, where $A \in \mathsf{Prim}$. We have:

    $$\begin{aligned}(T^{(i)}_A)^{\mathcal{J}} &= A^{\mathcal{I}_i} = N^{\mathcal{J}} \cap A^{\mathcal{I}_i} \cup (\neg N)^{\mathcal{J}} \cap A^{\mathcal{I}_i} \\ &= N^{\mathcal{J}} \cap A^{\mathcal{I}_i} \cup (\neg N)^{\mathcal{J}} \cap A^{\mathcal{I}_0} = N^{\mathcal{J}} \cap (A^{(i)})^{\mathcal{J}} \cup (\neg N)^{\mathcal{J}} \cap (A^{(0)})^{\mathcal{J}} = \\ &= ((N \sqcap A^{(i)}) \sqcup (\neg N \sqcap A^{(0)}))^{\mathcal{J}}\end{aligned}$$

    The first equality holds by definition of $(T^{(i)}_A)^{\mathcal{J}}$, the second one by the semantics, the third one by Claim 2 (b) and the definition of $N^{\mathcal{J}}$, the fourth one by definition of $(A^{(j)})^{\mathcal{J}}$, and the last one by the semantics.

  - $E = \neg C$. By definition of $(T^{(i)}_{\neg C})^{\mathcal{J}}$ and $(T^{(i)}_C)^{\mathcal{J}}$, we have the following:

    $$(T^{(i)}_{\neg C})^{\mathcal{J}} = (\neg C)^{\mathcal{I}_i} = \neg C^{\mathcal{I}_i} = \neg (T^{(i)}_C)^{\mathcal{J}}.$$

  - $E = C \sqcap D$. By definition of $(T^{(i)}_{C \sqcap D})^{\mathcal{J}}$, $(T^{(i)}_C)^{\mathcal{J}}$ and $(T^{(i)}_D)^{\mathcal{J}}$, we have the following:

    $$(T^{(i)}_{C \sqcap D})^{\mathcal{J}} = (C \sqcap D)^{\mathcal{I}_i} = C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i} = (T^{(i)}_C)^{\mathcal{J}} \cap (T^{(i)}_D)^{\mathcal{J}} = (T^{(i)}_C \sqcap T^{(i)}_D)^{\mathcal{J}}$$

  - $E = C \sqcup D$ is similar to the previous case.

  - $E = (\geq\ m\ r\ C)$. By definition of $\mathcal{J}$, we have that $x \in (T^{(i)}_{(\geq m\ r\ C)})^{\mathcal{J}}$ iff $x \in (\geq\ m\ r\ C)^{\mathcal{I}_i}$. Due to the definition of the semantics, the latter is the case iff $\#\{y \mid (x,y) \in r^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i}\} \geq m$.
    By Claim 2(c) and $N^{\mathcal{J}} = \{a^{\mathcal{J}} \mid a \in \mathsf{Ind}\}$, the above expression is equivalent to the following disjunction:

    $$x \in N^{\mathcal{J}} \wedge \bigvee_{0 \leq j \leq \min\{m, |\mathsf{Ind}|\}} \Big(\begin{array}{l} \#\{y \mid (x,y) \in r^{\mathcal{I}_i} \wedge y \in \big(N^{\mathcal{J}} \cap C^{\mathcal{I}_i}\big)\} \geq j \wedge \\ \#\{y \mid (x,y) \in r^{\mathcal{I}_0} \wedge y \in \big((\neg N)^{\mathcal{J}} \cap C^{\mathcal{I}_i}\big)\} \geq (m-j) \end{array} \Big)$$

    or

    $$x \in (\neg N)^{\mathcal{J}} \wedge \#\{y \mid (x,y) \in r^{\mathcal{I}_0} \wedge y \in C^{\mathcal{I}_i}\} \geq m$$

Using the definitions of $(T_C^{(i)})^{\mathcal{J}}$ and $(r^{(i)})^{\mathcal{J}}$, we obtain:

$$x \in N^{\mathcal{J}} \wedge \bigvee_{0 \leq j \leq \min\{m, |\mathsf{Ind}|\}} \Big(\begin{array}{l} \#\{y \mid (x,y) \in (r^{(i)})^{\mathcal{J}} \wedge y \in \big(N^{\mathcal{J}} \cap (T_C^{(i)})^{\mathcal{J}}\big)\} \geq j \wedge \\ \#\{y \mid (x,y) \in (r^{(0)})^{\mathcal{J}} \wedge y \in \big((\neg N)^{\mathcal{J}} \cap (T_C^{(i)})^{\mathcal{J}}\big)\} \\ \qquad \geq (m-j) \end{array} \Big)$$

or

$$x \in (\neg N)^{\mathcal{J}} \wedge \#\{y \mid (x,y) \in (r^{(0)})^{\mathcal{J}} \wedge y \in (T_C^{(i)})^{\mathcal{J}}\} \geq m,$$

which is equivalent to:

$$x \in \Bigg[\Bigg(N \sqcap \bigsqcup_{0 \leq j \leq \min\{m, |\mathsf{Ind}|\}} \Big(\big(\geq j \; r^{(i)} \; (N \sqcap T_C^{(i)})\big) \sqcap \big(\geq (m-j) \; r^{(0)} \; (\neg N \sqcap T_C^{(i)})\big)\Big)\Bigg) \sqcup \Big(\neg N \sqcap \big(\geq m \; r^{(0)} \; T_C^{(i)}\big)\Big)\Bigg]^{\mathcal{J}}.$$

– $E = (\leq m \; r \; C)$ is similar to the previous case.

Hence $\mathcal{J}$ satisfies $\mathcal{T}_{\mathsf{sub}}$.

(iii) Finally, it is easy to see that $(A_0)^{\mathcal{I}_n} = (T_{A_0}^{(n)})^{\mathcal{J}}$ and $\mathcal{I}_n \not\models A_0(a_0)$ imply $\mathcal{J} \not\models T_{A_0}^{(n)}(a_0)$. ❏

Observation 3.2.4. *The previous reduction can easily be adapted to the more general case where we want to decide if $\mathcal{T}, \mathcal{A}^{\alpha_1,\ldots,\alpha_n} \models \mathcal{B}$, where $\mathcal{B}$ is an ABox, i.e. a set of assertions, rather then a single ABox assertion φ.*

Proof. Similarly to the assumption from the afore presented reduction, we may assume that $\mathcal{B} = \{A_0(a_0), A_1(a_1), \ldots, A_l(a_l)\}$, where A_j are concept names for $j \leq l$, possibly defined in $\mathcal{T}$. Then it suffices to modify φ_{red}. We set $\varphi_{\mathsf{red}} := T_{\mathcal{B}}^{(n)}(a_0)$, where

$$T_{\mathcal{B}}^{(n)} := T_{A_0}^{(n)} \sqcap \exists r_{a_1}.T_{A_1}^{(n)} \sqcap \cdots \sqcap \exists r_{a_l}.T_{A_l}^{(n)}.$$

It is easy to show that the modified reduction is correct. ❏

Since the size of $\mathcal{A}_{\mathsf{red}}$, $\mathcal{T}_{\mathsf{red}}$, and φ_{red} are clearly polynomial in the size of the input, Lemma 3.2.3 immediately yields Theorem 3.2.2. Thus, for the DLs $\mathcal{L}$ considered in Theorem 3.2.2, upper complexity bounds for ABox consequence in $\mathcal{LO}$ carry over to projection in $\mathcal{L}$. Many such upper bounds are available from the literature. Lower complexity bounds carry over from ABox consequence in a DL $\mathcal{L}$ to projection in the same DL: $\mathcal{T}, \mathcal{A} \models \varphi$ iff φ is a consequence of applying the empty action $(\emptyset, \emptyset, \emptyset)$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$. Thus, we obtain tight bounds for projection in those DLs $\mathcal{L}$ where the addition of nominals does *not* increase the complexity of reasoning. Please note, that due to Observation 3.2.4, the upper bounds hold also in a more general case where the assertion φ is replaced by an ABox $\mathcal{B}$.

Corollary 3.2.5. *Executability and projection in $\mathfrak{A}_1$ w.r.t. acyclic TBoxes are*

1. PSPACE*-complete for $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCQ}$, and $\mathcal{ALCQO}$;*

2. *in* ExpTime *for* $\mathcal{ALCI}$;
3. ExpTime-*complete for* $\mathcal{ALCIO}$;
4. *in co-*NExpTime *for* $\mathcal{ALCQI}$;
5. *co-*NExpTime-*complete for* $\mathcal{ALCQIO}$.

Points 1, 4, and 5 hold even if numbers in number restrictions are coded in binary.

Proof. The corollary is a consequence of Theorem 3.2.2 and the following results: ABox consequence in

- $\mathcal{ALC}$ w.r.t. acyclic TBoxes is PSpace-hard [SSS91] (yields lower bounds of Point 1);
- $\mathcal{ALCQO}$ w.r.t. acyclic TBoxes is in PSpace [BLM$^+$05d] (yields upper bounds of Point 1);
- $\mathcal{ALCIO}$ w.r.t. acyclic TBoxes is ExpTime-complete, as follows from results in [ABM99] (yields Points 2 and 3);
- $\mathcal{ALCQIO}$ is co-NExpTime-complete as follows from results in [Tob00] and [PST00] (yields Points 4 and 5).

The bounds for executability are then obtained by the reductions of executability to projection and vice versa. ❑

In Section 3.2.2, we prove matching lower bounds for Points 2 and 4 of Corollary 3.2.5.

3.2.2 Hardness Results

The aim of this section is to prove that the upper bounds for projection obtained in the previous two sections are tight, i.e., that they have matching lower bounds. In Section 3.2.1, we have already obtained tight lower bounds for those DLs $\mathcal{L}$ where the complexity of ABox consequence in $\mathcal{L}$ coincides with the complexity of ABox consequence in $\mathcal{LO}$, $\mathcal{L}$'s extension with nominals. It thus remains to consider cases where ABox consequence in $\mathcal{LO}$ is more difficult than in $\mathcal{L}$: we prove an ExpTime lower bound for projection in $\mathcal{ALCI}$ and a co-NExpTime lower bound for projection in (a fragment of) $\mathcal{ALCQI}$.These bounds match Points 2 and 4 of Corollary 3.2.5. The results established in this section show that the additional complexity that is obtained by introducing nominals in the reduction of projection to ABox consequence cannot be avoided.

The idea for proving the lower bounds is to reduce, for $\mathcal{L} \in \{\mathcal{ALCI}, \mathcal{ALCQI}\}$, unsatisfiability of $\mathcal{LO}$ concepts to projection in $\mathcal{L}$. In the case of $\mathcal{ALCQI}$, we can even obtain a slightly stronger result by reducing concept unsatisfiability in $\mathcal{ALCFIO}$ to projection in $\mathcal{ALCFI}$, where $\mathcal{ALCFIO}$ is $\mathcal{ALCQIO}$ with numbers occurring in number restrictions limited to $\{0, 1\}$, and $\mathcal{ALCFI}$ is obtained from $\mathcal{ALCFIO}$ by dropping nominals.[4]

Theorem 3.2.6. *There exists an ABox $\mathcal{A}$ and an atomic action α formulated in $\mathcal{ALCI}$ ($\mathcal{ALCFI}$) such that, given an $\mathcal{ALCI}$-assertion ($\mathcal{ALCFI}$-assertion), the following tasks are* ExpTime-*hard (co-*NExpTime-*hard):*

- *decide whether φ is a consequence of applying α in $\mathcal{A}$;*
- *decide whether $\alpha, (\{\varphi\}, \emptyset, \emptyset)$ is executable in $\mathcal{A}$.*

[4] We have to admit the number 0 for being able to still use the abbreviation $\forall r.C$ that stands for $(\leqslant 0\ r\ \neg C)$.

Note that we cannot obtain the same hardness results for executability of *atomic* actions for the following reasons: (i) executability of atomic actions in any DL $\mathcal{L}$ can trivially be reduced to ABox consequence in $\mathcal{L}$, and (ii) the complexity of ABox consequence is identical to the complexity of concept unsatisfiability in $\mathcal{ALCI}$ and $\mathcal{ALCFI}$.

For the proof of Theorem 3.2.6, let $\mathcal{L} \in \{\mathcal{ALCIO}, \mathcal{ALCFIO}\}$ and C an $\mathcal{L}$-concept whose (un)satisfiability is to be decided. For simplicity, we assume that C contains only a single nominal $\{n\}$. This can be done w.l.o.g. since the complexity of (un)satisfiability in $\mathcal{ALCIO}$ and $\mathcal{ALCFIO}$ is oblivious to the number of available nominals [ABM99, Tob00, Tob01]. We reserve two concept names O and A_N and a role name u that do not occur in C. Let

$$\mathsf{rol}(C) := \{r, r^- \mid r \in \mathsf{N_R} \text{ used in } C\}$$

and let $C[\{n\}/A_N]$ denote the result of replacing the nominal $\{n\}$ in C with the concept name A_N. We define an ABox $\mathcal{A}$, an atomic action $\alpha = (\emptyset, \emptyset, \mathsf{post}_\alpha)$, and a concept D_C as follows:

$$\begin{aligned}
\mathcal{A}_C &:= \{a : (\neg O \sqcap \forall u.\neg O \sqcap \forall u. \bigsqcap_{r \in \mathsf{rol}(C)} \forall r.\exists u^-.\neg O)\} \\
\mathsf{post}_\alpha &:= a : O \\
D_C &:= A_N \sqcap \exists u.C[\{n\}/A_N] \sqcap (\forall u. \bigsqcap_{r \in \mathsf{rol}(C)} \forall r.\forall u^-.O) \sqcap (\forall u.(A_N \leftrightarrow O))
\end{aligned}$$

This reduction does not involve any TBoxes since, on the one hand, we do not need one to carry out the reduction and, on the other hand, concept unsatisfiability is already ExpTime-complete in $\mathcal{ALCIO}$ [ABM99] and co-NExpTime-complete in $\mathcal{ALCFIO}$, even without TBoxes. Hence the following lemma immediately yields Theorem 3.2.6.

Lemma 3.2.7. *The following statements are equivalent:*

(1) C is satisfiable.

(2) there are interpretations $\mathcal{I}$ and $\mathcal{I}'$ such that $\mathcal{I} \models \mathcal{A}_C$, $\mathcal{I} \Rightarrow_\alpha \mathcal{I}'$, and $\mathcal{I}' \models a : D_C$.

(3) $a : \neg D_C$ is not a consequence of applying α in $\mathcal{A}_C$.

(4) the composite action $\alpha, (\{a : \neg D_C\}, \emptyset, \emptyset)$ is not executable in $\mathcal{A}_C$.

Proof. We only prove that (1) and (2) are equivalent since the other equivalences are immediate consequences of the definitions of projection and executability.
(2) implies (1). Assume that there are interpretations $\mathcal{I}$ and $\mathcal{I}'$ such that $\mathcal{I} \models \mathcal{A}_C$, $\mathcal{I} \Rightarrow_\alpha \mathcal{I}'$, and $\mathcal{I}' \models a : D_C$. By the first conjunct of (the concept in the only assertion of) $\mathcal{A}_C$, by post_α, and Lemma 3.1.5, we have that $\mathcal{I}$ is identical to $\mathcal{I}'$ with the only exception that $a^\mathcal{I} = a^{\mathcal{I}'} \in O^{\mathcal{I}'} \setminus O^\mathcal{I}$. For simplicity, we will call this relationship *quasi-identity* of $\mathcal{I}$ and $\mathcal{I}'$ in what follows. By the second conjunct of D_C, there is an $x_0 \in \Delta^\mathcal{I} = \Delta^{\mathcal{I}'}$ such that $(a^{\mathcal{I}'}, x_0) = (a^\mathcal{I}, x_0) \in u^{\mathcal{I}'} = u^\mathcal{I}$, and $x_0 \in C[\{n\}/A_N]^{\mathcal{I}'} = C[\{n\}/A_N]^\mathcal{I}$. The last equality holds since O does not occur in $C[\{n\}/A_N]$ and by the quasi-identity of $\mathcal{I}$ and $\mathcal{I}'$. We first identify the "relevant part" of $\mathcal{I}$ and $\mathcal{I}'$: set

$$\begin{aligned}
\mathsf{rel}_0 &:= \{x_0\} \\
\mathsf{rel}_{i+1} &:= \mathsf{rel}_i \cup \{x \in \Delta^\mathcal{I} \mid (y, x) \in r^\mathcal{I} \text{ for some } y \in \mathsf{rel}_i \text{ and } r \in \mathsf{rol}(C)\} \\
\mathsf{rel} &:= \bigcup_{i \geq 0} \mathsf{rel}_i
\end{aligned}$$

The relevant part of $\mathcal{I}'$ can be defined analogously. Due to quasi-identity of $\mathcal{I}$ and $\mathcal{I}$', it is identical to the relevant part of $\mathcal{I}$. We now show the following:

Claim 1. For all $x \in \mathsf{rel}$, we have $(a^{\mathcal{I}}, x) = (a^{\mathcal{I}'}, x) \in u^{\mathcal{I}} = u^{\mathcal{I}'}$.

The proof of the claim is by induction on the smallest i such that $x \in \mathsf{rel}_i$. First let $i = 0$. Then $x = x_0$ and the claim holds since $(a^{\mathcal{I}}, x_0) \in u^{\mathcal{I}}$ by choice of x_0. Now let $i > 0$. Since i is smallest with $x \in \mathsf{rel}_i$, there is a $y \in \mathsf{rel}_{i-1}$ such that $(y, x) \in r^{\mathcal{I}} = r^{\mathcal{I}'}$ for some $r \in \mathsf{rol}(C)$. By induction, we have $(a^{\mathcal{I}}, y) \in u^{\mathcal{I}} = u^{\mathcal{I}'}$. Thus, the third conjunct of $\mathcal{A}_C$ implies that $y \in (\forall r.\exists u^-.\neg O)^{\mathcal{I}}$, and thus $x \in (\exists u^-.\neg O)^{\mathcal{I}}$. Let z be the witness for this, i.e. $(z, x) \in u^{\mathcal{I}} = u^{\mathcal{I}'}$ and $z \notin O^{\mathcal{I}}$. Since $(a^{\mathcal{I}}, y) \in u^{\mathcal{I}'}$, $(y, x) \in r^{\mathcal{I}'}$, and $(z, x) \in u^{\mathcal{I}'}$, we have $z \in O^{\mathcal{I}'}$ by the third conjunct of D_C. Since the only difference between $\mathcal{I}$ and $\mathcal{I}'$ is $a^{\mathcal{I}} \in O^{\mathcal{I}'} \setminus O^{\mathcal{I}}$, $z \in O^{\mathcal{I}'} \setminus O^{\mathcal{I}}$ implies $z = a^{\mathcal{I}}$. Since we have already shown that $(z, x) \in u^{\mathcal{I}}$, we are done.

This finishes the proof of Claim 1. Now we show that the concept name A_N may serve as a nominal on the relevant part of $\mathcal{I}'$ extended with $a^{\mathcal{I}'}$.

Claim 2. $A_N^{\mathcal{I}'} \cap (\mathsf{rel} \cup \{a^{\mathcal{I}'}\})$ is a singleton.

Proof: By the first conjunct of D_C, we have $a^{\mathcal{I}'} \in A_N^{\mathcal{I}'}$. Thus, $A_N^{\mathcal{I}'}$ is non-empty. Now let $x \in A_N^{\mathcal{I}'} \cap \mathsf{rel}$. By Claim 1, we have $(a^{\mathcal{I}'}, x) \in u^{\mathcal{I}'}$. Thus, the fourth conjunct of D_C together with $x \in A_N^{\mathcal{I}'}$ implies $x \in O^{\mathcal{I}'}$. Claim 1 and the third conjunct of $\mathcal{A}_C$ yield $x \notin O^{\mathcal{I}}$. Since the only difference between $\mathcal{I}$ and $\mathcal{I}'$ is $a^{\mathcal{I}'} \in O^{\mathcal{I}'} \setminus O^{\mathcal{I}}$, this is implies that $x = a^{\mathcal{I}'}$. Hence, $A_N^{\mathcal{I}'} \cap (\mathsf{rel} \cup \{a^{\mathcal{I}'}\})$ is a singleton.

Now define an interpretation $\mathcal{J}$ as $\mathcal{I}'$ extended with $\{n\}^{\mathcal{J}} := A_N^{\mathcal{I}'} \cap (\mathsf{rel} \cup \{a^{\mathcal{I}'}\})$. By Claim 2, $\{n\}^{\mathcal{J}}$ is a singleton as required. It is standard to prove the following claim by structural induction. The only interesting aspects are the case of the nominal $\{n\}$ where we use that $\{n\}^{\mathcal{J}} = A_N^{\mathcal{I}'}$, and the fact that all domain elements encountered during the proof are from rel.

Claim 3. For all $x \in \mathsf{rel}$ and all subconcepts D of C, we have $x \in D[\{n\}/A_N]^{\mathcal{J}}$ iff $x \in D^{\mathcal{J}}$.

Finally, $x_0 \in C[\{n\}/A_N]^{\mathcal{I}'}$ yields $x_0 \in C^{\mathcal{J}}$ by Claim 3, and thus C is satisfiable.

(1) implies (2). Let $\mathcal{J}$ be a model of C, and let $x_0 \in C^{\mathcal{J}}$. Let $\mathcal{I}$ be the interpretation that is identical to $\mathcal{J}$, but for the following modifications:

- $A_N^{\mathcal{I}} = \{n\}^{\mathcal{J}}$;
- $O^{\mathcal{I}} = \emptyset$;
- $a^{\mathcal{I}} = n^{\mathcal{J}}$;
- $u^{\mathcal{I}} = \{(a^{\mathcal{I}}, x) \mid x \in \Delta^{\mathcal{I}}\}$.

Moreover, let $\mathcal{I}'$ be the interpretation that is identical to $\mathcal{I}$ except that $O^{\mathcal{I}'} = \{n\}^{\mathcal{J}}$. It is readily checked that $\mathcal{I} \models \mathcal{A}_C$, $\mathcal{I} \Rightarrow_\alpha \mathcal{I}'$, and $\mathcal{I}' \models a : D_C$. ❑

Chapter 4

Restrictions and Extensions of $\mathfrak{A}_1$

The previous chapter was dedicated to the instantiations of the action formalism $\mathfrak{A}_1$ with DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$. In this chapter, we investigate instantiations of $\mathfrak{A}_1$ with less and more expressive description logics. Section 4.1 is dedicated to the instantiations of $\mathfrak{A}_1$ with the lightweight DLs $\mathcal{EL}$ and $\mathcal{EL}^{(\neg)}$, with and without TBoxes. $\mathfrak{A}_1$ based on various extensions of $\mathcal{ALCQIO}$ is investigated in Section 4.2. It turns out that $\mathfrak{A}_1$ can easily be modified to account for role inclusions, while expressive means such as transitive roles and acyclic TBoxes introduce semantic and/or computational problems. Finally, in Section 4.3 we show that if we generalize $\mathfrak{A}_1$ by admitting complex concepts in action post-conditions and adopt the possible model approach (PMA) semantics, this leads to both non-intuitive results and undecidability of reasoning.

4.1 Restrictions

In this section, we investigate the projection problem in $\mathfrak{A}_1$ instantiated with the description logics $\mathcal{EL}$ and $\mathcal{EL}^{(\neg)}$. Recall that $\mathcal{EL}$ is a lightweight DL which provides only for the following constructors: conjunction ($\sqcap$), existential restriction ($\exists$), and top ($\top$), and $\mathcal{EL}^{(\neg)}$ is the extension of $\mathcal{EL}$ that allows for atomic negation. Note that we allow negation in consequences of action post-conditions although $\mathcal{EL}$ does not provide negation. The reason is that we believe that actions without negated post-conditions are too restrictive to be useful.

As already mentioned, $\mathcal{EL}$ plays an important role in modelling life science ontologies [BC07, Spa01]. Many of such ontologies consist of acyclic TBoxes and can thus be used together with the action formalism $\mathfrak{A}_1$. Most importantly, standard reasoning tasks such as ABox consequence or subsumption are tractable in $\mathcal{EL}$ [Baa03, Bra04, BBL05].

Our results show that, in general, tractability does not transfer from ABox consequence to projection. Even for $\mathcal{EL}$ without TBoxes, the latter problem is shown to be co-NP-hard in Section 4.1.1. We remark that co-NP-hardness does not follow from hardness results for propositional action formalisms since $\mathcal{EL}$ does not have disjunction and negation occurs nowhere except in post-conditions. We also prove a matching co-NP upper bound for $\mathcal{EL}^{(\neg)}$. Moreover, if we allow for acyclic TBoxes, we show in Section 4.1.2 that the projection problem in $\mathcal{EL}$ is PSPACE-hard and thus not easier than in $\mathcal{ALC}$. This shows that the reduction of projection to ABox consequence from Section 3.2.1, although developed for expressive DLs, is optimal with respect to complexity of reasoning even for the lightweight $\mathcal{EL}$ if acyclic TBoxes are allowed. An intuitive explanation for projection being a hard problem in the lightweight

$\mathcal{EL}$ is that, even with *unconditional* action post-conditions, one can "simulate" disjunction. The source of intractability turn out to be existential restrictions in the initial ABox together with negated assertions in post-conditions.

In hardness proofs, we use actions of a restricted form: only atomic actions are admitted, the sets of pre-conditions and occlusions are empty, and post-conditions are unconditional. Recall that we call such actions *updates* and write $\mathcal{U} = \{\psi_1, \ldots, \psi_n\}$ instead of $\mathcal{U} = (\emptyset, \emptyset, \{\top(a)/\psi_1, \ldots, \top(a)/\psi_n\})$. For an interpretation $\mathcal{I}$, we will use $\mathcal{I}^{\mathcal{U}}_{\mathcal{T}}$ to denote the unique interpretation $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\mathcal{U}} \mathcal{I}'$. If $\mathcal{T}$ is empty, we just write $\mathcal{I}^{\mathcal{U}}$ instead.

4.1.1 Projection in $\mathcal{EL}$ with empty TBoxes

In [Sch93], Schaerf proves that instance checking (ABox consequence) in $\mathcal{EL}^{(\neg)}$ w.r.t. empty TBoxes is co-NP-hard regarding data complexity (ABox size). He uses a reduction from a variant of SAT that he calls 2+2-SAT. We use a similar reduction.

A *2+2 clause* is of the form $(p_1 \vee p_2 \vee \neg n_1 \vee \neg n_2)$, where each of p_1, p_2, n_1, n_2 is a propositional variable or a truth constant $\top$, $\bot$. A *2+2 formula* is a finite conjunction of 2+2 clauses. Now, 2+2-SAT is the problem of deciding whether a given 2+2 formula is satisfiable. It is shown in [Sch93] that 2+2-SAT is NP-complete.

Let $\varphi = c_1 \wedge \cdots \wedge c_n$ be a 2+2-formula in m propositional variables $q_1, \ldots, q_m$. Let $c_i = p_{i,1} \vee p_{i,2} \vee \neg n_{i,1} \vee \neg n_{i,2}$ for all $1 \leq i \leq n$. We use f, x, y, the propositional variables $q_1, \ldots, q_m$, and the clauses $c_1, \ldots, c_n$ as individual names. Moreover, we introduce individual names $q_\top$ and $q_\bot$ corresponding to the truth constants $\top$ and $\bot$. Define the ABox $\mathcal{A}_\varphi$ as follows, where c, p_1, p_2, n_1, n_2, and t are role names:

$$\begin{aligned}\mathcal{A}_\varphi := &\{c(f, c_1), \ldots, c(f, c_n)\} \\ &\cup \textstyle\bigcup_{1 \leq i \leq n} \{p_1(c_i, p_{i,1}), p_2(c_i, p_{i,2}), n_1(c_i, n_{i,1}), n_2(c_i, n_{i,2})\} \\ &\cup \{(A \sqcap \overline{A})(x), A(y), t(q_\top, y), t(q_\bot, x)\} \\ &\cup \{\exists t.A(q_1), \ldots, \exists t.A(q_m)\}\end{aligned}$$

Let $\mathcal{U} = \{\neg A(x)\}$ and $\psi = C_\varphi(f)$, where

$$C_\varphi := \exists c.(\exists p_1.\exists t.\overline{A} \sqcap \exists p_2.\exists t.\overline{A} \sqcap \exists n_1.\exists t.A \sqcap \exists n_2.\exists t.A)$$

Intuitively, for models $\mathcal{I}$ of $\mathcal{A}_\varphi$, interpretations $\mathcal{J} = \mathcal{I}^{\mathcal{U}}$ are supposed to represent truth assignments as follows: for $z \in \{q_\top, q_\bot, q_1, \ldots, q_m\}$, the truth value of z is $\top$ if $z^{\mathcal{J}} \in (\exists t.A)^{\mathcal{J}}$ and $\bot$ if $z^{\mathcal{J}} \in (\exists t.\overline{A})^{\mathcal{J}}$. Observe that, due to the t-links from $q_\top$ and $q_\bot$ to x and y and since we adopt the UNA, $q_\top^{\mathcal{I}} \in (\exists t.A)^{\mathcal{I}}$ and $q_\bot^{\mathcal{I}} \in (\exists t.\overline{A})^{\mathcal{I}}$ as required. It can be seen as follows that every $z \in \{q_1, \ldots, q_m\}$ satisfies $\exists t.A$ or $\exists t.\overline{A}$. Since $\{\exists t.A(z), A(x)\} \subseteq \mathcal{A}_\varphi$, we may distinguish two cases: in every model $\mathcal{I}$ of $\mathcal{A}_\varphi$, we have (i) $(z^{\mathcal{I}}, x^{\mathcal{I}}) \in t^{\mathcal{I}}$ or (ii) there is a $d \neq x^{\mathcal{I}}$ such that $(z^{\mathcal{I}}, x) \in t^{\mathcal{I}}$ and $x \in A^{\mathcal{I}}$. Let $\mathcal{J} = \mathcal{I}^{\mathcal{U}}$. In Case (i), we have $z^{\mathcal{I}} \in (\exists t.\overline{A})^{\mathcal{J}}$. And in Case (ii), we have $z^{\mathcal{I}} \in (\exists t.A)^{\mathcal{J}}$.

There is one complication: we may have ill-behaved models of $\mathcal{A}_\varphi$ in which there is a $z \in \{q_\top, q_\bot, q_1, \ldots, q_m\}$ that satisfies *both* $\exists t.A$ and $\exists t.\overline{A}$. Clearly, such interpretations do not correspond to truth assignments. To solve this problem, consider the query ψ. It expresses that the formula is false, i.e., that there is a clause that is not satisfied because all its positive and negative literals are violated. What we want to achieve is that $\mathcal{A}^{\mathcal{U}}_\varphi \not\models \psi$ iff ψ is satisfiable. The problem with ill-behaved models is that the counter-model showing

$\mathcal{A}_\varphi^{\mathcal{U}} \not\models \psi$ may be ill-behaved, and thus we cannot guarantee satisfiability of ψ. To solve this, we extend the ABox $\mathcal{A}_\varphi$ such that all ill-behaved models of the resulting ABox satisfy the query ψ. Let $d_\top, d_\bot, d_1, \ldots, d_m$ be additional individual names and define

$$\begin{aligned}\mathcal{A}'_\varphi := \quad & \{c(f, d_\top), c(f, d_\bot), c(f, d_1), \ldots, c(f, d_m)\} \\ & \cup \textstyle\bigcup_{i \in \{\top, \bot, 1, \ldots, m\}} \{p_1(d_i, q_i), p_2(d_i, q_i), n_1(d_i, q_i), n_2(d_i, q_i)\}\end{aligned}$$

We can show the following.

Lemma 4.1.1. *φ is satisfiable iff ψ is not a consequence of applying $\mathcal{U}$ in $\mathcal{A}_\varphi \cup \mathcal{A}'_\varphi$.*

Proof. "⇒" Let φ be satisfiable. Then there exists a valuation $v : \{q_0 \ldots q_m\} \to \{\top, \bot\}$ such that $v(\varphi) = \top$. Additionally, we set $v(q_\top) := \top$ and $v(q_\bot) := \bot$. We will use v to construct a model $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ of $\mathcal{A}_\varphi \cup \mathcal{A}'_\varphi$ such that $\mathcal{I}^\mathcal{U} \models \neg\psi$.

Let $\Delta^\mathcal{I} \supseteq \{\mathsf{f}, \mathsf{c}_1, \ldots \mathsf{c}_n, \mathsf{q}_\top, \mathsf{q}_\bot, \mathsf{q}_1, \ldots \mathsf{q}_m, \mathsf{d}_\top, \mathsf{d}_\bot, \mathsf{d}_1, \ldots \mathsf{d}_m, \mathsf{x}, \mathsf{y}\}$. Moreover, we set: $f^\mathcal{I} := \mathsf{f}$, $c_i^\mathcal{I} := \mathsf{c}_i$ for $i \in \{1, \ldots, n\}$, $q_i^\mathcal{I} := \mathsf{q}_i$ and $d_i^\mathcal{I} := \mathsf{d}_i$ for $i \in \{\top, \bot, 1, \ldots, m\}$, $x^\mathcal{I} := \mathsf{x}$ and $y^\mathcal{I} := \mathsf{y}$. Finally, we define:

$$\begin{aligned}
A^\mathcal{I} &:= \{\mathsf{x}, \mathsf{y}\} \\
(\overline{A})^\mathcal{I} &:= \{\mathsf{x}\} \\
c^\mathcal{I} &:= \{(\mathsf{f}, \mathsf{c}_1), \ldots, (\mathsf{f}, \mathsf{c}_n), (\mathsf{f}, \mathsf{d}_\top), (\mathsf{f}, \mathsf{d}_\bot), (\mathsf{f}, \mathsf{d}_1), \ldots, (\mathsf{f}, \mathsf{d}_m)\}\} \\
p_j^\mathcal{I} &:= \{(\mathsf{c_i}, \mathsf{q}_j) \mid q_j \text{ is the } j^{\text{th}} \text{ positive variable of } c_i\} \cup \{(\mathsf{d}_i, \mathsf{q}_i) \mid i \in \{\top, \bot, 1, \ldots, m\}\} \\
n_j^\mathcal{I} &:= \{(\mathsf{c_i}, \mathsf{q}_j) \mid q_j \text{ is the } j^{\text{th}} \text{ negative variable of } c_i\} \cup \{(\mathsf{d}_i, \mathsf{q}_i) \mid i \in \{\top, \bot, 1, \ldots, m\}\} \\
t^\mathcal{I} &:= \{(\mathsf{q}_\bot, \mathsf{x}), (\mathsf{q}_\top, \mathsf{y})\} \cup \{(\mathsf{q}_i, \mathsf{x}) \mid v(q_i) = \bot\} \cup \{(\mathsf{q}_i, \mathsf{y}) \mid v(q_i) = \top\}
\end{aligned}$$

Here, $j \in \{1, 2\}$. Obviously, $\mathcal{I}$ is a model of $\mathcal{A}_\varphi \cup \mathcal{A}'_\varphi$. The interpretation $\mathcal{I}^\mathcal{U}$ disagrees with $\mathcal{I}$ only in the extension of A; namely $A^{\mathcal{I}^\mathcal{U}} = \{\mathsf{y}\}$. Since $A^{\mathcal{I}^\mathcal{U}} = A^\mathcal{I} \setminus \{\mathsf{x}\}$, and $t^{\mathcal{I}^\mathcal{U}} = t^\mathcal{I}$, we obtain the following for $i \in \{\top, \bot, 1, \ldots, m\}$:

$$\mathcal{I}^\mathcal{U} \models (\forall t.\neg\overline{A})(q_i) \text{ iff } v(q_i) = \top \quad \text{and} \quad \mathcal{I}^\mathcal{U} \models (\forall t.\neg A)(q_i) \text{ iff } v(q_i) = \bot$$

Moreover, since $\neg C_\varphi \equiv \forall c.(\forall p_1.\forall t.\neg\overline{A} \sqcup \forall p_2.\forall t.\neg\overline{A} \sqcup \forall n_1.\forall t.\neg A \sqcup \forall n_2.\forall t.\neg A)$, and since v evaluates φ to true, we have that $\mathcal{I}^\mathcal{U} \models \neg C_\varphi(f)$.

"⇐"Let $\mathcal{I}$ be an interpretation such that $\mathcal{I} \models \mathcal{A}_\varphi \cup \mathcal{A}'_\varphi$ and $\mathcal{I}^\mathcal{U} \models \neg C_\varphi(f)$. Let $i \in \{1, \ldots, m\}$. Since $\{\exists t.A(q_i), A(x)\} \subseteq \mathcal{A}_\varphi$, we may distinguish two cases: (i) If $(q_i^\mathcal{I}, x^\mathcal{I}) \in t^\mathcal{I}$ then $q_i^\mathcal{I} \in (\exists t.\overline{A})^\mathcal{J}$; and (ii) if there is a $d \neq x^\mathcal{I}$ such that $(q_i^\mathcal{I}, x) \in t^\mathcal{I}$ and $x \in A^\mathcal{I}$, then $q_i^\mathcal{I} \in (\exists t.A)^\mathcal{J}$. Thus, we obtain that $\mathcal{I}^\mathcal{U} \models (\exists t.A \sqcup \exists t.\overline{A})(q_i)$ for all $1 \leq i \leq m$. Since the update $\mathcal{U}$ affects only the truth value of $A(x)$, we obtain that $\mathcal{I}^\mathcal{U} \models \mathcal{A}''_\varphi \cup \mathcal{A}'_\varphi \cup \mathcal{U}$, where the $\mathcal{ELU}$ ABox $\mathcal{A}''_\varphi$ is defined with:

$$\begin{aligned}\mathcal{A}''_\varphi := & \{c(f, c_1), \ldots, c(f, c_n)\} \\ & \cup \textstyle\bigcup_{1 \leq i \leq n} \{p_1(c_i, p_{i,1}), p_2(c_i, p_{i,2}), n_1(c_i, n_{i,1}), n_2(c_i, n_{i,2})\} \\ & \cup \{\overline{A}(x), A(y), t(q_\top, y), t(q_\bot, x)\} \\ & \cup \{(\exists t.A \sqcup \exists t.\overline{A})(q_1), \ldots, (\exists t.A \sqcup \exists t.\overline{A})(q_m)\}\end{aligned}$$

Moreover, since $\mathcal{I}^\mathcal{U} \models \mathcal{A}'_\varphi \cup \{\neg C_\varphi(f)\}$ we obtain that $\mathcal{I}^\mathcal{U}$ satisfies either $\exists t.A(q_i)$ or $\exists t.\overline{A}(q_i)$, but not both. To see this, assume that for an l, $1 \leq l \leq n$, it holds that $\mathcal{I}^\mathcal{U} \models$

$(\exists t.A \sqcap \exists t.\overline{A})(q_l)$. But then $\mathcal{I}^{\mathcal{U}} \models \mathcal{A}'_\varphi$ implies that $\mathcal{I}^{\mathcal{U}} \models (\exists p_1.\exists t.\overline{A} \sqcap \exists p_2.\exists t.\overline{A} \sqcap \exists n_1.\exists t.A \sqcap \forall n_2.\exists t.A)(d_l)$. Since $\mathcal{I}^{\mathcal{U}} \models c(f, d_l)$, we obtain $\mathcal{I}^{\mathcal{U}} \models C_\varphi(f)$, which is a contradiction.

Thus, the valuation $v : \{q_1, \ldots, q_n\} \to \{\top, \bot\}$ defined with:

- $v(q_i) = \top$ iff $\mathcal{I}^{\mathcal{U}} \models \exists t.A(q_i)$; and
- $v(q_i) = \bot$ iff $\mathcal{I}^{\mathcal{U}} \models \exists t.\overline{A}(q_i)$.

is well defined. $\mathcal{I}^{\mathcal{U}} \models \mathcal{A}''_\varphi \cup \{\neg C_\varphi(f)\}$ implies that $v(\varphi) = \top$, i.e. φ is satisfiable. ❑

Theorem 4.1.2. *Projection in $\mathcal{EL}$ with empty TBoxes is co-*NP*-hard, even for a singleton update.*

Upper Bounds

We will use the bounded model property of counter-models for projection in order to show that projection in $\mathcal{EL}^{(\neg)}$ is in co-NP. More precisely, we will show that, if an assertion φ is not a consequence of applying an action α in an ABox $\mathcal{A}$, all formulated in $\mathcal{EL}^{(\neg)}$, then there exist *counter-models* $\mathcal{I}$, $\mathcal{I}'$ against $\mathcal{A}^\alpha \models \psi$ whose size is linearly bounded by $|\mathcal{A}| + |\alpha|$, i.e. there exist interpretations $\mathcal{I}$, $\mathcal{I}'$ such that:

- $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_\alpha \mathcal{I}'$ and $\mathcal{I}' \models \neg\varphi$;
- $|\Delta^{\mathcal{I}}| = |\Delta^{\mathcal{I}'}|$ is bounded by $|\mathcal{A}| + |\alpha|$.

Since $\mathcal{I}$ and $\mathcal{I}'$ respect UNA on individual names, we need to assume that $\mathcal{I}$ and $\mathcal{I}'$ interpret only those individual names that appear in $\mathcal{A}$, α, and φ.

We start by defining the notion of a filtration of an interpretation w.r.t. an $\mathcal{EL}^{(\neg)}$-concept. We first introduce a few notions.

Definition 4.1.3 (Role depth, Top Existentials). The *role depth* (written $\mathsf{rd}(E)$) of concepts is defined inductively as follows:

$$\begin{array}{rcll} \mathsf{rd}(A) & = & \mathsf{rd}(\neg A) := 0 & \text{for } A \text{ a concept name} \\ \mathsf{rd}(C \sqcap D) & := & max\{\mathsf{rd}(C), \mathsf{rd}(D)\} & \\ \mathsf{rd}(\exists R.C) & := & \mathsf{rd}(C) + 1 & \end{array}$$

The set of *top existentials* (written $\mathsf{topex}(E)$) of concepts is defined inductively as follows:

$$\begin{array}{rcll} \mathsf{topex}(A) & = & \mathsf{topex}(\neg A) := \emptyset & \text{for } A \text{ a concept name} \\ \mathsf{topex}(C \sqcap D) & := & \mathsf{topex}(C) \cup \mathsf{topex}(D) & \\ \mathsf{topex}(\exists R.C) & := & \{\exists R.C\} & \end{array}$$

△

Definition 4.1.4 (Filtration w.r.t. a concept). Let C_0 be an $\mathcal{EL}^{(\neg)}$-concept, $\mathcal{I}$ be an interpretation and d_0 be an element of $\Delta^{\mathcal{I}}$ such that $d_0 \in C_0^{\mathcal{I}}$. A *node* in $\mathcal{I}$ is a pair (d, C), with $d \in \Delta^{\mathcal{I}}$ and $C \in \mathsf{sub}(C_0)$ such that $d \in C^{\mathcal{I}}$. An *edge* in $\mathcal{I}$ is a triple (d, e, r) such that $(d, e) \in r^{\mathcal{I}}$. An interpretation $\mathcal{J}$ is called a *filtration* of $\mathcal{I}$ w.r.t. C_0 and d_0 if it can be constructed from $\mathcal{I}$ in the following way. For each node $(d, \exists r.D)$ in $\mathcal{I}$, fix a witness $\tau(d, \exists r.D) \in \Delta^{\mathcal{I}}$ such that $(d, \tau(d, \exists r.D)) \in r^{\mathcal{I}}$ and $\tau(d, \exists r.D) \in D^{\mathcal{I}}$. Determine a set of *selected nodes* and a set of *selected edges* in $\mathsf{rd}(C_0) + 1$ rounds as follows:

- In round 0, select (d_0, C_0) (no edges are selected).
- In round i with $0 < i \leq \mathsf{rd}(C_0)$, do the following: for all nodes (d, C) that have been selected in round $i-1$ and for every $\exists r.D \in \mathsf{topex}(C)$ select the node $(\tau(d, \exists r.D), D)$ and the edge $(d, \tau(d, \exists r.D), r)$.

Define the interpretation $\mathcal{J}$ as the restriction of $\mathcal{I}$ to the set of selected nodes and edges, i.e.,

$$\begin{aligned} \Delta^{\mathcal{J}} &:= \{d \in \Delta^{\mathcal{I}} \mid (d, C) \text{ is selected for some } C \in \mathsf{sub}(C_0)\}, \\ A^{\mathcal{J}} &:= A^{\mathcal{I}} \cap \Delta^{\mathcal{J}} \text{ for all } A \in \mathsf{N_C}, \\ r^{\mathcal{J}} &:= \{(d, e) \in r^{\mathcal{I}} \mid (d, e, r) \text{ is selected}\} \text{ for all } r \in \mathsf{N_R}. \end{aligned}$$

The object d_0 is called the *root* of $\mathcal{J}$. △

It is easy to show by induction on the role depth of a concept C_0, that a filtration $\mathcal{J}$ of a model of the concept C_0 is also a model of C_0. Moreover, additionally to the root, $\Delta^{\mathcal{J}}$ contains at most one element for every occurrence of a subconcept of the form $\exists r.D$ in C_0. Thus, $|\Delta^{\mathcal{J}}|$ is bounded by the size of C_0.

Lemma 4.1.5. *If $\mathcal{I}$ is a model of C_0 and $\mathcal{J}$ is a filtration of $\mathcal{I}$ w.r.t. C_0 and d_0, then $\mathcal{J}$ is also a model of C_0 and $|\Delta^{\mathcal{J}}|$ is bounded by $|C_0|$.*

Lemma 4.1.6. *Let $\mathcal{A}$ be an ABox, $\alpha = \alpha_1, \ldots \alpha_n$ with $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ a composite action and φ_0 an assertion, all formulated in $\mathcal{EL}^{(\neg)}$. Let Ind be the set of individual names which appear in α, $\mathcal{A}$, and φ_0. If φ_0 is not a consequence of applying α in $\mathcal{A}$, then there exist interpretations $\mathcal{J}_0, \mathcal{J}_1, \ldots, \mathcal{J}_n$ interpreting only individuals from Ind such that $\mathcal{J}_0' \models \mathcal{A}$, $\mathcal{J}_{i-1} \Rightarrow_{\alpha_i} \mathcal{J}_i$ for $1 \leq i \leq n$, $\mathcal{J}_n \not\models \varphi_0$, and $|\Delta^{\mathcal{J}_i}|$ is not greater than $|\mathcal{A}| + |\mathsf{post}_1| + \cdots + |\mathsf{post}_n|$.*

Proof. Since φ_0 is not a consequence of applying α in $\mathcal{A}$, there exist interpretations $\mathcal{I}_0$, $\mathcal{I}_1$, $\ldots$, $\mathcal{I}_n$ such that $\mathcal{I}_0 \models \mathcal{A}$, $\mathcal{I}_{i-1} \Rightarrow_{\alpha_i} \mathcal{I}_i$ for $1 \leq i \leq n$, and $\mathcal{I}_n \not\models \varphi_0$. Let $\mathcal{I}_i^{C(a)}$ be the filtration of $\mathcal{I}_i$ w.r.t. C and $a^{\mathcal{I}_i}$. We define Δ to be the smallest set such that:

- $a^{\mathcal{I}_0}$ is in Δ, for all individuals a from Ind;
- $\Delta_{\mathcal{I}_0^{C(a)}} \subseteq \Delta$ for all $C(a) \in \mathcal{A}$;
- $\Delta_{\mathcal{I}_i^{D(b)}} \subseteq \Delta$ for all $D(b)$ and $0 \leq i < n$ such that $D(b)/\psi \in \mathsf{post}_i$ and $\mathcal{I}_{i-1} \models D(b)$.

Obviously, $|\Delta|$ is bounded by $|\mathcal{A}| + |\mathsf{post}_1| + \cdots + |\mathsf{post}_n|$. Moreover, let us define interpretations $\mathcal{J}_i$ for $i \leq n$ as follows:

$$\begin{aligned} \Delta^{\mathcal{J}_i} &:= \Delta, \\ A^{\mathcal{J}_i} &:= A^{\mathcal{I}_i} \cap \Delta \text{ for all } A \in \mathsf{N_C}, \\ r^{\mathcal{J}_i} &:= r^{\mathcal{I}_i} \cap (\Delta \times \Delta) \text{ for all } r \in \mathsf{N_R}, \\ a^{\mathcal{J}_i} &:= a^{\mathcal{I}_i} \text{ for all } a \in \mathsf{Ind}. \end{aligned}$$

From the properties of filtrations and the definition of Δ and $\mathcal{J}_i$, we obtain as an easy consequence that $\mathcal{J}_0 \models \mathcal{A}$ and $\mathcal{J}_{i-1} \models \{\varphi \mid \varphi/\psi \in \mathsf{post}_i, \mathcal{I}_{i-1} \models \varphi\}$, $1 \leq i \leq n$. Moreover, it is not difficult to see that for $\mathcal{EL}^{(\neg)}$ assertions that are not satisfied in $\mathcal{I}_i$, it also holds that they are not satisfied in $\mathcal{J}_i$ (obtained by restricting the domain of $\mathcal{I}_i$ to Δ). Thus, we have that $\mathcal{I}_{i-1} \not\models \varphi$ implies $\mathcal{J}_{i-1} \not\models \varphi$ for $\varphi/\psi \in \mathsf{post}_i$ and $1 \leq i \leq n$ and $\mathcal{J}_n \not\models \varphi_0$. Summing up, we have shown that $\mathcal{J}_{i-1} \Rightarrow_{\alpha_i} \mathcal{J}_i$ for $1 \leq i \leq n$. ❏

The previous lemma implies that non-projection in $\mathcal{EL}^{(\neg)}$ without TBoxes is in NP: Let $\mathcal{A}$ be an ABox, α a composite action, and φ_0 an assertion, all formulated in $\mathcal{EL}^{(\neg)}$. Guessing interpretations $\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_n$ such that and $|\Delta^{\mathcal{I}_0}| \leq |\mathcal{A}| + |\alpha|$ [1], and checking if $\mathcal{I}_0 \models \mathcal{A}$, $\mathcal{I}_{i-1} \Rightarrow_{\alpha_i} \mathcal{I}_i$ for $1 \leq i \leq n$ and $\mathcal{I}_n \models \neg\varphi$ can be done in a non-deterministic time which is polynomial in $|\mathcal{A}| + |\alpha|$. Thus we obtain the following:

Lemma 4.1.7. *Projection in $\mathcal{EL}^{(\neg)}$ with empty TBoxes is in co-*NP*.*

The previous Lemma together with Theorem 4.1.2 gives us the following result:

Theorem 4.1.8. *Projection in $\mathcal{EL}$ and $\mathcal{EL}^{(\neg)}$ with empty TBoxes is co-*NP*-complete.*

4.1.2 Projection in $\mathcal{EL}$ with acyclic TBoxes

In this section we show that, if we allow for acyclic TBoxes, projection in $\mathcal{EL}$ becomes PSPACE-hard. This result holds even if we compute projection w.r.t. an update, e.g. an atomic action with *empty* pre-conditions and occlusions, and only *unconditional* post-conditions.

To this end, we reduce validity of quantified Boolean formulas (QBF) to projection in $\mathcal{EL}$.

Definition 4.1.9 (Quantified Boolean Formula (QBF)). A *quantified Boolean formula (QBF)* is of the form

$$\phi = Q_1 p_1 \ldots Q_n p_n . \varphi(p_1, \ldots, p_n),$$

where $Q_i \in \{\forall, \exists\}$, and $\varphi(p_1, \ldots, p_n)$ is a propositional formula using only the propositional variables $p_1, \ldots, p_n$.

Validity of QBF is defined via induction on the length of quantifier prefix. For a propositional formula φ, we define $\varphi[p/v]$, $v \in \{\top, \bot\}$, to be the propositional formula obtained by replacing p with v in φ. The QBF ϕ is *valid* iff:

1. $Q_1 = \forall$: $Q_2 p_2 \ldots Q_n p_n . \varphi[p_1/\top]$ and $Q_2 p_2 \ldots Q_n p_n . \varphi[p_1/\bot]$ is valid;
2. $Q_1 = \exists$: $Q_2 p_2 \ldots Q_n p_n . \varphi[p_1/\top]$ or $Q_2 p_2 \ldots Q_n p_n . \varphi[p_1/\bot]$ is valid.

△

It is known that validity of QBF is a PSPACE-hard problem, even if the matrix $\varphi(p_1, \ldots, p_n)$ is in CNF [SM73]. To every QBF we can assign a "graphical" representation, called quantifier tree.

Definition 4.1.10 (Quantifier Tree). A *quantifier tree* of the QBF $\phi = Q_1 p_1 \ldots Q_n p_n . \varphi(p_1, \ldots, p_n)$ is a tree of branching factor at most 2 such that:

- each level of the tree (except the leaves) corresponds to one of the quantifiers of ϕ;
- in $\forall p_i$-levels, each node has two successors, one for $p_i = \top$, and one for $p_i = \bot$;
- in $\exists p_i$-levels, each node has one successor, either for $p_i = \top$, or for $p_i = \bot$.

Thus, every branch of a quantifier tree corresponds to a truth assignment to the variables $p_1, \ldots, p_n$. A tree is *valid* iff on each branch of the tree, the propositional formula φ evaluates to true ($\top$).

△

[1] Obviously, it suffices to guess only extensions of concept and role names appearing in $\mathcal{A}$, α, and φ_0

It is not difficult to show that a QBF ϕ is valid iff it has a valid quantifier tree.

Let $\phi = Q_1 p_1 \dots Q_n p_n.\varphi$ where $Q_i \in \{\forall, \exists\}$, be a quantified Boolean formula with φ in CNF. We define an $\mathcal{EL}$ acyclic TBox $\mathcal{T}_\phi$, ABox $\mathcal{A}_\phi$, a query ψ_ϕ, and an update $\mathcal{U}$ such that $\mathcal{T}_\phi, \mathcal{A}_\phi^{\mathcal{U}} \not\models \psi_\phi$ iff ϕ is valid. As in Section 4.1.1, we call models $\mathcal{I}$ and $\mathcal{J}$ of $\mathcal{T}$ *counter-models* against $\mathcal{T}_\varphi, \mathcal{A}_\varphi^{\mathcal{U}} \models \psi_\phi$ iff $\mathcal{I} \models \mathcal{A}_\varphi$, $\mathcal{I} \Rightarrow_{\mathcal{U}}^{\mathcal{T}} \mathcal{J}$, and $\mathcal{J} \not\models \psi_\phi$. The general idea of the reduction is to achieve that, if $\mathcal{I}, \mathcal{J}$ are such counter-models, then $\mathcal{J}$ encodes a valid quantifier tree for ϕ. The purpose of the reduction TBox $\mathcal{T}_\varphi$ is to establish a tree structure in $\mathcal{I}$ and $\mathcal{J}$ that is the core of this encoding. We use a role name r to represent the edges of the quantifier tree, and the concept names $L_0, \dots, L_n$ to identify its n levels. The truth values of the variables $p_1, \dots, p_n$ are represented via the concept names $P_1, \dots, P_n$ (indicating truth) and $\overline{P}_1, \dots, \overline{P}_n$ (indicating falsity). We also use role names t, s and concept names $A_1, \dots, A_n$, $\overline{A}_1, \dots, \overline{A}_n$, to be explained later.

We first define $\mathcal{T}_\phi$ as:

$$\begin{aligned}
\mathcal{T}_\phi \;:=\; & \{L_{i-1} \doteq \bigsqcap_{1 \le j < i} \exists t.A_j \sqcap \exists r.(P_i \sqcap L_i) \sqcap \exists r.(\overline{P}_i \sqcap L_i) \mid 1 \le i \le n, Q_i = \forall\} \\
& \cup \{L_{i-1} \doteq \bigsqcap_{1 \le j < i} \exists t.A_j \sqcap \exists r.L_i \mid 1 \le i \le n, Q_i = \exists\} \\
& \cup \{L_n \doteq \bigsqcap_{1 \le j \le n} \exists t.A_j\}
\end{aligned}$$

The reduction ABox $\mathcal{A}_\varphi$ will include an assertion $L_0(a)$. The TBox $\mathcal{T}_\varphi$ thus establishes a binary tree of depth n rooted at $a^{\mathcal{I}}$ in $\mathcal{I}$ with the right number of successors at each level and with the concept names P_i, $\overline{P}_i$ set appropriately in levels L_i with $Q_i = \forall$. Since none of the L_i, P_i, $\overline{P}_i$, and r will occur in the update, $a^{\mathcal{J}}$ is a root of the same tree in $\mathcal{J}$. To make this tree a valid quantifier tree for ϕ, it remains to ensure that the tree in $\mathcal{J}$ satisfies the following:

(a) On every branch of the tree, every variable is interpreted as true or false (not yet guaranteed since both P_i and $\overline{P}_i$ may be false in a level L_i with $Q_i = \exists$).

(b) On no branch, a variable is interpreted as both true and false.

(c) Every branch describes a truth assignment that satisfies φ.

To enforce (a)-(c), we introduce a second representation of truth values, which is used as the main such representation from now on: $\exists t.A_j$ indicates truth of p_j and $\exists t.\overline{A}_j$ indicates falsity. In contrast to the representation via P_j and $\overline{P}_j$, in which the truth value of p_j is only stored at level j, the representation via $\exists t.A_j$ and $\exists t.\overline{A}_j$ stores the truth value of p_j at any level $i \ge j$. In particular, this means that the (unique) leaf of a branch stores the whole truth assignment associated with the branch, and thus we can ensure (c) locally at the leaves.

We start using the new representation by enforcing a central property:

$$\text{If } d \in L_j^{\mathcal{I}}, \text{ then } d \in (\exists t.A_i)^{\mathcal{J}} \text{ or } (\exists t.\overline{A}_i)^{\mathcal{J}} \text{ is true, for } 1 \le i \le j \le n. \qquad (*)$$

To establish $(*)$, we exploit the same effect as in the co-NP-hardness proof in Section 4.1.1. More precisely, we use (i) the concepts $\bigsqcap_{1 \le j < i} \exists t.A_j$ in $\mathcal{T}_\varphi$, (ii) assertions $(A_1 \sqcap \overline{A}_1)(b_1), \dots, (A_n \sqcap \overline{A}_n)(b_n)$ which will be in $\mathcal{A}_\phi$, and (iii) the reduction update, which is defined as

$$\mathcal{U} := \{\neg A_1(b_1), \dots, \neg A_n(b_n)\}.$$

Due to the $\bigsqcap_{1\leq j<i} \exists t.A_j$ concepts, each $d \in L_j^{\mathcal{I}}$ satisfies $\exists t.A_j$ in $\mathcal{I}$. The choice in $(*)$ then corresponds to whether or not the r-successor stipulated by this concept is b_j.

Obviously, $(*)$ guarantees (a). The definition of $\mathcal{U}$ explains why we cannot use the second representation of truth assignments already in $\mathcal{T}_\phi$. Namely, $\mathcal{T}_\phi$ is used together with the assertion $L_0(a) \in \mathcal{A}_\phi$, and thus talks about $\mathcal{I}$. Since $A_1, \dots, A_n$ occur negated in $\mathcal{U}$, truth of concepts $\exists t.A_i$ and $\exists t.\overline{A}_i$ in $\mathcal{I}$ may be destroyed when moving with $\mathcal{U}$ from $\mathcal{I}$ to $\mathcal{J}$.

We proceed by ensuring that every node $d \in L_i^{\mathcal{I}}$ satisfies the following three properties:

1. if d is in $P_i^{\mathcal{J}}$, then it is in $(\exists t.A_i)^{\mathcal{J}}$, and likewise for $\overline{P}_i$ and $\exists t.\overline{A}_i$;
2. d is in at most one of $(\exists t.A_j)^{\mathcal{J}}$ and $(\exists t.\overline{A}_j)^{\mathcal{J}}$, for $1 \leq j \leq i$.
3. if d is in $(\exists t.A_j)^{\mathcal{J}}$ with $1 \leq j \leq i$, then so are all its r-successors; and likewise for $\exists t.\overline{A}_j$;

Note that Point 1 links the two representations of truth values, Point 2 addresses (b), and Point 3 ensures that truth values stored via the second representation are pushed down towards the leafs.

We define a set of concepts $\mathcal{C}$ such that, to enforce Points 1 to 3, it suffices to ensure that all concepts in $\mathcal{C}$ are false at the root of the quantifier tree in $\mathcal{J}$:

$$\begin{aligned}\mathcal{C} \;:=\; & \{\exists r^i.(P_i \sqcap \exists t.\overline{A}_i), \exists r^i.(\overline{P}_i \sqcap \exists t.A_i) \mid 1 \leq i \leq n\} \cup \\ & \{\exists r^i.(\exists t.A_j \sqcap \exists r.\exists t.\overline{A}_j), \exists r^i.(\exists t.\overline{A}_j \sqcap \exists r.\exists t.A_j) \mid 1 \leq j \leq i < n\} \cup \\ & \{\exists r^i.(\exists t.A_j \sqcap \exists t.\overline{A}_j) \mid 1 \leq j \leq i \leq n\}\end{aligned}$$

Note that the i-th line in the definition of $\mathcal{C}$ corresponds to Point i above. Also note that the first two lines of $\mathcal{C}$ rely on $(*)$ to have the desired effect.

Before we describe how $\mathcal{C}$ can be incorporated into the reduction, let us describe how to ensure that, at every leaf, the formula φ evaluates to true. The idea is to come up with another set of concepts $\mathcal{D}$ that are made false at the root of the quantifier tree in $\mathcal{J}$. We use $\overline{\varphi}$ to denote the dual of φ, i.e. the formula obtained from ϑ by swapping $\vee$ and $\wedge$ and p_i and $\neg p_i$, for $1 \leq i \leq n$. Obviously, $\overline{\varphi}$ is equivalent to $\neg\varphi$, $\overline{\varphi}$ is of the same length as φ, and $\overline{\varphi}$ is in DNF. Let $\overline{\varphi} = \overline{\varphi}_1 \vee \dots \vee \overline{\varphi}_m$, where the $\overline{\varphi}_i$ are conjunctions of literals. Now $\mathcal{D}$ consists of the concepts $C_{\overline{\varphi}_i}$, for $1 \leq i \leq m$, where:

$$C_{\overline{\varphi}_i} := \overline{\varphi}_i[p_j/\exists t.A_j, \neg p_j/\exists t.\overline{A}_j, \wedge/\sqcap]$$

Clearly, $\mathcal{D}$ is as required. If $\mathcal{EL}$ would include disjunction, we could now easily put $\mathcal{C}$ and $\mathcal{D}$ to work and thus finish the reduction by setting $\psi_\phi := \bigsqcup_{C \in \mathcal{C} \cup \mathcal{D}} C(a)$, where a denotes the root of the quantifier tree. Since $\mathcal{J}$ is a part of a counter-model and thus violates ψ_ϕ, this has the desired effect that all concepts in $\mathcal{C} \cup \mathcal{D}$ are false at a. Alas, there is no disjunction in $\mathcal{EL}$ and we need to invest more work to employ $\mathcal{C}$ and $\mathcal{D}$.

We introduce individual names $a_0, a_1, \dots, a_k$, where a_k denotes the root of the quantifier tree. Suppose we ensure that $\mathcal{J}$ is a model of the ABox

$$\mathcal{A} \;=\; \{s(a_i, a_{i+1}), s(a_i, a_k) \mid 0 \leq i < k\} \cup \{C_i(a_i), D_{i+1}(a_k) \mid 1 \leq i < k\}.$$

Then the structure of $\mathcal{J}$ is as shown in Figure 4.1. Let $\mathcal{C} \cup \mathcal{D} = \{C_1, \dots, C_k\}$, and recursively define concepts $D_1, \dots, D_k$ as follows:

$$\begin{aligned} D_i &:= \exists s.(C_i \sqcap D_{i+1}), \text{ for } 1 \leq i \leq k-1, \\ D_k &:= \exists s.C_k \end{aligned}$$

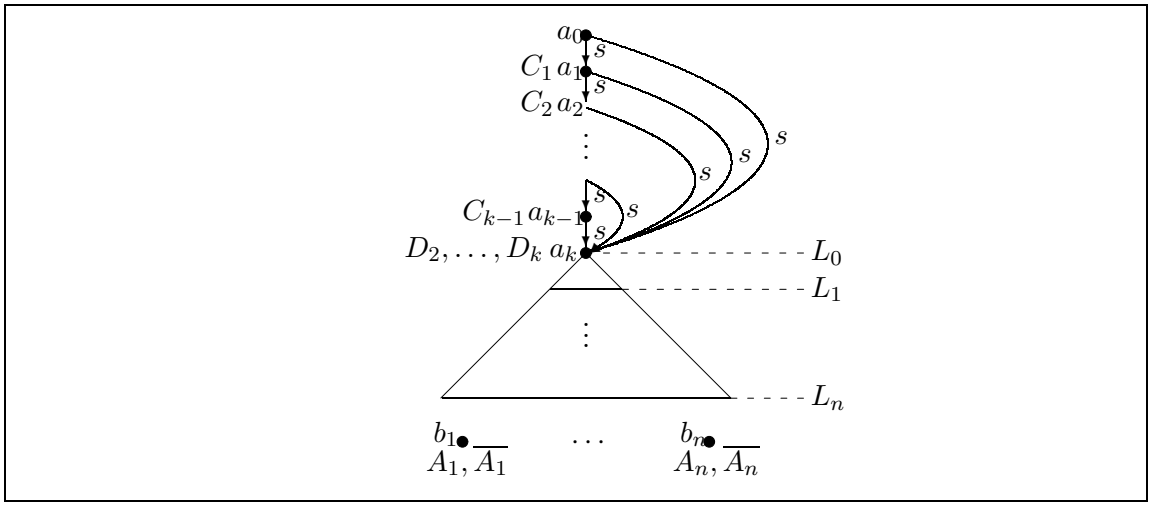

Figure 4.1: Structure of the reduction ABox $\mathcal{A}_\psi$.

To enforce that all concepts $C_1, \ldots, C_k$ are false at a_k in the model $\mathcal{J}$ in Figure 4.1, we can choose

$$\psi_\varphi := D_1(a_0).$$

To see that this has the intended effect, note that the above choice of ψ_ϕ enforces that $\neg D_1 \equiv \forall s.(\neg C_1 \sqcup \neg D_2)$ is true at a_0. Thus, $\neg C_1$ is true at a_k and $\neg D_2$ is true at a_1. It remains to repeat this argument $k-1$ times.

Unfortunately, including the ABox $\mathcal{A}$ as part of $\mathcal{A}_\phi$ does not result in $\mathcal{J}$ being a model of $\mathcal{A}$. The reason is that if the part of $\mathcal{I}$ that witnesses the truth of the assertions $C_i(a_i), D_{i+1}(a_k)$ in $\mathcal{A}_\phi$ involves the individuals $b_1, \ldots, b_n$ that occur in $\mathcal{U}$, then these assertions may be invalidated by $\mathcal{U}$ while transforming $\mathcal{I}$ to $\mathcal{J}$. The solution to this problem is as follows. For any concept C and individual name a, let $\mathsf{tree}_{C(a)}$ be an ABox that enforces a tree-shaped structure of connected individual names such that C is true at a. For example, if $C = \exists r.(A \sqcap \exists s.B \sqcap \exists r.(A \sqcap B))$, then

$$\mathsf{tree}_C(a) = \{r(a,c), A(c), s(c,c'), B(c'), r(c,c'')A(c''), B(c'')\}.$$

W.l.o.g., we assume that the individuals $b_1, \ldots, b_n$ do not occur in such ABoxes. In $\mathcal{A}_\phi$, we now use ABoxes of the form $\mathsf{tree}_{C(a)}$ instead of the original assertions $C_i(a_i)$ and $D_{i+1}(a_k)$. Since $b_1, \ldots, b_n$ are the only individuals occurring in $\mathcal{U}$ and we adopt the UNA, the generated structures are left untouched when $\mathcal{U}$ transforms $\mathcal{I}$ into $\mathcal{J}$. Summing up, the ABox $\mathcal{A}_\phi$ is thus as follows:

$$\begin{aligned}\mathcal{A}_\phi \;:=\; & \bigcup_{0 \le i < k} \{s(a_i, a_{i+1}), s(a_i, a_k)\} \cup \bigcup_{1 \le i < k} \mathsf{tree}_{C_i(a_i)} \cup \bigcup_{1 \le i < k} \mathsf{tree}_{D_{i+1}(a_k)} \\ & \cup \bigcup_{1 \le j \le n} \{(A_j \sqcap \overline{A}_j)(b_j)\} \cup \{L_0(a_k)\}\end{aligned}$$

Then the following lemma holds:

Lemma 4.1.11. *QBF ϕ is valid iff ψ_ϕ is not a consequence of applying $\mathcal{U}$ in $\mathcal{A}_\phi$ w.r.t. $\mathcal{T}_\phi$.*

Proof. Let us first define a sequence of $\mathcal{ELU}$ concepts $L'_0, L'_1, \ldots, L'_n$:

$$L'_{i-1} := \begin{cases} \bigsqcap_{1\leq j<i}(\exists t.A_j \sqcup \exists t.\overline{A}_j) \sqcap \exists r.(P_i \sqcap L'_i) \sqcap \exists r.(\overline{P}_i \sqcap L'_i), & Q_i = \forall \\ \bigsqcap_{1\leq j<i}(\exists t.A_j \sqcup \exists t.\overline{A}_j) \sqcap \exists r.L'_i, & Q_i = \exists \end{cases} \quad 1 \leq i \leq n$$

$$L'_n := \bigsqcap_{1\leq j\leq n}(\exists t.A_j \sqcup \exists t.\overline{A}_j)$$

Moreover, we define an $\mathcal{ELU}$ ABox $\mathcal{A}'_\phi$ as

$$\mathcal{A}'_\phi := \bigcup_{0\leq i<k}\{s(a_i,a_{i+1}), s(a_i,a_k)\} \cup \bigcup_{1\leq i<k} \mathsf{tree}_{C_i(a_i)} \cup \bigcup_{1\leq i<k} \mathsf{tree}_{D_{i+1}(a_k)} \cup \bigcup_{1\leq j\leq n}\{\overline{A}_j(b_j)\} \cup \{L'_0(a_k)\}$$

We divide the proof into several claims:

Claim 1. For every $\mathcal{I}, \mathcal{I}'$ such that $\mathcal{I} \models \mathcal{A}_\phi, \mathcal{T}_\phi$ and $\mathcal{I} \Rightarrow_\mathcal{U} \mathcal{I}'$ it holds that $\mathcal{I}' \models \mathcal{A}'_\phi$.

Proof: Recall the ABoxes $\mathsf{tree}_{C_i(a_i)}$ and $\mathsf{tree}_{D_{i+1}(a_k)}$ contain only atomic assertions and do not use individuals $\{b_1, \ldots, b_n\}$. Thus, since $\mathcal{I} \models \mathcal{A}_\phi$, and since $\mathcal{I}$ and $\mathcal{I}'$ differ only on the interpretation of A_j for b_j, $1 \leq j \leq n$, by definition of $\mathcal{A}_\phi$ and UNA we have that

$$\mathcal{I}' \models \bigcup_{0\leq i<k}\{s(a_i,a_{i+1}), s(a_i,a_k)\} \cup \bigcup_{1\leq i<k} \mathsf{tree}_{C_i(a_i)} \cup \bigcup_{1\leq i<k} \mathsf{tree}_{D_{i+1}(a_k)} \cup \bigcup_{1\leq j<n}\{\overline{A}_j(b_j)\}$$

The only non-trivial part is to show that $\mathcal{I}' \models L'_0(a_k)$: Let x be such that $x \in (\exists t.A_j)^\mathcal{I}$, for some $1 \leq j \leq n$. If $(x, b_j^\mathcal{I}) \notin t^\mathcal{I}$, it holds that $x \in (\exists t.A_j)^{\mathcal{I}'}$. If $(x, b_j^\mathcal{I}) \in t^\mathcal{I}$, we have that $x \in (\exists t.\overline{A}_j)^{\mathcal{I}'}$. In both cases it holds that $x \in (\exists t.A_j \sqcup \exists t.\overline{A})^{\mathcal{I}'}$. We obtain the following for $1 \leq i \leq n$:

$$x \in (\bigsqcup_{1\leq j\leq i} \exists t.A_j)^\mathcal{I} \text{ implies } x \in (\bigsqcup_{1\leq j\leq i}(\exists t.A_j \sqcup \exists t.\overline{A}))^{\mathcal{I}'} \qquad (*)$$

By using $(*)$ it is easy to show by induction on i that $x \in L^\mathcal{I}_{n-i}$ implies that $x \in (L'_{n-i})^{\mathcal{I}'}$. Since $\mathcal{I} \models \mathcal{A}_\phi$, it holds that $\mathcal{I} \models L_0(a_k)$, and thus we obtain that $\mathcal{I}' \models L'_0(a_k)$.

This completes the proof of Claim 1.

Claim 2. There exist interpretations $\mathcal{I}, \mathcal{I}'$ such that $\mathcal{I} \models \mathcal{T}_\phi, \mathcal{A}_\phi$, $\mathcal{I} \Rightarrow_\mathcal{U} \mathcal{I}'$ and $\mathcal{I}' \models \neg\psi_\phi$ iff $L'_0 \sqcap \bigsqcap_{1\leq i\leq k} \neg C_i$ is satisfiable.

Proof: "⇒" Let $\mathcal{I}$ be such that $\mathcal{I} \models \mathcal{T}_\phi, \mathcal{A}_\phi$, $\mathcal{I} \Rightarrow_\mathcal{U} \mathcal{I}'$, and $\mathcal{I}' \models \neg\psi_\phi$. By Claim 1, we have that $\mathcal{I}' \models \mathcal{A}'_\phi$ and thus $\mathcal{I}' \models L'_0(a_k)$. It remains to show that $\mathcal{I}' \models \neg C_i(a_k)$ for all $1 \leq i \leq k$. Since $\mathcal{I}' \models \bigcup_{1\leq i<k} \mathcal{A}_{C_i(a_i)} \cup \bigcup_{1\leq i<k} \mathcal{A}_{D_{i+1}(a_k)}$ and $\mathcal{A}_{C_i(a_i)} \models C_i(a_i)$, $\mathcal{A}_{D_{i+1}(a_k)} \models D_{i+1}(a_k)$ for $1 \leq i < k$, we have that $\mathcal{I}' \models C_i(a_i)$, $\mathcal{I}' \models D_{i+1}(a_k)$ for $1 \leq i < k$.

We show that the following holds for all $1 \leq i \leq k$: (i) $\mathcal{I}' \models \neg D_i(a_{i-1})$; (ii) $\mathcal{I}' \models \neg C_i(a_k)$. Proof is by induction on i:

- $i = 1$. (i) $\mathcal{I}' \models \neg D_1(a_0)$ holds since $\mathcal{I}' \models \neg\psi_\phi$ and $\psi_\phi = D_1(a_0)$. (ii) $\mathcal{I}' \models \mathcal{A}'_\phi$ implies $\mathcal{I}' \models s(a_0, a_k)$ and $\mathcal{I}' \models D_2(a_k)$. Thus $\mathcal{I}' \models \neg D_1(a_0)$ and $\neg D_1 \equiv \forall s.(\neg C_1 \sqcup \neg D_2)$ imply that $\mathcal{I}' \models \neg C_1(a_k)$.
- $i = m$, $2 \leq m \leq k$. (i) By I.H., we have that $\mathcal{I}' \models \neg D_{m-1}(a_{m-2})$. $\mathcal{I}' \models \mathcal{A}'_\varphi$ implies $\mathcal{I}' \models s(a_{m-2}, a_{m-1})$ and $\mathcal{I}' \models C_{m-1}(a_{m-1})$. Thus $\mathcal{I}' \models \neg D_{m-1}(a_{m-2})$ and $\neg D_{m-1} \equiv \forall s.(\neg C_{m-1} \sqcup \neg D_m)$ imply that $\mathcal{I}' \models \neg D_m(a_{m-1})$. (ii) Case $m \leq k-1$ is similar to the base case. For $m = k$, by I.H., we have that $\mathcal{I}' \models \neg D_k(a_{k-1})$. $\mathcal{I}' \models \mathcal{A}'_\phi$ implies $\mathcal{I}' \models s(a_{k-1}, a_k)$, and since $\neg D_k \equiv \forall s.\neg C_k$, we obtain $\mathcal{I}' \models \neg C_k(a_k)$.

We have found an interpretation $\mathcal{I}'$ such that $\mathcal{I}' \models (L'_0 \sqcap \bigsqcap_{1\leq i\leq k} \neg C_i)(a_k)$, thus $L'_0 \sqcap \bigsqcap_{1\leq i\leq k} \neg C_i$ is satisfiable.

"$\Leftarrow$" Let $L'_0 \sqcap \bigsqcap_{1\leq i\leq k} \neg C_i$ be satisfiable. Then there is an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ with $\mathsf{a}_k \in \Delta^{\mathcal{J}}$ such that $\mathsf{a}_k \in (L'_0 \sqcap \bigsqcap_{1\leq i\leq k} \neg C_i)^{\mathcal{J}}$.

We show how to construct an interpretation $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ such that

$$\mathcal{I}' \models \mathcal{A}'_\phi \cup \{(\bigsqcap_{1\leq i\leq k} \neg C_i)(a_k)\} \cup \mathcal{U}.$$

Let $\mathcal{I}_i = (\Delta_i, \cdot^{\mathcal{I}_i})$ be models of $\mathsf{tree}_{C_i(a_i)}$ for $1 \leq i \leq k-1$, and let $\mathcal{I}_k = (\Delta_k, \cdot^{\mathcal{I}_k})$ be a model of $\bigcup_{1\leq i\leq k-1} \mathsf{tree}_{D_{i+1}(a_k)}$ satisfying the following conditions:

(i) $a_i^{\mathcal{I}_i} = \mathsf{a}_i$, with $\mathsf{a}_i \in \Delta_i$, for $1 \leq i \leq k$

(ii) $\Delta_i \cap \Delta^{\mathcal{J}} = \emptyset$ for $1 \leq i \leq k-1$

(iii) $\Delta_k \cap \Delta^{\mathcal{J}} = \{\mathsf{a}_k\}$

Moreover, let $\mathsf{b}_1, \ldots, \mathsf{b}_n$ be such that $\{\mathsf{b}_1, \ldots, \mathsf{b}_n\} \cap (\Delta^{\mathcal{J}} \cup \bigcup_{1\leq i\leq k} \Delta_k) = \emptyset$.

We set $\Delta^{\mathcal{I}'} := \{\mathsf{a}_0, \ldots, \mathsf{a}_k\} \cup \{\mathsf{b}_1, \ldots, \mathsf{b}_n\} \cup \Delta^{\mathcal{J}} \cup \bigcup_{1\leq i\leq k} \Delta_k$.

$$\begin{array}{rcl}
a_i^{\mathcal{I}'} & := & \mathsf{a}_i, \quad 0 \leq i \leq k \\
b_j^{\mathcal{I}'} & := & \mathsf{b}_j, \quad 1 \leq j \leq n \\
s^{\mathcal{I}'} & := & \bigcup_{1\leq i\leq k} \{(\mathsf{a}_{i-1}, \mathsf{a}_i), (\mathsf{a}_{i-1}, \mathsf{a}_k)\} \cup s^{\mathcal{I}_k} \\
r^{\mathcal{I}'} & := & r^{\mathcal{J}} \cup \bigcup_{1\leq i\leq k} r^{\mathcal{I}_i} \\
A_j^{\mathcal{I}'} & := & A_j^{\mathcal{J}} \cup \bigcup_{1\leq i\leq k} A_j^{\mathcal{I}_i}, \quad 1 \leq j \leq n \\
\overline{A}_j^{\mathcal{I}'} & := & \overline{A}_j^{\mathcal{J}} \cup \bigcup_{1\leq i\leq k} \overline{A}_j^{\mathcal{I}_i} \cup \{\mathsf{b}_j\}, \quad 1 \leq j \leq n \\
P_j^{\mathcal{I}'} & := & P_j^{\mathcal{J}} \cup \bigcup_{1\leq i\leq k} P_j^{\mathcal{I}_i}, \quad 1 \leq j \leq n \\
\overline{P}_j^{\mathcal{I}'} & := & \overline{P}_j^{\mathcal{J}} \cup \bigcup_{1\leq i\leq k} \overline{P}_j^{\mathcal{I}_i}, \quad 1 \leq j \leq n
\end{array}$$

$$t^{\mathcal{I}'} \;:=\; t^{\mathcal{J}} \cup \bigcup_{1\leq i\leq k} r^{\mathcal{I}_i} \cup \{(d,\mathsf{b}_j) \mid d\in\Delta^{\mathcal{J}}, d\in(\exists t.\overline{A}_j)^{\mathcal{J}}\}$$

It is not difficult to see that $\mathcal{I}'$ indeed models $\mathcal{A}'_\phi \cup \{(\bigsqcap_{1\leq i\leq k}\neg C_i)(a_k)\} \cup \mathcal{U}$: Note that for $d\in\Delta^{\mathcal{J}}$, it holds that $d\in(\exists t.\overline{A}_j)^{\mathcal{J}}$ iff $d\in(\exists t.\overline{A}_j)^{\mathcal{I}'}$. As a direct consequence we obtain that $\mathcal{J}\models(L'_0\sqcap\bigsqcap_{1\leq i\leq k}\neg C_i)(a_k)$ implies $\mathcal{I}'\models(L'_0\sqcap\bigsqcap_{1\leq i\leq k}\neg C_i)(a_k)$. The rest follows directly from the definition of $\mathcal{I}'$ and requirements (i)-(iii) on interpretations $\mathcal{I}_1,\ldots,\mathcal{I}_k$.

In the next step we show by reverse induction that $\mathcal{I}'\models\neg D_i(a_{i-1})$ for $1\leq i\leq k$:

(i) $i=k$: Since $\neg D_k\equiv\forall s.\neg C_k$, $s^{\mathcal{I}'}(\mathsf{a}_{k-1})=\{\mathsf{a}_k\}$, and $\mathsf{a}_k\in(\neg C_k)^{\mathcal{I}'}$, we obtain that $\mathsf{a}_{k-1}\in(\neg D_k)^{\mathcal{I}'}$;

(ii) $i=m<k$. Since $\neg D_m\equiv\forall s.(\neg C_m\sqcup\neg D_{m+1})$, $s^{\mathcal{I}'}(\mathsf{a}_{m-1})=\{\mathsf{a}_i,\mathsf{a}_k\}$, $\mathsf{a}_k\in(\neg C_m)^{\mathcal{I}'}$, and $\mathsf{a}_i\in(\neg D_{m+1})^{\mathcal{I}'}$ (by I.H.), we obtain that $\mathsf{a}_{m-1}\in(\neg D_m)^{\mathcal{I}'}$.

Since $\psi_\phi=D_1(a_0)$, we obtain as a direct consequence that $\mathcal{I}'\models\neg\psi_\phi$.

Finally, having $\mathcal{I}'=(\Delta^{\mathcal{I}'},\cdot^{\mathcal{I}'})$, we construct $\mathcal{I}=(\Delta^{\mathcal{I}'},\cdot^{\mathcal{I}})$ such that: (i) $\mathcal{I}$ interprets all primitive concept, role and individual names as $\mathcal{I}'$, with the following exception: $A_j^{\mathcal{I}}:=A_j^{\mathcal{I}'}\cup\{\mathsf{b}_j\}$ for $1\leq j\leq n$; (ii) $\mathcal{I}\models\mathcal{T}_\phi$.

Obviously, we have that $\mathcal{I}\Rightarrow_{\mathcal{U}}\mathcal{I}'$. It remains to show that $\mathcal{I}\models\mathcal{A}_\phi$. The only non-trivial part is to show that $\mathcal{I}\models L_0(a_k)$, the rest is analogous to the proof of Claim 1. The definition of $t^{\mathcal{I}'}$ and the fact that $\mathsf{b}_j\in(A_j)^{\mathcal{I}}$ imply that for $x\in\Delta^{\mathcal{J}}$, the following holds:

$$x\in(\exists t.A_j\sqcup\exists t.\overline{A}_j)^{\mathcal{I}'} \text{ implies } x\in(\exists t.A_j)^{\mathcal{I}}. \qquad (**)$$

Since $\mathcal{I}'\models L'_0(a_k)$, as an easy consequence of $(**)$ we obtain that $\mathcal{I}\models L_0(a_k)$.

Thus, we have found interpretations $\mathcal{I},\mathcal{I}'$ such that $\mathcal{I}\models\mathcal{T}_\phi,\mathcal{A}_\phi$, $\mathcal{I}\Rightarrow_{\mathcal{U}}\mathcal{I}'$ and $\mathcal{I}'\models\neg\psi_\phi$.

This finishes the proof of Claim 2.

Claim 3. ϕ is valid iff $L'_0\sqcap\bigsqcap_{1\leq i\leq k}\neg C_i$ is satisfiable.

Proof. Recall that:

$$\begin{aligned}\bigsqcap_{1\leq i\leq k}\neg C_i \;=\;& \bigsqcap_{1\leq j\leq i<n}(\neg\exists r^i.(\exists t.A_j\sqcap\exists r.\exists t.\overline{A}_j))\sqcap\neg\exists r^i.(\exists t.\overline{A}_j\sqcap\exists r.\exists t.A_j))\\ &\sqcap\bigsqcap_{1\leq j\leq i<n}(\neg\exists r^i.(P_j\sqcap\exists t.\overline{A}_j)\sqcap\neg\exists r^i.(\overline{P}_j\sqcap\exists t.A_j)\sqcap\neg\exists r^i.(\exists t.A_j\sqcap\exists t.\overline{A}_j))\\ &\sqcap\bigsqcap_{1\leq i\leq m}\neg\exists r^n.C_{\overline{\varphi}_i}\end{aligned}$$

Thus, it holds that $\bigsqcap_{1\leq i\leq k}\neg C_i\equiv\mathsf{C}$, where

$$\begin{aligned}\mathsf{C} \;:=\;& \bigsqcap_{1\leq j\leq i<n}\forall r^i.(\exists t.A_j\rightarrow\forall r.(\neg\exists t.\overline{A}_j))\sqcap\forall r^i.(\exists t.\overline{A}_j\rightarrow\forall r.(\neg\exists t.A_j))\\ &\sqcap\bigsqcap_{1\leq j\leq i<n}(\forall r^i.(P_j\rightarrow\neg\exists t.\overline{A}_j)\sqcap\forall r^i.(\overline{P}_j\rightarrow\neg\exists t.A_j)\sqcap\forall r^i.\neg(\exists t.A_j\sqcap\exists t.\overline{A}_j))\\ &\sqcap\forall r^n.C_\varphi\end{aligned}$$

with $C_\varphi := \varphi[p_j/\exists t.A_j, \neg p_j/\exists t.\overline{A}_j, \wedge/\sqcap, \vee/\sqcup]$.

"$\Leftarrow$" If ϕ is valid, then ϕ has a valid quantifier tree T_ϕ. We use T_ϕ to construct a model $\mathcal{I}$ of $L'_0 \sqcap \bigsqcap_{1\leq i\leq k} \neg C_i$. We define the domain $\Delta^\mathcal{I}$ as

$$\Delta^\mathcal{I} := \{d \mid d \text{ is a vertix of } T_\phi\} \cup \{\mathsf{c}_1, \ldots, \mathsf{c}_n\} \cup \{\mathsf{b}_1, \ldots, \mathsf{b}_n\}$$

We set $A_j^\mathcal{I} := \{\mathsf{c}_j\}$ and $\overline{A}_j^\mathcal{I} := \{\mathsf{b}_j\}$ for $1 \leq j \leq n$. Finally, we define:

$$\begin{aligned}
r^\mathcal{I} &:= \{(d, d') \mid d' \text{ is a successor of } d' \text{ in } T_\phi\} \\
P_j^\mathcal{I} &:= \{d \mid d \text{ or an ancestor of } d \text{ has an incoming edge labelled with } p_j = \top\} \\
\overline{P}_j^\mathcal{I} &:= \{d \mid d \text{ or an ancestor of } d \text{ has an incoming edge labelled with } p_j = \bot\} \\
t^\mathcal{I} &:= \bigcup_{1\leq j\leq k} \{(d, \mathsf{c}_j) \mid d \text{ or an ancestor of } d \text{ has an incoming edge labelled with } p_j = \top\} \\
&\quad \cup \bigcup_{1\leq j\leq k} \{(d, \mathsf{b}_j) \mid d \text{ or an ancestor of } d \text{ has an incoming edge labelled with } p_j = \bot\}
\end{aligned}$$

It is not difficult to see that $\mathcal{I}$ is indeed a model of $L'_0 \sqcap \mathsf{C}$.

"$\Rightarrow$" Let $L'_0 \sqcap \mathsf{C}$ be satisfiable, and let $\mathcal{I}$ be its model with $y \in (L'_0 \sqcap \mathsf{C})^\mathcal{I}$. We extract a valid quantifier tree T_ϕ of ϕ by unravelling $\mathcal{I}$ at y in the following way:

T1 $y \in (L'_0)^\mathcal{I}$ is the root of T_ϕ

T2 if $x \in (L'_{i-1})^\mathcal{I}$ is a node of T_ϕ and $Q_i = \forall$, then choose $x_1, x_2 \in L_i^\mathcal{I}$ such that $(x, x_1) \in r^\mathcal{I}$, $(x, x_1) \in r^\mathcal{I}$ and $x_1 \in (\exists t.A_i)^\mathcal{I}$, $x_2 \in (\exists t.\overline{A}_i)^\mathcal{I}$, and make them children of x in T_ϕ. The edge (x, x_1) is labelled with $p_i = \top$ and (x, x_2) is labelled with $p_i = \bot$.

T3 if $x \in (L'_{i-1})^\mathcal{I}$ is a node of T_ϕ and $Q_i = \exists$, then choose a $x_1 \in L_i^\mathcal{I}$ such that $(x, x_1) \in r^\mathcal{I}$ and make it a child of x in T_ϕ. If $x_1 \in (\exists t.A_i)^\mathcal{I}$, then the edge (x, x_1) is labelled with $p_i = \top$ and if $x_1 \in (\exists t.\overline{A}_i)^\mathcal{I}$, it is labelled with $p_i = \bot$.

We will show that T_ϕ is well-defined and valid. Let x be a node of T_ϕ.

(i) The definition of L'_i ensures that for $x \in (L'_i)^\mathcal{I}$ it holds that $x \in (\exists t.A_i)^\mathcal{I} \cup (\exists t.\overline{A}_i)^\mathcal{I}$. Moreover, $y \in \mathsf{C}^\mathcal{I}$, and the conjunct $\forall r^i.\neg(\exists t.A_i \sqcap \exists t.\overline{A}_i)$ ensure that $x \notin (\exists t.A_i)^\mathcal{I} \cap (\exists t.\overline{A}_i)^\mathcal{I}$. Thus, the incoming edge of T_ϕ is labelled with exactly one label: either $p_i = \top$ or $p_i = \bot$.

(ii) The definition of L'_{i-1} for $Q_i = \forall$ ensures that $x \in (L'_{i-1})^\mathcal{I}$ has r-successors x_1 and x_2 with $x_1 \in (L'_i \sqcap P_i)^\mathcal{I}$ and $x_2 \in (L'_i \sqcap \overline{P}_i)^\mathcal{I}$. Since $x_1, x_2 \in (L'_i)^\mathcal{I}$, by (i) we have that for $j = 1, 2$: either $x_j \in (\exists t.A_i)^\mathcal{I}$ or $x_j \in (\exists t.\overline{A}_i)^\mathcal{I}$. Then the conjuncts $\forall r^i.(P_j \to \neg\exists t.\overline{A}_j)$ and $\forall r^i.(P_j \to \neg\exists t.\overline{A}_j)$ of C imply that $x_1 \in (\exists t.A_i)^\mathcal{I}$ and $x_2 \in (\exists t.\overline{A}_i)^\mathcal{I}$, i.e. it is ensured that the successors x_1, x_2 required in the step T2 indeed exist.

(iii) Conjuncts $\forall r^i.(\exists t.A_j \to \forall r.(\neg\exists t.\overline{A}_j))$ and $\forall r^i.(\exists t.\overline{A}_j \to \forall r.(\neg\exists t.A_j))$, together with $\forall r^i.\neg(\exists t.A_j \sqcap \exists t.\overline{A}_j)$ and the fact that for each node $x^{(i)}$ of T_ϕ such that $x^{(i)} \in (L'_i)^\mathcal{I}$ it holds that $x^{(i)} \in \bigcup_{1\leq j\leq i}(\exists t.A_j \sqcup \exists t.\overline{A}_j)^\mathcal{I}$ ensure that the labels "$\exists t.A_j$" and "$\exists t.\overline{A}_j$" are propagated from $x \in (L'_j)^\mathcal{I}$ to its successors, most importantly to the leaves of T_ϕ. The conjunct $\forall r^n.C_\varphi$

ensures that the formula φ evaluates to true at all the leaves of T_ϕ.

Since ϕ has a valid quantifier tree T_ϕ, ϕ is valid. This finishes the proof of Claim 3.

Claim 2 and Claim 3 show that a QBF ϕ is valid iff there exist interpretations $\mathcal{I}, \mathcal{I}'$ such that $\mathcal{I} \models \mathcal{T}_\phi, \mathcal{A}_\phi$, $\mathcal{I} \Rightarrow_\mathcal{U} \mathcal{I}'$ and $\mathcal{I}' \models \neg\psi_\phi$. This completes the proof of the lemma. ❑

Since the size of $\mathcal{T}_\phi$, $\mathcal{A}_\phi$, ψ_ϕ and $\mathcal{U}$ is polynomial in n, we have proven the following:

Lemma 4.1.12. *Projection in $\mathcal{EL}$ with acyclic TBoxes is* PSPACE*-hard.*

Since $\mathcal{EL}$ is a sublogic of $\mathcal{ALC}$ and projection in $\mathcal{ALC}$ is in PSPACE (see Theorem 3.2.1), we obtain the following result:

Theorem 4.1.13. *Projection in $\mathcal{EL}$ with acyclic TBoxes is* PSPACE*-complete.*

The PSPACE-hardness reduction from this section and the co-NP-hardness reduction from the previous section exploit only (weaker) unconditional post-conditions. With conditional post-conditions, disjunction does not have to be simulated by means of a complex construction in the reduction ABox; it is obtained almost for free. Consider the set of post-conditions

$$\mathcal{P} := \{C_1(a)/X(b), C_2(a)/X(b), \ldots, C_k(a)/X(b)\}.$$

Then $X(b)$ is a consequence of applying the action $(\emptyset, \emptyset, \mathcal{P})$ in an ABox $\mathcal{A}$ if and only if $(C_1 \sqcup C_2 \sqcup \cdots \sqcup C_k)(a)$ is a consequence of $\mathcal{A}$. This means that with conditional post-conditions we can simulate disjunction even in the fragment of $\mathcal{EL}$ without quantifiers. Hence we want to emphasize that our hardness results rely on the new source of complexity introduced by existential restrictions, and not on conditional post-conditions, as it would be the case in the propositional fragment of $\mathcal{EL}$.

4.2 Extensions

In this section we will consider instantiations of $\mathfrak{A}_1$ with extensions of $\mathcal{ALCQIO}$ towards OWL DL. We will show that for some of the expressive means, such as role inclusions, the syntax and semantics of $\mathfrak{A}_1$-actions can easily be extended in order to account for them. The only reason why we did not consider role inclusions by default in $\mathfrak{A}_1$ was our wish to keep semantics of $\mathfrak{A}_1$-actions as simple as possible. We show that, in the extended formalism, reasoning about actions can be reduced to standard DL reasoning, similar to the reduction from Section 3.2.1. On the other hand, expressive means such as transitive roles or cyclic TBoxes impose semantic problems, which will be discussed in Section 4.2.2.

4.2.1 Role Inclusions

Description logics underlying the web ontology language OWL provide for role inclusions. Please recall that role inclusions are expressions of the form:

$$r \sqsubseteq s$$

where r and s are (possibly inverse) roles. Moreover, recall that a *role box* $\mathcal{R}$ is a finite set of role inclusions. We use $\mathsf{Inv}(s)$ to denote s^- if s is a role name, and to denote r if

$s = r^-$, for a role name r. With $\sqsubseteq^*_{\mathcal{R}}$ we denote the transitive-reflexive closure of $\sqsubseteq$ relation in $\mathcal{R} \cup \{\mathsf{Inv}(r) \subseteq \mathsf{Inv}(r) \mid r \subseteq s \in \mathcal{R}\}$.

In this section we will show that role inclusions do not impose any semantic and computational problems if considered together with $\mathfrak{A}_1$ actions. We first modify the semantics of $\mathfrak{A}_1$-actions in order to account for ramifications caused by role inclusions, and then adapt the reduction of projection to DL reasoning such that it corresponds to the new semantics.

Semantics of Actions

We modify the semantics of $\mathfrak{A}_1$-actions in order to account for background role inclusions. Let $\mathcal{I}$ be an interpretation, $\mathcal{R}$ a role box and $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ an $\mathfrak{A}_1$-action. Since $\mathcal{R}$ may contain inverse roles, we define $\overline{\mathsf{occ}}$ and $\overline{\mathsf{post}}$ to be closures of occ and post under role inverses:

$$\begin{aligned}\overline{\mathsf{occ}} &:= \mathsf{occ} \cup \{r^-(b,a) \mid r(a,b) \in \mathsf{occ}\}\\ \overline{\mathsf{post}} &:= \mathsf{post} \cup \{\varphi/r^-(b,a) \mid \varphi/r(a,b) \in \mathsf{post}\} \cup \{\varphi/\neg r^-(b,a) \mid \varphi/\neg r(a,b) \in \mathsf{post}\}\end{aligned}$$

Let α transform the interpretation $\mathcal{I}$ into $\mathcal{I}'$ and let s and r be (possibly) inverse roles. There are two types of ramifications we need to take into account:

(i) If $\mathcal{I} \models \varphi$, $\varphi/s(a,b) \in \overline{\mathsf{post}}$ and $s \sqsubseteq^*_{\mathcal{R}} r$, we expect that not only $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in s^{\mathcal{I}'}$, but also $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}'}$.

(ii) Since $\mathfrak{A}_1$-actions contain occlusions, we want to ensure that if $s(a,b) \in \overline{\mathsf{occ}}$ and $s \sqsubseteq^*_{\mathcal{R}} r$, then not only $s(a,b)$ may "change freely", but also $r(a,b)$, however only in case $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}$.

In order to account for these two cases, we will modify $D^{\mathcal{I}}_\alpha(r)$ from Definition 3.1.4, rather than changing $\alpha^{\mathcal{I}}_+(r)$ and $\alpha^{\mathcal{I}}_-(r)$. Intuitively, in order to handle case (i), we ensure that $(a^{\mathcal{I}}, b^{\mathcal{I}})$ is not in $D^{\mathcal{I}}_\alpha(r)$ – i.e., it is not in the part of the domain for which we can determine the extension of r in $\mathcal{I}'$ "automatically". However, it does not mean that $r(a,b)$ may change freely since we will ensure that $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}'}$ in this case by requiring that both the pre-interpretation $\mathcal{I}$ and the post-interpretation $\mathcal{I}'$ are models of the role box $\mathcal{R}$.

We extend the notions of $\alpha^{\mathcal{I}}_+(r)$, $\alpha^{\mathcal{I}}_-(r)$ from Definition 3.1.4 and define $\mathsf{occ}^{\mathcal{I}}(r)$ for r a possibly inverse role:

$$\begin{aligned}\alpha^{\mathcal{I}}_+(r) &:= \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/r(a,b) \in \overline{\mathsf{post}} \wedge \mathcal{I} \models \varphi\}\\ \alpha^{\mathcal{I}}_-(r) &:= \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/\neg r(a,b) \in \overline{\mathsf{post}} \wedge \mathcal{I} \models \varphi\}\\ \mathsf{occ}^{\mathcal{I}}(r) &:= \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid r(a,b) \in \overline{\mathsf{occ}}\}\end{aligned}$$

We modify $D^{\mathcal{I}}_\alpha(r)$ in the following way:

$$D^{\mathcal{I}}_\alpha(r) := ((\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus \mathsf{occ}^{\mathcal{I}}_\alpha(r)) \cup (\alpha^{\mathcal{I}}_+(r) \cup \alpha^{\mathcal{I}}_-(r))$$

where

$$\begin{aligned}\mathsf{occ}^{\mathcal{I}}_\alpha(r) := \{&(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}} \mid (a^{\mathcal{I}}, b^{\mathcal{I}}) \in \alpha^{\mathcal{I}}_-(s) \cup \mathsf{occ}^{\mathcal{I}}(s) \text{ for some } r \sqsubseteq^*_{\mathcal{R}} s\\ &\qquad \text{or } (b^{\mathcal{I}}, a^{\mathcal{I}}) \in \alpha^{\mathcal{I}}_-(s) \cup \mathsf{occ}^{\mathcal{I}}(s) \text{ for some } r^- \sqsubseteq^*_{\mathcal{R}} s\} \cup\\ \{&(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}} \mid (a^{\mathcal{I}}, b^{\mathcal{I}}) \in \alpha^{\mathcal{I}}_+(s) \cup \mathsf{occ}^{\mathcal{I}}(s) \text{ for some } s \sqsubseteq^*_{\mathcal{R}} r\\ &\qquad \text{or } (b^{\mathcal{I}}, a^{\mathcal{I}}) \in \alpha^{\mathcal{I}}_+(s) \cup \mathsf{occ}^{\mathcal{I}}(s) \text{ for some } s \sqsubseteq^*_{\mathcal{R}} r^-\}\end{aligned}$$

For concept names A, the notions of $\alpha^{\mathcal{I}}_+(A)$, $\alpha^{\mathcal{I}}_-(A)$ and $D^{\mathcal{I}}_\alpha(A)$ are defined as in Definition 3.1.4. Now we are ready to define a modified semantics of actions.

Definition 4.2.1. Let $\mathcal{T}$ be an acyclic TBox, $\mathcal{R}$ a role box, $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ an action for $\mathcal{T}$, and $\mathcal{I}, \mathcal{I}'$ models of $\mathcal{T}$ and $\mathcal{R}$ sharing the same domain and interpretation of all individual names and respecting UNA for individual names. We say that α *may transform* $\mathcal{I}$ *to* $\mathcal{I}'$ *w.r.t.* $\mathcal{T}$ *and* $\mathcal{R}$ (written $\mathcal{I} \Rightarrow_\alpha^{\mathcal{T},\mathcal{R}} \mathcal{I}'$) iff, for each primitive concept A and role name r, we have

$$\begin{array}{rcl} \alpha_+^{\mathcal{I}}(A) \cap \alpha_-^{\mathcal{I}}(A) = \emptyset \quad \text{and} \quad \alpha_+^{\mathcal{I}}(r) \cap \alpha_-^{\mathcal{I}}(s) = \emptyset \text{ if } r \sqsubseteq_{\mathcal{R}}^* s \\ A^{\mathcal{I}'} \cap D_\alpha^{\mathcal{I}}(A) & = & ((A^{\mathcal{I}} \cup \alpha_+^{\mathcal{I}}(A)) \setminus \alpha_-^{\mathcal{I}}(A)) \cap D_\alpha^{\mathcal{I}}(A) \\ r^{\mathcal{I}'} \cap D_\alpha^{\mathcal{I}}(r) & = & ((r^{\mathcal{I}} \cup \alpha_+^{\mathcal{I}}(r)) \setminus \alpha_-^{\mathcal{I}}(r)) \cap D_\alpha^{\mathcal{I}}(r) \end{array}$$

△

Note that we replaced the requirement $\alpha_+^{\mathcal{I}}(r) \cap \alpha_-^{\mathcal{I}}(r) = \emptyset$ from the definition of $\mathcal{I} \Rightarrow_\alpha^{\mathcal{T}} \mathcal{I}'$ (c.f. Definition 3.1.4) with the stronger one:

$$\alpha_+^{\mathcal{I}}(r) \cap \alpha_-^{\mathcal{I}}(s) = \emptyset \text{ if } r \sqsubseteq_{\mathcal{R}}^* s$$

We say that an action $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ is *consistent w.r.t. acyclic TBox* $\mathcal{T}$ *and a role box* $\mathcal{R}$ iff for every interpretation $\mathcal{I}$, there exists $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow_\alpha^{\mathcal{T},\mathcal{R}} \mathcal{I}'$. It is not difficult to see that an action α is consistent iff the following conditions are satisfied:

1. for all $\varphi_1/A(a), \varphi_2/\neg A(a) \in \mathsf{post}$ it holds that $\{\varphi_1, \varphi_2\}$ is inconsistent w.r.t. $\mathcal{T}$ and $\mathcal{R}$.
2. for all $\varphi_1/r(a), \varphi_2/\neg s(a) \in \mathsf{post}$ with $r \sqsubseteq_{\mathcal{R}}^* s$ it holds that $\{\varphi_1, \varphi_2\}$ is inconsistent w.r.t. $\mathcal{T}$ and $\mathcal{R}$.

In what follows, we assume that actions are consistent with respect to background TBoxes and role boxes. The reasoning problems projection and executability, i.e. notions of *consequence of applying* α *in* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$ *and* $\mathcal{R}$, and *executable* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$ *and* $\mathcal{R}$ are defined as the analogous notions from Definition 3.1.6, with the exception that the relation $\Rightarrow_\alpha^{\mathcal{T},\mathcal{R}}$ is used instead of $\Rightarrow_\alpha^{\mathcal{T}}$. Projection and executability are mutually reducible in polynomial time. The proof is analogous to the proof of Lemma 3.1.7.

Reduction to DL reasoning

The reduction of projection to ABox consequence from Section 3.2.1 can easily be modified to account for role inclusions. We assume that the underlying DL is $\mathcal{ALCQIOH}$, i.e., the extension of $\mathcal{ALCQIO}$ with role inclusions. Let $\mathcal{A}$ be an ABox, $\alpha_1, \ldots, \alpha_n$ a composite action with $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$, $\mathcal{T}$ an acyclic TBox, $\mathcal{R}$ a role box, and φ_0 an assertion. We are interested in deciding whether φ_0 is a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$ and $\mathcal{R}$. We use the same notation and introduce the same concept, role and individual names for the reduction as in Section 3.2.1.

Let $\mathcal{T}_{\mathsf{red}}$ and φ_{red} be defined as in Section 3.2.1. Moreover, we define $\mathcal{R}_{\mathsf{red}}$ as follows:

$$\mathcal{R}_{\mathsf{red}} = \{r^{(i)} \sqsubseteq s^{(i)} \mid r \sqsubseteq s \in \mathcal{R}, i \leq n\}.$$

Finally, we define ABoxes $\mathcal{A}_{\mathsf{min}_r}^{(i)}$ in the following way: Part 1 tackling minimization of changes for concept names is defined in the same way as in $\mathcal{A}_{\mathsf{min}}^{(i)}$ from Section 3.2.1. Concerning Part 2, $\mathcal{A}_{\mathsf{min}_r}^{(i)}$ contains the following assertions for all $a, b \in \mathsf{Ind}$ and every role name r with

- $s(a,b) \notin \overline{\mathsf{occ}}_i$ for all (possibly inverse) roles s with $r \sqsubseteq^*_{\mathcal{R}} s$ and $s(b,a) \notin \overline{\mathsf{occ}}_i$ for all (possibly inverse) roles s with $r^- \sqsubseteq^*_{\mathcal{R}} s$:

$$a : \Big(\big(\exists r^{(i-1)}.\{b\} \;\sqcap \bigsqcap_{r \sqsubseteq^*_{\mathcal{R}} s \; \varphi/\neg s(a,b) \in \overline{\mathsf{post}}_i} \neg \mathsf{p}_{i-1}(\varphi) \\ \sqcap \bigsqcap_{r^- \sqsubseteq^*_{\mathcal{R}} s \; \varphi/\neg s(b,a) \in \overline{\mathsf{post}}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \rightarrow \exists r^{(i)}.\{b\} \Big)$$

- $s(a,b) \notin \mathsf{occ}_i$ for all (possibly inverse) roles s with $s \sqsubseteq^*_{\mathcal{R}} r$ and $s(b,a) \notin \mathsf{occ}_i$ for all (possibly inverse) roles s with $s \sqsubseteq^*_{\mathcal{R}} r^-$:

$$a : \Big(\big(\forall r^{(i-1)}.\neg\{b\} \;\sqcap \bigsqcap_{s \sqsubseteq^*_{\mathcal{R}} r \; \varphi/s(a,b) \in \overline{\mathsf{post}}_i} \neg \mathsf{p}_{i-1}(\varphi) \\ \sqcap \bigsqcap_{s \sqsubseteq^*_{\mathcal{R}} r^- \; \varphi/s(b,a) \in \overline{\mathsf{post}}_i} \neg \mathsf{p}_{i-1}(\varphi)\big) \rightarrow \forall r^{(i)}.\neg\{b\} \Big).$$

$\mathcal{A}^r_{\mathsf{red}}$ is obtained by replacing $\mathcal{A}^{(i)}_{\mathsf{min}}$ in the definition of $\mathcal{A}_{\mathsf{red}}$ from Section 3.2.1 by $\mathcal{A}^{(i)}_{\mathsf{min}_r}$ for $i \leq n$.
Then the following lemma holds:

Lemma 4.2.2. *φ is a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ and $\mathcal{R}$ iff φ_{red} is a consequence of $\mathcal{A}^r_{\mathsf{red}}$ w.r.t. $\mathcal{T}_{\mathsf{red}}$ and $\mathcal{R}_{\mathsf{red}}$ under UNA.*

Proof. We do not present all proof details, as a proof is an easy modification of the proof of Lemma 3.2.3. Thus we only show how the modified parts of the reduction correspond to the modified semantics of actions.

"$\Rightarrow$" Proof by contraposition. Assume that φ_{red} is not a consequence of $\mathcal{A}^r_{\mathsf{red}}$ w.r.t. $\mathcal{T}_{\mathsf{red}}$ and $\mathcal{R}_{\mathsf{red}}$. i.e. there is a joint model $\mathcal{J}$ of $\mathcal{A}_{\mathsf{red}}$, $\mathcal{T}_{\mathsf{red}}$, $\mathcal{R}_{\mathsf{red}}$ and $\neg\varphi_{\mathsf{red}}$.

We define interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$, based on $\mathcal{J}$, as in the proof of Lemma 3.2.3. Recall that $\mathcal{I}_0, \ldots, \mathcal{I}_n$ share the same domain and interpretation of individual names with $\mathcal{J}$. Moreover, for concept names A, $A^{\mathcal{I}_i} := (T^{(i)}_A)^{\mathcal{J}}$, while for role names r, $r^{\mathcal{I}_i} := (r^{(i)})^{\mathcal{J}} \cap (N^{\mathcal{J}} \times N^{\mathcal{J}}) \cup (r^{(0)})^{\mathcal{J}} \cap \big(\Delta^{\mathcal{J}} \times (\neg N)^{\mathcal{J}} \cup (\neg N)^{\mathcal{J}} \times \Delta^{\mathcal{J}}\big)$.

It remains to show that: (i) $\mathcal{I}_0 \models \mathcal{A}$; (ii) $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T},\mathcal{R}}_{\alpha_i} \mathcal{I}_i$ for all $1 \leq i \leq n$; and (iii) $\mathcal{I}_n \not\models \varphi$. Since (i) and (iii) can be shown in the same way as in the proof of Lemma 3.2.3, as well as the following subtasks of (ii): $\mathcal{I}_i \models \mathcal{T}, i \leq n$ and the required correlation between interpretations $A^{\mathcal{I}_i}$ and $A^{\mathcal{I}_{i+1}}$ for primitive concept names A. Thus we focus on the remaining subtasks of (ii):

- $\mathcal{I}_i \models \mathcal{R}$ is a direct consequence of $\mathcal{J} \models \mathcal{R}_{\mathsf{red}}$ and the definition of $r^{\mathcal{I}_i}$ for $i \leq n$.
- Let r be a role name and let $\alpha_{i+}^{\mathcal{I}_{i-1}}(r)$, $\alpha_{i-}^{\mathcal{I}_{i-1}}(r)$, $\mathsf{occ}_{\alpha_i}^{\mathcal{I}_{i-1}}(r)$, and $D_{\alpha_i}^{\mathcal{I}_{i-1}}(r)$ be defined as in Definition 4.2.1. Then $\alpha_{i+}^{\mathcal{I}_{i-1}}(r) \cap \alpha_{i-}^{\mathcal{I}_{i-1}}(s) = \emptyset$ for all s such that $r \sqsubseteq^*_{\mathcal{R}} s$ since α_i is consistent with $\mathcal{T}$ and $\mathcal{R}$. It remains to show that if $(x,y) \in D_{\alpha_i}^{\mathcal{I}_{i-1}}(r)$, then

$$x \in r^{\mathcal{I}_i} \text{ iff } x \in (r^{\mathcal{I}_{i-1}} \cup \alpha_{i+}^{\mathcal{I}_{i-1}}(r)) \setminus \alpha_{i-}^{\mathcal{I}_{i-1}}(r) \qquad (*)$$

  Since the case $(x,y) \notin N^{\mathcal{J}} \times N^{\mathcal{J}}$ is trivial, we assume that $(x,y) = (a^{\mathcal{J}}, b^{\mathcal{J}})$, where $a, b \in \mathsf{Ind}$.

  Since $\mathcal{J} \models \mathcal{A}^{(i)}_{\mathsf{post}}$, we obtain the following implications:

(i) $(a^{\mathcal{J}}, b^{\mathcal{J}}) \in \alpha_{i+}^{\mathcal{I}_{i-1}}(r)$ implies $(a^{\mathcal{J}}, b^{\mathcal{J}}) \in r^{\mathcal{I}_i}$, and

(ii) $(a^{\mathcal{J}}, b^{\mathcal{J}}) \in \alpha_{i-}^{\mathcal{I}_{i-1}}(r)$ implies $(a^{\mathcal{J}}, b^{\mathcal{J}}) \notin r^{\mathcal{I}_i}$.

Moreover, $\mathcal{J} \models \mathcal{A}^{(i)}_{\mathsf{min}_r}$ implies that:

(iii) $(a^{\mathcal{J}}, b^{\mathcal{J}}) \in r^{\mathcal{I}_{i-1}} \setminus \mathsf{occ}_{\alpha_i}^{\mathcal{I}_{i-1}}(r)$ implies $(a^{\mathcal{J}}, b^{\mathcal{J}}) \in r^{\mathcal{I}_i}$, and

(iv) $(a^{\mathcal{J}}, b^{\mathcal{J}}) \in r^{\mathcal{I}_i} \cup \mathsf{occ}_{\alpha_i}^{\mathcal{I}_{i-1}}(r)$ implies $(a^{\mathcal{J}}, b^{\mathcal{J}}) \in r^{\mathcal{I}_{i-1}}$.

It is not difficult to see that $(*)$ follows from (i)-(iv) and the definition of $\mathsf{occ}_{\alpha_i}^{\mathcal{I}_{i-1}}(r)$ and $D_{\alpha_i}^{\mathcal{I}_{i-1}}(r)$.

"$\Leftarrow$" This direction as also an easy adaption of the corresponding direction of the proof of Lemma 3.2.3. ❑

Since ABox consequence is known to be in co-NExpTime in $\mathcal{ALCQIOH}$ (as a sublogic of C^2) [PST00], we conclude that projection in $\mathcal{ALCQIOH}$ is in co-NExpTime. Since projection is co-NExpTime-hard already in $\mathcal{ALCQI}$, c.f. Theorem 3.2.6, we obtain the following result:

Theorem 4.2.3. *Projection in $\mathcal{ALCQIOH}$ is co-*NExpTime*-complete, even if numbers in number restrictions are coded in binary.*

4.2.2 Problematic Extensions

In the previous section we have shown that allowing for role inclusions along with the action formalism $\mathfrak{A}_1$ does not cause any semantic or computational problems. In this section, we show that two other natural extensions, such as allowing for transitive roles, or cyclic instead of acyclic TBoxes cause *semantic* and/or *computational problems*.

Transitive Roles

Transitive roles are offered by most modern DL systems [TH06, HM01a], and also by the ontology language OWL [HPSvH03]. They can be added to $\mathcal{ALCQIO}$ by reserving a subset of roles $\mathsf{N_{tR}}$ of $\mathsf{N_R}$ such that all $r \in \mathsf{N_{tR}}$ are required to be interpreted as transitive relations $r^{\mathcal{I}}$ in all models $\mathcal{I}$. We will show that admitting the use of transitive roles in post-conditions yields semantic problems.

By Lemma 3.1.5, $\mathfrak{A}_1$-actions without occlusions $\alpha = (\mathsf{pre}, \emptyset, \mathsf{post})$ are deterministic in the sense that $\mathcal{I} \Rightarrow_\alpha^{\mathcal{T}} \mathcal{I}'$ and $\mathcal{I} \Rightarrow_\alpha^{\mathcal{T}} \mathcal{I}''$ implies $\mathcal{I}' = \mathcal{I}''$. This is not any more the case for actions referring to transitive roles.

Consider the action $\alpha = (\emptyset, \emptyset, \{\neg\mathsf{has\text{-}part}(\mathsf{car}, \mathsf{valve})\})$ that removes a valve from a car. Let $\mathsf{has\text{-}part}$ be a transitive role and take the model

$$\begin{aligned} \Delta^{\mathcal{I}} &:= \{\mathsf{car}, \mathsf{engine}, \mathsf{valve}\} \\ \mathsf{has\text{-}part}^{\mathcal{I}} &:= \{(\mathsf{car}, \mathsf{engine}), (\mathsf{engine}, \mathsf{valve}), (\mathsf{car}, \mathsf{valve})\} \\ z^{\mathcal{I}} &:= z \text{ for } z \in \Delta^{\mathcal{I}}. \end{aligned}$$

Due to transitivity of $\mathsf{has\text{-}part}$, it does not suffice only to remove the edge $\mathsf{has\text{-}part}(\mathsf{car}, \mathsf{valve})$ when applying α in $\mathcal{I}$. Indeed, we also need to remove either $(\mathsf{car}, \mathsf{engine})$ or $(\mathsf{engine}, \mathsf{valve})$ (or even both). Thus, both interpretations $\mathcal{I}'$ and $\mathcal{I}''$, where $\mathcal{I}'$ is obtained from $\mathcal{I}$ by setting

$$\mathsf{has\text{-}part}^{\mathcal{I}'} := \{(\mathsf{car}, \mathsf{engine})\}$$

and $\mathcal{I}''$ is obtained from $\mathcal{I}$ by setting

$$\mathsf{has\text{-}part}^{\mathcal{I}''} := \{(\mathsf{engine}, \mathsf{valve})\}.$$

are possible outcomes of executing action α in $\mathcal{I}$.

In the area of reasoning about actions, it is well-known that non-determinism of this kind requires extra effort to obtain sensible consequences of action executions [Lin96, Thi00]. In the above example, it is unlikely that both outcomes of the action application are equally desirable. Thus, we need a mechanism for eliminating unwanted outcomes or preferring the desired ones. We leave such extensions as future work.

Cyclic TBoxes

Assume that we admit cyclic TBoxes as defined in Section 2.1. For cyclic TBoxes, just like for acyclic ones, we can differentiate between primitive and defined concept names. Hence we can define the semantics of actions in the presence of cyclic TBoxes in the same way as for the acyclic ones, c.f. Definition 3.1.4.

However, semantic problems arise due to a crucial difference between cyclic and acyclic TBoxes: for acyclic TBoxes, the interpretation of primitive concepts *uniquely* determines the extension of the defined ones, while this is not the case for cyclic ones. Together with the fact that the transition relation between interpretations $\Rightarrow_\alpha^{\mathcal{T}}$ only takes into account primitive concepts, this means that the minimization of changes induced by action application does not work as expected. To see this, consider the following example:

$$\begin{aligned} \mathcal{A} &:= \{\mathsf{Dog}(a)\} \\ \mathcal{T} &:= \{\mathsf{Dog} \doteq \exists \mathsf{parent}.\mathsf{Dog}\} \\ \mathsf{post} &:= \{\mathsf{Cat}(b)\} \end{aligned}$$

Then, $\mathsf{Dog}(a)$ is *not* a consequence of applying $\alpha = (\emptyset, \emptyset, \mathsf{post})$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$, as one would intuitively expect. This is due to the following counter-model. Define an interpretation $\mathcal{I}$ as follows:

$$\begin{aligned} \Delta^{\mathcal{I}} &:= \{b\} \cup \{d_0, d_1, d_2, \ldots\} \\ \mathsf{Dog}^{\mathcal{I}} &:= \{d_0, d_1, d_2, \ldots\} \\ \mathsf{Cat}^{\mathcal{I}} &:= \emptyset \\ \mathsf{parent}^{\mathcal{I}} &:= \{(d_i, d_{i+1}) \mid i \in \quad\} \\ a^{\mathcal{I}} &:= d_0 \\ b^{\mathcal{I}} &:= b \end{aligned}$$

The interpretation $\mathcal{I}'$ is defined as $\mathcal{I}$, with the exception that $\mathsf{Cat}^{\mathcal{I}'} = \{b\}$ and $\mathsf{Dog}^{\mathcal{I}'} := \emptyset$. Using the fact that Dog is a defined concept and thus not considered in the definition of $\Rightarrow_\alpha^{\mathcal{T}}$, it is easy to see that $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_\alpha^{\mathcal{T}} \mathcal{I}'$, and $\mathcal{I}' \not\models \mathsf{Dog}(a)$.

There appears to exists a way how to solve this problem: if we adopt the least or greatest fixpoint semantics for TBoxes as first proposed by Nebel [Neb91], it is indeed the case that primitive concepts uniquely determine defined concepts. Thus, it may be interesting to analyze actions with cyclic TBoxes under fixpoint semantics as future work. Moreover, in the next chapter we will present a more general action formalism, designed to account for general TBoxes, thus being also suitable for cyclic TBoxes.

4.3 Complex Concepts in Post-Conditions and GCIs: PMA

Even more general than admitting cyclic TBoxes is to allow general concept inclusions (GCIs). Recall that a *GCI* is an expression $C \sqsubseteq D$, with C and D (possibly complex) concepts. Since (sets of) GCIs strictly generalize cyclic TBoxes, when admitting GCIs in connection with actions, we run into the same problems as with cyclic TBoxes. However, the problems are even more serious in the case of GCIs: first, GCIs do not allow an obvious partitioning of concept names into primitive and defined ones. Thus, in the definition of $\Rightarrow_\alpha^{\mathcal{T}}$, the only choice is to minimize *all* concept names. Second, the vanished distinction between primitive and defined concepts means that we can no longer restrict concepts C in post-conditions $\varphi/C(a)$ to literals over *primitive* concept names. The best we can do is to restrict such concepts to literals over arbitrary concept names. However, together with the two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$ with C a complex concept, the literal post-condition $\varphi/A(a)$ is equivalent to the complex one $\varphi/C(a)$. Thus, it seems that GCIs cannot be admitted without simultaneously admitting arbitrarily complex concepts in post-conditions.

Let a *generalized* $\mathfrak{A}_1$-action be an action where post-conditions are of the form φ/ψ for arbitrary assertions φ and ψ. In other words, ψ is no longer restricted to be a literal over primitive concepts. One possible way to define the semantics of generalized $\mathfrak{A}_1$-actions is to adopt the *possible model approach (PMA)* semantics. PMA was first introduced by Winslett [Win88] and further elaborated e.g. in [Win90, EG92, Her96]. The idea is to introduce a proximity relation $\preceq_{\mathcal{I}}$ between interpretations. Intuitively, $\mathcal{I}' \preceq_{\mathcal{I}} \mathcal{I}''$ means that $\mathcal{I}'$ is "closer" to $\mathcal{I}$ than $\mathcal{I}''$ is. Then an action α may transform $\mathcal{I}$ into $\mathcal{I}'$ iff: (i) $\mathcal{I}'$ satisfies post-conditions of α; and (ii) $\mathcal{I}'$ is the "closest" interpretation to $\mathcal{I}$ satisfying (i). For description logic interpretations $\mathcal{I}$ and $\mathcal{I}'$ this means that the difference in interpretations of concept and role names in $\mathcal{I}$ and $\mathcal{I}'$ is *minimized*.

As we will see in the next section, PMA semantics can be defined for generalized $\mathfrak{A}_1$-actions in a clean and elegant way. Moreover, it is a generalization of the standard $\mathfrak{A}_1$- actions semantics (c.f. Definition 3.1.4) in the sense that not only the minimization of changes for *primitive*, but for *all* concept names is enforced. This inevitably boils down to minimizing changes of *complex concepts*, which turns out to cause both *semantic* and *computational* problems. These problems are going to be discussed in Section 4.3.2.

4.3.1 PMA Semantics of Generalized $\mathfrak{A}_1$-Actions

We start by formally introducing the proximity relation $\preccurlyeq_{\mathcal{I},\alpha}$

Definition 4.3.1 (Preferred Interpretations). Let $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ be a generalized $\mathfrak{A}_1$-action, let $\mathcal{T}$ be general TBox, and $\mathcal{I}$ a model of $\mathcal{T}$. We define the binary relation $\preccurlyeq_{\mathcal{I},\alpha}$ on models of $\mathcal{T}$ by setting $\mathcal{I}' \preccurlyeq_{\mathcal{I},\alpha} \mathcal{I}''$ iff

- $((A^{\mathcal{I}} \triangledown A^{\mathcal{I}'}) \setminus \{a^{\mathcal{I}} \mid A(a) \in \mathsf{occ}\}) \subseteq A^{\mathcal{I}} \triangledown A^{\mathcal{I}''}$;

- $((r^{\mathcal{I}} \triangledown r^{\mathcal{I}'}) \setminus \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid r(a,b) \in \mathsf{occ}\}) \subseteq r^{\mathcal{I}} \triangledown r^{\mathcal{I}''}$.

for all concept names A and role names r. △

Intuitively, applying the action α transforms the interpretation $\mathcal{I}$ into the interpretation $\mathcal{I}'$ if $\mathcal{I}'$ satisfies the post-conditions and is closest to $\mathcal{I}$ (as expressed by $\preccurlyeq_{\mathcal{I},\alpha}$) among all interpretations satisfying the post-conditions. Since we consider *conditional* post-conditions, defining when they are satisfied actually involves both $\mathcal{I}$ and $\mathcal{I}'$. We say that the pair of interpretations $\mathcal{I}, \mathcal{I}'$ *satisfies the set of post-conditions* post ($\mathcal{I}, \mathcal{I}' \models \mathsf{post}$) iff the following holds for all post-conditions φ/ψ in post: $\mathcal{I}' \models \psi$ whenever $\mathcal{I} \models \varphi$.

Definition 4.3.2 (PMA Semantics of Actions). Let $\mathcal{T}$ be a general TBox, $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ a generalized $\mathfrak{A}_1$-action, and $\mathcal{I}, \mathcal{I}'$ models of $\mathcal{T}$ sharing the same domain and interpretation of all individual names and respecting UNA for individual names. Then α *may transform* $\mathcal{I}$ to $\mathcal{I}'$ ($\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha} \mathcal{I}'$) iff

1. $\mathcal{I}, \mathcal{I}' \models \mathsf{post}$, and
2. there does not exist a model $\mathcal{J}$ of $\mathcal{T}$ such that $\mathcal{I}, \mathcal{J} \models \mathsf{post}$, $\mathcal{J} \neq \mathcal{I}'$, and $\mathcal{J} \preccurlyeq_{\mathcal{I},\alpha} \mathcal{I}'$.

△

It is not difficult to see that the PMA semantics is equivalent with the semantics of standard $\mathfrak{A}_1$-actions from Definition 3.1.4, if the precedence relation $\preccurlyeq_{\mathcal{I},\alpha}$ *does not* take into account defined concept names.

4.3.2 Drawbacks of PMA

For simplicity, we assume that occlusions are disallowed and that GCIs are not admitted. As we shall discuss in the following, there are both semantic and computational problems with generalized $\mathfrak{A}_1$-actions: first, they offer an expressivity that is difficult to control and often yields unexpected consequences. Second, reasoning with generalized actions easily becomes undecidable.

Semantic Problems

Clearly, generalized actions such as the trivial $\alpha = (\emptyset, \emptyset, \{a : A \sqcup B\}$ are not deterministic and thus introduce similar complications as discussed for transitive roles in Section 4.2.2. However, disjunction is not the only constructor to introduce non-determinism when allowed in post-conditions. An even "higher degree" of non-determinism is introduced by existential and universal value restrictions:

- If a post-condition contains $a : \exists r.A$ and this assertion was not already satisfied before the execution of the action, then the non-determinism lies in the choice of a witness object, i.e., *any* domain element $x \in \Delta^{\mathcal{I}}$ may be chosen to satisfy $(a^{\mathcal{I}}, x) \in r^{\mathcal{I}}$ and $x \in A^{\mathcal{I}}$ after execution of the action. Note that some such x may already satisfy the former condition, some may satisfy the latter, and some neither.

  The fact that *any* domain element is a potential witness object implies that, e.g., $\mathsf{mary} : \mathsf{Female}$ is not a consequence of applying the action

  $$(\emptyset, \emptyset, \{\mathsf{mary} : \exists \mathsf{has\text{-}child}.\neg\mathsf{Female}\})$$

  in the ABox $\{\mathsf{mary} : \mathsf{Female}\}$—an effect that may not be intended.

- If a post-condition contains $a : \forall r.A$ and this assertion was not already satisfied before the execution of the action, we also have a non-deterministic situation: for each object $x \in \Delta^{\mathcal{I}}$ such that $(a^{\mathcal{I}}, x) \in r^{\mathcal{I}}$ and $x \notin A^{\mathcal{I}}$ holds before the execution of the action, we have to decide whether $(a^{\mathcal{J}}, x) \notin r^{\mathcal{J}}$ or $x \in A^{\mathcal{J}}$ should be satisfied after execution of the action.[2]

  Similarly to the existential case, we may obtain surprising results due to the fact that *any* domain element $x \in \Delta^{\mathcal{I}}$ may satisfy $(a^{\mathcal{I}}, x) \in r^{\mathcal{I}}$ and $x \in A^{\mathcal{I}}$ unless explicitly stated otherwise. This means that, e.g., $\mathsf{tire2}{:}\neg\mathsf{Filled}$ is not a consequence of applying the action

  $$(\emptyset, \emptyset, \{\mathsf{car1}{:}\forall\mathsf{tire}.\mathsf{Filled}\})$$

  in the ABox $\{\mathsf{tire}(\mathsf{car2}, \mathsf{tire2}),\ \mathsf{tire2}{:}\neg\mathsf{Filled}\}$.

Complex concepts with many nested operators may obviously introduce a rather high degree of non-determinism. While simple non-determinism such as the one introduced by transitive roles or post-conditions $a : C \sqcup D$ may be dealt with in a satisfactory way [Lin96, Thi00], none of the mainstream action formalisms allows arbitrary formulas in post-conditions to avoid having to deal with the resulting massive degree of non-determinism. Indeed, most formalisms such as the basic situation calculus restrict themselves to literals in post-conditions [Rei01]—just as our standard $\mathfrak{A}_1$-actions do.

Computational Problems

To illustrate the surprising expressivity of generalized actions, we give an example of "abusing" complex post-conditions to enforce certain properties of models. We will show how to enforce an *infinite grid* structure, which is the essential part of many undecidability proofs.

Let roles x and y denote the horizontal and vertical grid successors respectively. Similarly as in the proof of Theorem 3.2.6, we can simulate (a localized version of the) reflexive-transitive closure of roles. Thus we may assume that a role u connects a fixed individual a (which stands for the $(0, 0)$ point of the grid) with all domain elements that are reachable by chains of x- and y-roles from a. In order to form an infinite (x, y)-grid, it suffices to enforce that: (i) x, y and their inverses are functional; (ii) $x \cdot y$ and $y \cdot x$ commute.

Consider an ABox $\mathcal{A}$, action $\alpha = (\emptyset, \emptyset, \mathsf{post}_\alpha)$, and assertion $\varphi = \neg C(a)$, where

$$\begin{aligned}
\mathcal{A} &:= \{a : \forall u.\neg B\} \\
\mathsf{post}_\alpha &:= \{a : \forall u.\big(\forall x^-.\forall y^-.(\forall x.\forall y.B \sqcap \forall y.\forall x.\neg B) \sqcup B\big)\} \\
C &:= \forall u.B \sqcap \forall u.\big((= 1\ x\ \top) \sqcap (= 1\ y\ \top) \sqcap (= 1\ x^-\ \top) \sqcap (= 1\ y^-\ \top)\big)
\end{aligned}$$

Let $\mathcal{I}$, $\mathcal{I}'$ be interpretations such that $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \Rightarrow_\alpha \mathcal{I}'$, and $\mathcal{I}' \not\models \varphi$ (i.e. $\mathcal{I}' \models C(a)$). Then $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$ and $a^{\mathcal{I}} = a^{\mathcal{I}'}$. As we already discussed, we may assume that u has the following property: for all $d \in \Delta^{\mathcal{I}}$, $(a^{\mathcal{I}}, d) \in u^{\mathcal{I}} \cap u^{\mathcal{I}'}$ – i.e. all domain elements can be reached by u from a. Obviously, then the second conjunct of C implies that (i) holds in $\mathcal{I}'$.

Now we give an intuitive explanation why (ii) holds in $\mathcal{I}'$. Since $\mathcal{I} \models \mathcal{A}$, we have that $B^{\mathcal{I}} = \emptyset$. Note that when satisfying the post-condition post_α, for every domain element d, there is a choice: either change $\neg B$ to B at d, or satisfy $\forall x^-.\forall y^-.(\forall x.\forall y.B \sqcap \forall y.\forall x.\neg B)$. Obviously, the latter is preferred by the change minimizing PMA semantics – if at only one

[2] There may even be cases where it is intended that both conditions are satisfied after action execution; this is, however, not justified by the PMA semantics of generalized actions.

d, $\forall x^-.\forall y^-.(\forall x.\forall y.B \sqcap \forall y.\forall x.\neg B)$ could be satisfied in $\mathcal{I}'$, d could remain in $\neg B$. However, since after action application we have that $B^{\mathcal{I}'} = \Delta^{\mathcal{I}'}$, it means that $\neg(\forall x.\forall y.B \sqcap \forall y.\forall x.\neg B)$ is valid in the frame $\mathcal{F}^3$ that $\mathcal{I}'$ is based upon. It is not difficult to see that in such a frame (where it also holds that x and y are bijections) $x \cdot y$ and $y \cdot x$ commute.

Executability and projection for generalized $\mathfrak{A}_1$-actions under PMA semantics easily becomes undecidable. This holds already for actions formulated in the DL $\mathcal{ALCFI}$ that has been introduced in Section 3.2.2. Recall that $\mathcal{ALCFI}$ is obtained from $\mathcal{ALCQI}$ by limiting numbers occurring in number restrictions to $\{0, 1\}$. This result should be contrasted with the fact that, by Theorem 3.2.1, reasoning with standard $\mathfrak{A}_1$-actions is decidable even for powerful extensions of $\mathcal{ALCFI}$. It is still an open problem whether this undecidability result can be strengthened to simpler description logics, in particular $\mathcal{ALC}$.

The (technically involved) proof of the following theorem is by reduction of the undecidable domino problem [Ber66] to non-consequence and non-executability. It uses the afore presented technique to enforce a $\times$ -grid.

Theorem 4.3.3 ([BLM+05d]). *There exists a generalized atomic action α and an ABox $\mathcal{A}$ formulated in $\mathcal{ALCFI}$ such that the following problems are undecidable, even w.r.t. the empty TBox: given a concept C,*

- *decide whether the assertion $a : C$ is a consequence of applying α in $\mathcal{A}$;*
- *decide whether the composite action α, α' is executable in $\mathcal{A}$, where $\alpha' = (\{C(a)\}, \emptyset, \emptyset)$.*

[3] Frames are introduced in Definition 2.1.6.

Chapter 5

Action Formalism $\mathfrak{A}_2$: Complex Post-Conditions and GCIs

Description logic ontologies frequently consist of general TBoxes that describe complex relations between concepts. For example, a general TBox may describe relevant concepts from the domain of universities such as lecturers, students, courses, and libraries. From the reasoning about actions perspective, TBoxes correspond to state constraints. A general TBox for the university domain could state that every student that is registered for a course has access to a university library. If we execute an action that registers the student Dirk for a computer science course, then after the action Dirk should also have access to a university library to comply with the state constraint imposed by the TBox. Thus, general TBoxes as state constraints induce a ramification problem which we henceforth call the *TBox ramification problem*. Moreover, since there is no clear notion of a concept name "being defined" in a general TBox, an action formalism supporting general TBoxes has to allow for *all concept names* in action post-conditions, as well as *complex concepts*.

We have shown in Section 4.3.1 that an attempt to *automatically* solve the frame problem in the presence of complex concepts in post-conditions (and TBox ramification problem), by adopting a Winslett-style PMA semantics [Win88], leads to semantic and computational problems. Since there appears to be no general automated solution to the TBox ramification problem other than resorting to very inexpressive DLs [GLPR06], we propose to leave it to the designer of an action description to fine-tune the ramifications of the action. A similar approach is taken in the SitCalc and the Fluent Calculus: the ramifications of the action can be controlled by specifying causal relationships between predicates [Lin95, Thi97]. While causality appears to be a satisfactory approach for addressing the ramification problem that is induced by Boolean state constraints, it seems not powerful enough for attacking the ramifications introduced by general TBoxes, which usually involve complex quantification patterns.[1] We therefore advocate a different approach: when describing an action, the user can specify the predicates that can change through the execution of the action, as well as those that cannot change. To allow an adequate fine-tuning of ramifications, we admit complex statements about the change of predicates such as "the concept name A can change from positive to negative only at the individual a, and from negative to positive only where the complex concept C was satisfied before the action was executed".

[1] As discussed in Chapter 3, in $\mathfrak{A}_1$ acyclic TBoxes can be used to capture causality relations between concept names.

The rest of this chapter is organizes as follows. In Section 5.1.1 we introduce the action formalism $\mathfrak{A}_2$ that admits complex concepts in action post-conditions and supports general TBoxes as state constraints. The relation between action formalisms $\mathfrak{A}_1$ and $\mathfrak{A}_2$ is discussed in Section 5.1.3. Since $\mathfrak{A}_2$-actions contain complex occlusion patterns for fine-tuning the ramifications, besides executability and projection, consistency of actions becomes an important reasoning task. In Section 5.1.2, we introduce two notions of consistency (weak and strong). We show that, for many standard propositionally closed DLs, the reasoning problems executability, projection, and weak action consistency in $\mathfrak{A}_2$ are decidable. For DLs contained in $\mathcal{ALCIO}$, we use a type-elimination method in Section 5.2.1 to show that projection in $\mathfrak{A}_2$ is ExpTime-complete. For DLs $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$, in Section 5.2.2 we show how to reduce projection to ABox consequence in $\mathcal{ALCQIO}$ extended with the Boolean operations on roles, thus proving that projection for these logics is co-NExpTime-complete. As a rule of thumb, our results show that reasoning in the action formalism $\mathfrak{A}_2$ instantiated with a description logic $\mathcal{L}$ is of the same complexity as standard reasoning in $\mathcal{LO}$ with general TBoxes. In Section 5.3 we show that strong consistency is undecidable even when the action formalism is instantiated with the basic DL $\mathcal{ALC}$. Finally, in Section 5.4, we discuss the practicability of $\mathfrak{A}_2$ and single out its natural fragment for which reasoning can be passed on to the standard DL reasoners.

5.1 The Formalism

5.1.1 Action Descriptions

The action formalism $\mathfrak{A}_2$ proposed in this chapter is, just like $\mathfrak{A}_1$, not restricted to a particular DL. However, for our complexity results we consider the description logics between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$.

Like in $\mathfrak{A}_1$, the main syntactic ingredients of our approach to reasoning about actions are action descriptions, ABoxes for describing the current knowledge about the state of affairs in the application domain, and *general* TBoxes for describing general knowledge about the application domain similar to state constraints in the SitCalc and Fluent Calculus. On the semantic side, interpretations are used to describe the state of affairs in the application domain. Before we go deeper into the semantics, we introduce the syntax of $\mathfrak{A}_2$-action descriptions. Recall that we use $\mathcal{LO}$ to denote the extension of a description logic $\mathcal{L}$ with nominals. A *concept literal* is a concept name or the negation thereof, and a *role literal* is defined analogously.

Definition 5.1.1 ($\mathfrak{A}_2$-Action). Let $\mathcal{L}$ be a description logic. An *atomic $\mathcal{L}$-$\mathfrak{A}_2$-action* $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ consists of

- a finite set pre of $\mathcal{L}$ ABox assertions, the *pre-conditions*;
- the *occlusion pattern* occ which is a set of mappings $\{\mathsf{occ}_{\varphi_1}, \ldots, \mathsf{occ}_{\varphi_n}\}$ indexed by $\mathcal{L}$ ABox assertions $\varphi_1, \ldots, \varphi_n$ such that each occ_{φ_i} assigns
  - to every concept literal B an $\mathcal{LO}$-concept $\mathsf{occ}_{\varphi_i}(B)$,
  - to every role literal s a finite set $\mathsf{occ}_{\varphi_i}(s)$ of pairs of $\mathcal{LO}$-concepts.

  Moreover, the number of concept literals B such that $\mathsf{occ}_{\varphi_i}(B) \neq \bot$ and role literals s such that $\mathsf{occ}_{\varphi_i}(s) \neq \{(\bot, \bot)\}$, for some $\mathsf{occ}_{\varphi_i} \in \mathsf{occ}$, is finite.

- a finite set post of *conditional post-conditions* of the form φ/ψ, where φ and ψ are $\mathcal{L}$ ABox assertions.

A *composite* $\mathfrak{A}_2$*-action* is a finite sequence of atomic actions $\alpha_1, \ldots, \alpha_k$. △

Definition 5.1.2 (Size of $\mathfrak{A}_2$-Actions). The size of an atomic $\mathfrak{A}_2$-action $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ is defined as $|\alpha| = |\mathsf{pre}| + |\mathsf{occ}| + |\mathsf{post}|$, where:

- $|\mathsf{pre}|$ is the size of the ABox pre,
- $|\mathsf{occ}| = \sum_{\mathsf{occ}_{\varphi_i} \in \mathsf{occ}} \big(\sum_{\mathsf{occ}_{\varphi_i}(B) \neq \bot} |\mathsf{occ}_{\varphi_i}(B)| + \sum_{\mathsf{occ}_{\varphi_i}(s) \neq \{(\bot,\bot)\}} \sum_{(C,D) \in \mathsf{occ}_{\varphi_i}(s)} (|C| + |D|) \big)$
- $|\mathsf{post}| = \sum_{\varphi/\psi \in \mathsf{post}} (|\varphi| + |\psi|)$

The size of a composite $\mathfrak{A}_2$-action $\alpha_1, \ldots, \alpha_n$ is defined with $|\alpha_1, \ldots, \alpha_k| = |\alpha_1| + \cdots + |\alpha_k|$. △

Pre-conditions and post-conditions have the same role as in the action formalism $\mathfrak{A}_1$. The purpose of the occlusion patterns is to control ramifications: they provide a description of where concept and role names may change during the execution of an action. More precisely, suppose $\mathsf{occ} = \{\mathsf{occ}_{\varphi_1}, \ldots, \mathsf{occ}_{\varphi_n}\}$ and $\varphi_{i_1}, \ldots, \varphi_{i_m}$ are the assertions which are true before the action was executed. If A is a concept name, then instances of the concept

$$\mathsf{occ}_{\varphi_{i_1}}(A) \sqcup \cdots \sqcup \mathsf{occ}_{\varphi_{i_m}}(A)$$

may change from A to $\neg A$ during the execution of the action provided, but instances of $\neg(\mathsf{occ}_{\varphi_{i_1}}(A) \sqcup \cdots \sqcup \mathsf{occ}_{\varphi_{i_m}}(A))$ may not. Likewise, instances of

$$\mathsf{occ}_{\varphi_{i_1}}(\neg A) \sqcup \cdots \sqcup \mathsf{occ}_{\varphi_{i_m}}(\neg A)$$

may change from $\neg A$ to A. For role names, $(C, D) \in \mathsf{occ}_{\varphi_{i_k}}(r)$ means that pairs from $C^{\mathcal{I}} \times D^{\mathcal{I}}$ that have been connected by r before the action may lose this connection through the execution of the action, and similarly for the occlusion of negated role names. More details on how occlusions relate to ramifications will be given after we have introduced the semantics.

For defining the semantics in a succinct way, it is convenient to introduce the following abbreviation. For an action α with $\mathsf{occ} = \{\mathsf{occ}_{\varphi_1}, \ldots, \mathsf{occ}_{\varphi_n}\}$, an interpretation $\mathcal{I}$, a concept literal B, and a role literal s, we set

$$\begin{aligned} (\mathsf{occ}(B))^{\mathcal{I}} &:= \bigcup_{\mathcal{I} \models \varphi_i} (\mathsf{occ}_{\varphi_i}(B))^{\mathcal{I}} \\ (\mathsf{occ}(s))^{\mathcal{I}} &:= \bigcup_{(C,D) \in \mathsf{occ}_{\varphi_i}(s),\ \mathcal{I} \models \varphi_i} (C^{\mathcal{I}} \times D^{\mathcal{I}}) \end{aligned}$$

Thus, $\mathsf{occ}(B)^{\mathcal{I}}$ describes those elements of $\Delta^{\mathcal{I}}$ that may change from B to $\neg B$ when going to $\mathcal{I}'$, and similarly for $\mathsf{occ}(s)^{\mathcal{I}}$. By syntax of $\mathfrak{A}_2$-actions, $(\mathsf{occ}(X))^{\mathcal{I}} \neq \emptyset$ only for finitely many concept and role literals X.

Definition 5.1.3 (Action semantics). Let $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ be an atomic $\mathfrak{A}_2$-action and $\mathcal{I}, \mathcal{I}'$ interpretations sharing the same domain and interpretation of all individual names and respecting UNA on individual names. We say that α *may transform* $\mathcal{I}$ *to* $\mathcal{I}'$ w.r.t. a TBox $\mathcal{T}$ ($\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha} \mathcal{I}'$) iff the following holds:

- $\mathcal{I}, \mathcal{I}'$ are models of $\mathcal{T}$;
- for all $\varphi/\psi \in \mathsf{post}$: $\mathcal{I} \models \varphi$ implies $\mathcal{I}' \models \psi$ (written $\mathcal{I}, \mathcal{I}' \models \mathsf{post}$);
- for each $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$, we have

$$\begin{array}{rcll} A^{\mathcal{I}} \setminus A^{\mathcal{I}'} & \subseteq & (\mathsf{occ}(A))^{\mathcal{I}} & \qquad (\neg A)^{\mathcal{I}} \setminus (\neg A)^{\mathcal{I}'} \subseteq (\mathsf{occ}(\neg A))^{\mathcal{I}} \\ r^{\mathcal{I}} \setminus r^{\mathcal{I}'} & \subseteq & (\mathsf{occ}(r))^{\mathcal{I}} & \qquad (\neg r)^{\mathcal{I}} \setminus (\neg r)^{\mathcal{I}'} \subseteq (\mathsf{occ}(\neg r))^{\mathcal{I}} \end{array}$$

The composite $\mathfrak{A}_2$-action $\alpha_1, \ldots, \alpha_n$ *may transform* $\mathcal{I}$ to $\mathcal{I}'$ w.r.t. $\mathcal{T}$ ($\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_n} \mathcal{I}'$) iff there are models $\mathcal{I}_0, \ldots, \mathcal{I}_n$ of $\mathcal{T}$ with $\mathcal{I} = \mathcal{I}_0$, $\mathcal{I}' = \mathcal{I}_n$, and $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_i$ for $1 \leq i \leq n$. △

Example 5.1.4. *Let us consider an example in order to to explain how occlusions provide a way to control the ramifications induced by general TBoxes. The TBox $\mathcal{T}$ contains the following GCIs which say that everybody registered for a course has access to a university library, and that every university has a library:*

$$\begin{array}{rcl} \exists \mathsf{registered\_for.Course} & \sqsubseteq & \exists \mathsf{access\_to.Library} \\ \mathsf{University} & \sqsubseteq & \exists \mathsf{has\_facility.Library} \end{array}$$

The upper GCI cannot be expressed in terms of an acyclic TBox and is thus outside the scope of the formalism $\mathfrak{A}_1$. The ABox $\mathcal{A}$ which describes the current state of the world (in an incomplete way) says that computer science is a course held at TU Dresden, SLUB is the library of TU Dresden, and Dirk is neither registered for a course nor has access to a library:

$$\begin{array}{lll} \mathsf{Course(cs)} & \mathsf{held\_at(cs, tud)} & \neg\exists \mathsf{registered\_for.Course(dirk)} \\ \mathsf{University(tud)} & \mathsf{has\_facility(tud, slub)} & \neg\exists \mathsf{access\_to.Library(dirk)} \\ \mathsf{Library(slub)} & & \end{array}$$

The action

$$\alpha := (\emptyset, \mathsf{occ}, \{\mathsf{taut}/\mathsf{registered\_for(dirk, cs)}\})$$

describes the registration of Dirk for the computer science course. For simplicity, the set of pre-conditions is empty and taut *is some ABox assertion that is trivially satisfied, say* $\top(\mathsf{cs})$*. To obtain* occ*, we may start by strictly following the law of inertia, i.e., requiring that the only changes are those that are explicitly stated in the post-condition. Thus,* occ *consists of just one mapping* $\mathsf{occ}_{\mathsf{taut}}$ *such that*

$$\mathsf{occ}_{\mathsf{taut}}(\neg\mathsf{registered\_for}) := \{(\{\mathsf{dirk}\}, \{\mathsf{cs}\})\}$$

and all concept and role literals except $\neg$registered_for *are mapped to* $\bot$ *and* $\{(\bot, \bot)\}$*, respectively. This achieves the desired effect that only the pair* (dirk, cs) *can be added to "*registered_for*" and nothing else can be changed.*

It is not hard to see that this attempt to specify occlusions for α is too strict. Intuitively, not allowing any changes is appropriate for Course*,* Library*,* University*,* held_at*,* has_facility *and their negations since the action should have no impact on these predicates. However, not letting* $\neg$access_to *change leads to a problem with the ramifications induced by the TBox: as Dirk has no access to a library before the action and* $\neg$access_to *is not allowed to change, he cannot have access to a library after execution of the action as required by the TBox. Thus,*

the action is inconsistent in the following sense: there is no model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ and model $\mathcal{I}'$ of $\mathcal{T}$ such that $\mathcal{I} \Rightarrow_\alpha^\mathcal{T} \mathcal{I}'$. To take care of the TBox ramifications and regain consistency, we can modify occ. *One option is to set*

$$\mathsf{occ}_{\mathsf{taut}}(\neg \mathsf{access\_to}) := \{(\{\mathsf{dirk}\}, \mathsf{Library})\}$$

and thus allow Dirk to have access to a library after the action. Another option is to set

$$\mathsf{occ}_{\mathsf{taut}}(\neg \mathsf{access\_to}) := \{(\{\mathsf{dirk}\}, \{\mathsf{slub}\})\}$$

which allows Dirk to have access to SLUB after the action, but not to any other library.

Two remarks regarding this example are in order. First, the occlusion occ consists only of a single mapping $\mathsf{occ}_{\mathsf{taut}}$. The reason for this is that there is only a single post-condition in the action. If we have different post-conditions φ/ψ and φ'/ψ such that φ and φ' are not equivalent, there will usually be different occlusion mappings (indexed with φ and φ') to deal with the ramifications that the TBox induces for these post-conditions. Second, the example explains the need for extending $\mathcal{L}$ to $\mathcal{LO}$ when describing occlusions (c.f. Definition 5.1.1): without nominals, we would not have been able to properly formulate the occlusions although all other parts of the example are formulated without using nominals (as a concept-forming operator).

5.1.2 Reasoning about Actions

As illustrated by the example, it is important for the action designer to decide consistency of actions to detect ramification problems that are not properly addressed by the occlusions. In the following, we propose two notions of consistency.[2]

Definition 5.1.5 (Consistency). Let $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ be an atomic action and $\mathcal{T}$ a TBox. We say that

- α is *weakly consistent with* $\mathcal{T}$ iff there are models $\mathcal{I}, \mathcal{I}'$ of $\mathcal{T}$ such that $\mathcal{I} \models \mathsf{pre}$ and $\mathcal{I} \Rightarrow_\alpha^\mathcal{T} \mathcal{I}'$.
- α is *strongly consistent with* $\mathcal{T}$ iff for all models $\mathcal{I}$ of $\mathcal{T}$ and pre, there is a model $\mathcal{I}'$ of $\mathcal{T}$ such that $\mathcal{I} \Rightarrow_\alpha^\mathcal{T} \mathcal{I}'$.

△

Intuitively, strong consistency is the most desirable form of consistency: if the preconditions of an action are satisfied by an interpretation $\mathcal{I}$, then the action can transform $\mathcal{I}$ into a new interpretation $\mathcal{I}'$. Unfortunately, strong consistency will turn out to be undecidable. For this reason we introduce also weak consistency, which is still sufficient to detect serious ramification problems. In the example above, the first attempt to define the occlusions results in an action that is not even weakly consistent. After each of the two possible modifications, the action is strongly consistent. We will see later that weak consistency is decidable while strong consistency is not.

We do not define the reasoning tasks *executability* and *projection* in $\mathfrak{A}_2$ as we assume that they are defined in the same way as for $\mathfrak{A}_1$-actions in Definition 3.1.6 (but w.r.t. the transition

[2]Note that (strong) consistency was also defined in the action formalism $\mathfrak{A}_1$, but was not treated as a proper reasoning task.

relation $\Rightarrow_\alpha^{\mathcal{T}}$ from Definition 5.1.3). To make sure that a composite action $\alpha = \alpha_1, \ldots, \alpha_n$ can be successfully executed, α has to be executable and the atomic actions $\alpha_1, \ldots, \alpha_n$ have to be strongly consistent: without strong consistency, it could be that although the action α is executable w.r.t. the ABox $\mathcal{A}$ describing the knowledge about the current state of the world, the actual state of the world $\mathcal{I}$ is such that there is no interpretation $\mathcal{I}'$ with $\mathcal{I} \Rightarrow_\alpha^{\mathcal{T}} \mathcal{I}'$. Even worse, such a situation may arise also after we have already executed some of the atomic actions in the sequence α.

It is not difficult to see that the action formalism $\mathfrak{A}_2$ is a generalization of the the action formalism $\mathfrak{A}_1$ that is introduced in Chapter 3, for details see Section 5.1.3. Executability and projection can be mutually reduced in a polynomial time, and the proof is the same as the proof of Lemma 3.1.7.

It can also be seen that an action α weakly consistent with a TBox $\mathcal{T}$ iff $\bot(a)$ is not a consequence of applying α in pre w.r.t. $\mathcal{T}$; (ii) φ is a consequence of applying $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff the action $(\mathcal{A} \cup \mathsf{pre}, \mathsf{occ}, \mathsf{post} \cup \{\mathsf{taut}/\neg\varphi\})$ is not weakly consistent with $\mathcal{T}$. Here and in the coming sections, taut denotes an arbitrary valid assertion, such as $\top(a)$. We have shown that weak consistency can be reduced to (non-)projection and vice versa and hence complexity results carry over from one to the other. In the following sections, we will concentrate on projection.

5.1.3 Relation to $\mathfrak{A}_1$

We show that the action formalism $\mathfrak{A}_2$ introduced in this chapter is a generalization of the action formalism $\mathfrak{A}_1$ introduced in Chapter 3. Obviously, general TBoxes generalize acyclic TBoxes. Actions in Chapter 3 are more restricted than the actions introduced in this paper. For example, the former do not allow post-conditions $\varphi/C(a)$ with C a complex concept or defined concept name. Thus, in post-conditions of this form, C must be a primitive concept name.

Let $\mathcal{T}$ be an acyclic TBox and $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ an $\mathfrak{A}_1$-action. We use P to denote the set $\{\varphi \mid \exists\psi : \varphi/\psi \in \mathsf{post}\}$. We now construct an $\mathfrak{A}_2$-action α' that is equivalent to α in the sense that for all models $\mathcal{I}, \mathcal{I}'$ of $\mathcal{T}$, $\mathcal{I} \Rightarrow_\alpha^{\mathfrak{A}_1,\mathcal{T}} \mathcal{I}'$ iff $\mathcal{I} \Rightarrow_{\alpha'}^{\mathfrak{A}_2,\mathcal{T}} \mathcal{I}'$ [3]. Set $\alpha' := (\mathsf{pre}, \mathsf{occ}', \mathsf{post})$, where the occlusion pattern is defined as

$$\mathsf{occ}' := \{\mathsf{occ}_{\mathsf{taut}}\} \cup \{\mathsf{occ}_\varphi \mid \varphi/\psi \in \mathsf{post}\},$$

with the components $\mathsf{occ}_{\mathsf{taut}}$ and occ_φ, $\varphi/\psi \in \mathsf{post}$ are as follows:

- for every primitive concept name A and $\varphi \in P$,
  - $\mathsf{occ}_{\mathsf{taut}}(A) := \mathsf{occ}_{\mathsf{taut}}(\neg A) := \bigsqcup_{A(a) \in \mathsf{occ}} \{a\}$;
  - $\mathsf{occ}_\varphi(A) := \bigsqcup_{\varphi/\neg A(a) \in \mathsf{post}} \{a\}$;
  - $\mathsf{occ}_\varphi(\neg A) := \bigsqcup_{\varphi/A(a) \in \mathsf{post}} \{a\}$;
- for every role name r and $\varphi \in P$,

[3] We added the superscripts $\cdot^{\mathfrak{A}_1}$ and $\cdot^{\mathfrak{A}_2}$ in order to differentiate between transition relations $\Rightarrow_\alpha^{\mathcal{T}}$ in $\mathfrak{A}_1$ and $\mathfrak{A}_2$.

- $\mathsf{occ}_{\mathsf{taut}}(r) := \mathsf{occ}_{\mathsf{taut}}(\neg r) := \bigcup_{r(a,b)\in\mathsf{occ}} \{(\{a\},\{b\})\}$;
- $\mathsf{occ}_{\varphi}(r) := \bigcup_{\varphi/\neg r(a,b)\in\mathsf{post}} \{(\{a\},\{b\})\}$;
- $\mathsf{occ}_{\varphi}(\neg r) := \bigcup_{\varphi/r(a,b)\in\mathsf{post}} \{(\{a\},\{b\})\}$;

- for every defined concept name C and $\varphi \in P$,

$$\mathsf{occ}_{\mathsf{taut}}(C) = \mathsf{occ}_{\mathsf{taut}}(\neg C) = \mathsf{occ}_{\varphi}(C) = \mathsf{occ}_{\varphi}(\neg C) := \top.$$

The following lemma shows that α and α' are indeed equivalent.

Lemma 5.1.6. *For all models $\mathcal{I}$ and $\mathcal{I}'$ of $\mathcal{T}$, $\mathcal{I} \Rightarrow_{\alpha}^{\mathfrak{A}_1,\mathcal{T}} \mathcal{I}'$ iff $\mathcal{I} \Rightarrow_{\alpha'}^{\mathfrak{A}_2,\mathcal{T}} \mathcal{I}'$.*

Proof. "⇒". Let $\mathcal{I} \Rightarrow_{\alpha}^{\mathfrak{A}_1,\mathcal{T}} \mathcal{I}'$. We show that this implies $\mathcal{I} \Rightarrow_{\alpha'}^{\mathfrak{A}_2,\mathcal{T}} \mathcal{I}'$. This amounts to verifying the conditions from Definition 5.1.3.

- $\mathcal{I}$ and $\mathcal{I}'$ are models of $\mathcal{T}$ and share the same domains and interpret individuals in the same way by definition of $\Rightarrow_{\alpha}^{\mathfrak{A}_1,\mathcal{T}}$.

- $\mathcal{I},\mathcal{I}' \models \mathsf{post}$: Let $\varphi/\psi \in \mathsf{post}$ and $\mathcal{I} \models \varphi$. We assume that $\psi = A(a)$, as the cases $\neg A(a)$, $r(a,b)$, or $\neg r(a,b)$ can be treated similarly. Then it holds that $a^{\mathcal{I}} \in \alpha_+^{\mathcal{I}}(A)$ and $a^{\mathcal{I}} \in D_{\alpha}^{\mathcal{I}}(A)$. Since $\alpha_+^{\mathcal{I}}(A) \cap \alpha_-^{\mathcal{I}}(A) = \emptyset$, we have that $a^{\mathcal{I}} \notin \alpha_-^{\mathcal{I}}(A)$. Then, since $A^{\mathcal{I}'} \cap D_{\alpha}^{\mathcal{I}}(A) = ((A^{\mathcal{I}} \cup \alpha_+^{\mathcal{I}}(A)) \setminus \alpha_-^{\mathcal{I}}(A)) \cap D_{\alpha}^{\mathcal{I}}(A)$, we have that $a^{\mathcal{I}'} = a^{\mathcal{I}} \in A^{\mathcal{I}'}$ and thus, $\mathcal{I}' \models \psi$.

- Let A be a concept name.

  - If A is primitive, then for every $x \in A^{\mathcal{I}} \setminus A^{\mathcal{I}'}$, we have either $x \in \alpha_-^{\mathcal{I}}(A)$ or $x \notin D_{\alpha}^{\mathcal{I}}(A)$ since $A^{\mathcal{I}'} \cap D_{\alpha}^{\mathcal{I}}(A) = ((A^{\mathcal{I}} \cup \alpha_+^{\mathcal{I}}(A)) \setminus \alpha_-^{\mathcal{I}}(A)) \cap D_{\alpha}^{\mathcal{I}}(A)$. From either of them, we can get $x \in (\mathsf{occ}'(A))^{\mathcal{I}}$, and thus $A^{\mathcal{I}} \setminus A^{\mathcal{I}'} \subseteq (\mathsf{occ}'(A))^{\mathcal{I}}$ holds. It can be shown analogously that $(\neg A)^{\mathcal{I}} \setminus (\neg A)^{\mathcal{I}'} \subseteq (\mathsf{occ}'(\neg A))^{\mathcal{I}}$.
  - If A is defined in $\mathcal{T}$, we have $(\mathsf{occ}'(A))^{\mathcal{I}} = (\mathsf{occ}'(\neg A))^{\mathcal{I}} = \Delta_{\mathcal{I}}$. Thus, $A^{\mathcal{I}} \setminus A^{\mathcal{I}'} \subseteq (\mathsf{occ}'(A))^{\mathcal{I}}$ and $A^{\mathcal{I}'} \setminus A^{\mathcal{I}} \subseteq (\mathsf{occ}'(\neg A))^{\mathcal{I}}$

- Let r be a role name. We need to show that $r^{\mathcal{I}} \setminus r^{\mathcal{I}'} \subseteq (\mathsf{occ}'(r))^{\mathcal{I}}$ and $(\neg r)^{\mathcal{I}} \setminus (\neg r)^{\mathcal{I}'} \subseteq (\mathsf{occ}'(\neg r))^{\mathcal{I}}$. This case is analogous to the one of primitive concept names.

"⇐". Assume $\mathcal{I} \Rightarrow_{\alpha'}^{\mathfrak{A}_2,\mathcal{T}} \mathcal{I}'$. We show that then, $\mathcal{I} \Rightarrow_{\alpha}^{\mathfrak{A}_1,\mathcal{T}} \mathcal{I}'$. This amounts to checking the conditions from Definition 3.1.4.

- $\mathcal{I}$ and $\mathcal{I}'$ are models of $\mathcal{T}$ and share the same domains and interpret individuals in the same way by definition of $\Rightarrow_{\alpha'}^{\mathfrak{A}_2,\mathcal{T}}$;

- $\alpha_+^{\mathcal{I}}(A) \cap A_- = \emptyset$: Assume the opposite. Then there are $\varphi_1/A(a), \varphi_2/\neg A(a) \in \mathsf{post}$ such that $\mathcal{I} \models \varphi_1$ and $\mathcal{I} \models \varphi_2$. But then $\mathcal{I}' \models A(a)$, $\mathcal{I}' \models \neg A(a)$, which is impossible. Analogously, we have that $\alpha_+^{\mathcal{I}}(r) \cap \alpha_-^{\mathcal{I}}(r) = \emptyset$.

- for every primitive concept name A, it holds that $A^{\mathcal{I}'} \cap D_{\alpha}^{\mathcal{I}}(A) = ((A^{\mathcal{I}} \cup \alpha_+^{\mathcal{I}}(A)) \setminus \alpha_-^{\mathcal{I}}(A)) \cap D_{\alpha}^{\mathcal{I}}(A)$:

  Consider an arbitrary $x \in D_{\alpha}^{\mathcal{I}}(A)$ with $x \in ((A^{\mathcal{I}} \cup \alpha_+^{\mathcal{I}}(A)) \setminus \alpha_-^{\mathcal{I}}(A))$. Since $\alpha_+^{\mathcal{I}}(A) \cap \alpha_-^{\mathcal{I}}(A) = \emptyset$, there are two cases to consider:

– If $x \notin \alpha_+^{\mathcal{I}}(A)$, then $x \in A^{\mathcal{I}} \setminus \alpha_-^{\mathcal{I}}(A)$. Thus $x \in D_\alpha^{\mathcal{I}}(A) \setminus (\alpha_+^{\mathcal{I}}(A) \cup \alpha_-^{\mathcal{I}}(A))$. By definition of occ', we have that $x \notin (\mathsf{occ}'(A))^{\mathcal{I}}$. By definition of $\Rightarrow_{\alpha'}^{\mathfrak{A}_2,\mathcal{T}}$, $x \in A^{\mathcal{I}} \setminus (\mathsf{occ}'(A))^{\mathcal{I}}$ implies $x \in A^{\mathcal{I}'}$.

– If $x \in \alpha_+^{\mathcal{I}}(A)$, then there is a $\varphi/A(a) \in \mathsf{post}$ with $a^{\mathcal{I}} = x$ and $\mathcal{I} \models \varphi$. By definition of $\mathcal{I} \Rightarrow_{\alpha'}^{\mathfrak{A}_2,\mathcal{T}} \mathcal{I}'$ we obtain that $\mathcal{I}' \models A(a)$. Thus, $x \in A^{\mathcal{I}'}$.

For the other direction, consider an arbitrary x with $x \in A^{\mathcal{I}'} \cap D_\alpha^{\mathcal{I}}(A)$. Then $x \in A^{\mathcal{I}'}$ implies that $x \notin \alpha_-^{\mathcal{I}}(A)$ by definition of $\mathcal{I} \Rightarrow_{\alpha'}^{\mathfrak{A}_2,\mathcal{T}} \mathcal{I}'$. It is enough to show $x \in A^{\mathcal{I}} \cup \alpha_+^{\mathcal{I}}(A)$. Assume that $x \notin A^{\mathcal{I}}$. We show that then $x \in \alpha_+^{\mathcal{I}}(A)$. From $x \notin A^{\mathcal{I}}$ and $x \in A^{\mathcal{I}'}$ we obtain that $x \in (\mathsf{occ}'(\neg A))^{\mathcal{I}}$ by definition of $\Rightarrow_{\alpha'}^{\mathfrak{A}_2,\mathcal{T}}$. Thus $x \in D_\alpha^{\mathcal{I}}(A) \setminus \alpha_-^{\mathcal{I}}(A)$ implies that $x \in \alpha_+^{\mathcal{I}}(A)$.

- It can be shown that $r^{\mathcal{I}'} \cap D_\alpha^{\mathcal{I}}(r) = ((r^{\mathcal{I}} \cup \alpha_+^{\mathcal{I}}(r)) \setminus \alpha_-^{\mathcal{I}}(r)) \cap D_\alpha^{\mathcal{I}}(r)$ for every role name r analogously.

❑

5.2 Deciding Projection

5.2.1 Projection in ExpTime

We show that projection and weak consistency are ExpTime-complete for $\mathfrak{A}_2$-actions formulated in $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCI}$, $\mathcal{ALCIO}$. Thus, in these DLs reasoning about actions is not more difficult than the standard DL reasoning problems such as concept satisfiability and subsumption w.r.t. general TBoxes. The complexity results established in this section are obtained by proving that projection in $\mathcal{ALCIO}$ is in ExpTime. We use a Pratt-style type elimination technique as first proposed in [Pra79]. From this point, we may omit "$\mathfrak{A}_2$-" when talking about $\mathfrak{A}_2$-actions in this chapter.

Let $\alpha_1, \ldots, \alpha_n$ be a composite action with $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ for $1 \leq i \leq n$, $\mathcal{T}$ a general TBox, $\mathcal{A}_0$ an ABox and φ_0 an assertion. We want to decide whether φ_0 is a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. In what follows, we call $\alpha_1, \ldots, \alpha_n$, $\mathcal{T}$, $\mathcal{A}_0$ and φ_0 the *input*. W.l.o.g. we make the following assumptions:

- concepts used in the input are built only from the constructors $\{a\}$, $\neg$, $\sqcap$, and $\exists r.C$;
- φ_0 is of the form $\varphi_0 = C_0(a_0)$, where C_0 is a (complex) concept;
- $\mathcal{A}_0$ and $\alpha_1, \ldots, \alpha_n$ contain only concept assertions.

The last two assumptions can be made because every assertion $r(a, b)$ can be replaced with $(\exists r.\{b\})(a)$, and every $\neg r(a, b)$ with $(\neg\exists r.\{b\})(a)$.

Before we can describe the algorithm, we introduce several notions and abbreviations. With Sub, we denote the set of subconcepts of the concepts which occur in the input. With Ind, we denote the set of individual names used in the input, and set $\mathsf{Nom} := \{\{a\} \mid a \in \mathsf{Ind}\}$. The algorithm that we give in the following checks for the existence of a counter-model witnessing that φ_0 is *not* a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. Such a counter-model consists of $n+1$ interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ such that $\mathcal{I}_0 \models \mathcal{A}_0$, $\mathcal{I}_0 \Rightarrow_{\alpha_1}^{\mathcal{T}} \mathcal{I}_1, \ldots, \mathcal{I}_{n-1} \Rightarrow_{\alpha_n}^{\mathcal{T}} \mathcal{I}_n$, and $\mathcal{I}_n \not\models \varphi_0$. To distinguish the extension of concept and role names in the different

interpretations, we introduce concept names $A^{(i)}$ and role names $r^{(i)}$ for every concept name A and role name r used in the input, and every $i \leq n$. For a concept $C \in \mathsf{Sub}$ that is not a concept name, we use $C^{(i)}$, $i \leq n$, to denote the concept obtained by replacing all concept names A and role names r occurring in C by $A^{(i)}$ and $r^{(i)}$ respectively. We define the set of concepts Cl as:

$$\mathsf{Cl} = \bigcup_{i \leq n} \{C^{(i)}, \neg C^{(i)} \mid C \in \mathsf{Sub} \cup \mathsf{Nom}\}$$

The notion of a type plays a central role in the projection algorithm to be devised.

Definition 5.2.1. A set of concepts $t \subseteq \mathsf{Cl}$ is a *type* for Cl iff it satisfies the following conditions:

- for all $\neg D \in \mathsf{Cl}$: $\neg D \in t$ iff $D \notin t$;
- for all $D \sqcap E \in \mathsf{Cl}$: $D \sqcap E \in t$ iff $\{D, E\} \subseteq t$;
- for all $C \sqsubseteq D \in \mathcal{T}$ and $i \leq n$, $C^{(i)} \in t$ implies $D^{(i)} \in t$;
- for all $a, b \in \mathsf{Ind}$: $\{\{a\}, \{b\}\} \subseteq t$ implies $a = b$.

A type is *anonymous* if it does not contain a nominal. Let $\mathfrak{T}_{\mathsf{ano}}$ be the set of all anonymous types. △

Intuitively, a type describes the concept memberships of a domain element in all $n + 1$ interpretations. Our algorithm starts with a set containing (almost) all types, then repeatedly eliminates those types that cannot be realized in a counter-model witnessing that φ_0 is not a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$, and finally checks whether the surviving types give rise to such a counter-model. The picture is slightly complicated by the presence of ABoxes and nominals. These are treated via core type sets to be introduced next.

Definition 5.2.2. $\mathfrak{T}_S$ is a *core type set* iff $\mathfrak{T}_S$ is a minimal set of types such that, for all $a \in \mathsf{Ind}$, there is a $t \in \mathfrak{T}_S$ with $\{a\} \in \mathfrak{T}_S$.

A core type set $\mathfrak{T}_S$ is called *proper* if the following conditions are satisfied:

1. for all $C(a) \in \mathcal{A}_0$, $\{a\} \in t \in \mathfrak{T}_S$ implies $C^{(0)} \in t$;
2. for all $C(a)/D(b) \in \mathsf{post}_i$, $1 \leq i \leq n$: if there is a $t \in \mathfrak{T}_S$ with $\{\{a\}, C^{(i-1)}\} \subseteq t$ then there is a $t' \in \mathfrak{T}_S$ with $\{\{b\}, D^{(i)}\} \subseteq t'$.

△

Intuitively, a core type set carries information about the "named" part of the interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$, where the named part of an interpretation consists of those domain elements that are identified by nominals. Let m be the size of the input. It is not difficult to check that the number of core type sets is exponential in m. Also, checking whether a core type set is proper can be done in linear time.

The following definition specifies the conditions under which a type is eliminated. We start with introducing some convenient abbreviations. Consider an action $\alpha_\ell = (\mathsf{pre}_\ell, \mathsf{occ}_\ell, \mathsf{post}_\ell)$. For a role name r and $\mathsf{occ}_\varphi \in \mathsf{occ}_\ell$, we set $\mathsf{occ}_\varphi(r^-) := \{(Y, X) \mid (X, Y) \in \mathsf{occ}_\varphi(r)\}$, and analogously for $\mathsf{occ}(\neg r^-)$. Let t, t' be types, $\mathfrak{T}$ a set of types, $C(a)$ an ABox assertion, B a concept literal, and s a role literal. We write

- $\mathfrak{T} \models C(a)$ if there exists a $t \in \mathfrak{T}$ with $\{\{a\}, C\} \subseteq t$;
- $t, \mathfrak{T} \models \mathsf{occ}_\ell(B)$ if there is an $\mathsf{occ}_{C(a)} \in \mathsf{occ}_\ell$ such that (i) $\mathfrak{T} \models C(a)$ and (ii) for the concept $D = \mathsf{occ}_{C(a)}(B)$, we have $D^{(\ell)} \in t$;
- $t, t', \mathfrak{T} \models \mathsf{occ}_\ell(s)$ if there is an $\mathsf{occ}_{C(a)} \in \mathsf{occ}_\ell$ and a pair $(D, E) \in \mathsf{occ}_{C(a)}(s)$ such that (i) $\mathfrak{T} \models C(a)$, (ii) $D^{(\ell)} \in t$, and (iii) $E^{(\ell)} \in t'$.

Intuitively, $t, \mathfrak{T} \models \mathsf{occ}_\ell(B)$ states that when the action α_ℓ is executed in a model that realizes only types from $\mathfrak{T}$, then instances of t may change from B to $\neg B$.

Recall that for a role r, we use $\mathsf{Inv}(r)$ to denote r^- if r is a role name and s if $r = s^-$.

Definition 5.2.3. Let $\mathfrak{T}$ be a set of types for CI. Then a type $t \in \mathfrak{T}$ is *good in* $\mathfrak{T}$ iff

- for all concept names A and $i < n$:

  (C1) if $\{A^{(i)}, \neg A^{(i+1)}\} \subseteq t$, then $t, \mathfrak{T} \models \mathsf{occ}_{i+1}(A)$;
  (C2) if $\{\neg A^{(i)}, A^{(i+1)}\} \subseteq t$, then $t, \mathfrak{T} \models \mathsf{occ}_{i+1}(\neg A)$.

- for all $(\exists r.C)^{(i)} \in t$, there exists a type $t' \in \mathfrak{T}$ and a set $\rho \subseteq \{0, \dots, n\}$ such that for all $\ell \leq n$, the following conditions are satisfied:

  (R1) $C^{(i)} \in t'$ and $i \in \rho$;
  (R2) if $(\neg\exists r.D)^{(\ell)} \in t$ and $\ell \in \rho$, then $\neg D^{(\ell)} \in t'$;
  (R3) if $(\neg\exists \mathsf{Inv}(r).D)^{(\ell)} \in t'$ and $\ell \in \rho$, then $\neg D^{(\ell)} \in t$;
  (R4) if $n > \ell \in \rho$ and $\ell + 1 \notin \rho$ then $t, t', \mathfrak{T} \models \mathsf{occ}_{\ell+1}(r)$;
  (R5) if $n > \ell \notin \rho$ and $\ell + 1 \in \rho$ then $t, t', \mathfrak{T} \models \mathsf{occ}_{\ell+1}(\neg r)$.

△

Intuitively, the above definition checks whether there can be any instances of t in an interpretation in which all domain elements have a type in $\mathfrak{T}$. More precisely, t' is the type that is needed to satisfy the existential restriction $(\exists r.C)^{(i)} \in t$. The set ρ determines the extension of the role r: if $\ell \in \rho$, then the instance of t' that we introduce as a witness for $(\exists r.C)^{(i)}$ is reachable via an r-edge from the instance of t in the interpretation $\mathcal{I}_\ell$.

The type elimination algorithm is given in a pseudo-code notation in Figure 5.1, where C_0 is the concept from the ABox assertion $\varphi_0 = C_0(a_0)$.

Lemma 5.2.4. *$\mathcal{ALCIO}$-$\mathsf{elim}(\mathcal{A}_0, \mathcal{T}, \alpha_1, \dots, \alpha_n, \varphi_0)$ returns true iff φ_0 is a consequence of applying $\alpha_1, \dots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$.*

Proof. "⇒". We prove this direction by contraposition. Assume that φ_0 is not a consequence of applying $\alpha_1, \dots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. Then there are $\mathcal{I}_0, \dots, \mathcal{I}_n$ such that $\mathcal{I}_i \Rightarrow^{\mathcal{T}}_{\alpha_{i+1}} \mathcal{I}_{i+1}$ for all $i < n$, $\mathcal{I}_0 \models \mathcal{A}_0$, and $\mathcal{I}_n \not\models C_0(a_0)$. We show that the algorithm returns false in this case. Let $\Delta := \Delta^{\mathcal{I}_0}$ $(= \dots = \Delta^{\mathcal{I}_n})$ and $x \in \Delta$. We define

$$t_{\mathsf{CI}}(x) := \{C^{(i)} \in \mathsf{CI} \mid x \in C^{\mathcal{I}_i} \text{ for some } i \leq n\}.$$

Claim 1. For all $x \in \Delta$, $t_{\mathsf{CI}}(x)$ is a type for CI.

Proof: For all $x \in \Delta$ and all $j \leq n$, the following holds:

$\mathcal{ALCIO}$-elim$(\mathcal{A}_0, \mathcal{T}, \alpha_1, \ldots, \alpha_n, \varphi_0)$
  for all proper core type sets $\mathfrak{T}_S$ do
    $i := 0$;
    $\mathfrak{T}_0 := \mathfrak{T}_S \cup \mathfrak{T}_{\mathsf{ano}}$
    repeat
      $\mathfrak{T}_{i+1} := \{t \in \mathfrak{T}_i \mid t \text{ is good in } \mathfrak{T}_i\}$;
      $i := i + 1$;
    until $\mathfrak{T}_i = \mathfrak{T}_{i-1}$;
    if $\mathfrak{T}_S \subseteq \mathfrak{T}_i$ and there is a $t \in \mathfrak{T}_i$ with $\{\{a_0\}, \neg C_0^{(n)}\} \subseteq t$ then
      return false
    endif
  endfor
  return true

Figure 5.1: The type elimination algorithm.

- for all $\neg C^{(j)} \in \mathsf{Cl}$: $(\neg C)^{(j)} \in t_{\mathsf{Cl}}(x)$ iff $x \in (\neg C)^{\mathcal{I}_j}$ iff $x \notin C^{\mathcal{I}_j}$ iff $C^{(j)} \notin t_{\mathsf{Cl}}(x)$;
- for all $(C \sqcap D)^{(j)} \in \mathsf{Cl}$: $(C \sqcap D)^{(j)} \in t_{\mathsf{Cl}}(x)$ iff $x \in (C \sqcap D)^{\mathcal{I}_j}$ iff $x \in C^{\mathcal{I}_j}$ and $x \in D^{\mathcal{I}_j}$ iff $C^{(j)} \in t_{\mathsf{Cl}}(x)$ and $D^{(j)} \in t_{\mathsf{Cl}}(x)$ iff $\{C, D\} \subseteq t_{\mathsf{Cl}}(x)$;
- for all $C \sqsubseteq D \in \mathcal{T}$, $\mathcal{I}_j \models \mathcal{T}$ implies that if $x \in C^{\mathcal{I}_j}$, then $x \in D^{\mathcal{I}_j}$. Thus, $C^{(j)} \in t_{\mathsf{Cl}}(x)$ implies $D^{(j)} \in t_{\mathsf{Cl}}(x)$;
- for all $a, b \in \mathsf{Ind}$, $\{\{a\}, \{b\}\} \subseteq t_{\mathsf{Cl}}(x)$ iff $x = a^{\mathcal{I}_0} = b^{\mathcal{I}_0}$. Since $\mathcal{I}_0$ satisfies UNA on individual names, this can be the case only if $a = b$.

This finishes the proof of Claim 1. We set $\mathfrak{T}_S := \{t_{\mathsf{Cl}}(a^{\mathcal{I}_0}) \mid a \in \mathsf{Ind}\}$ and $\mathfrak{T} := \{t_{\mathsf{Cl}}(x) \mid x \in \Delta\}$. Then we have the following:

Claim 2: $\mathfrak{T}_S$ is a proper core type set.

Proof: By the definition of $\mathfrak{T}_S$, it is easy to see that it is a core type set. Moreover, it is proper:

1. for all $C(a) \in \mathcal{A}_0$, $\mathcal{I}_0 \models \mathcal{A}_0$ implies $\mathcal{I}_0 \models C(a)$. Thus, we know that $a^{\mathcal{I}_0} \in C^{\mathcal{I}_0}$. $\{a\} \in t \in \mathfrak{T}_S$ implies $C^{(0)} \in t (= t_{\mathsf{Cl}}(a^{\mathcal{I}_0}))$;
2. for all $C(a)/D(b) \in \mathsf{post}_i$, $1 \leq i \leq n$: if there is a $t \in \mathfrak{T}_S$ with $\{\{a\}, C^{(i-1)}\} \subseteq t$ then $a^{\mathcal{I}_{i-1}} \in C^{\mathcal{I}_{i-1}}$. Thus, $\mathcal{I}_{i-1} \models C(a)$. Since $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_i$, we know that $\mathcal{I}_i \models D(b)$. Hence, $b^{\mathcal{I}_i} \in D^{\mathcal{I}_i}$. Let $t' := t_{\mathsf{Cl}}(b^{\mathcal{I}_i})$. $b^{\mathcal{I}_i} \in D^{\mathcal{I}_i}$ implies $\{\{b\}, D^{(i)}\} \subseteq t'$.

Claim 3. For every $t \in \mathfrak{T}$, t is good in $\mathfrak{T}$.

Proof: Let $t = t_{\mathsf{Cl}}(x)$, for an $x \in \Delta$.

(i) for all concept names A and $i < n$, $\{A^{(i)}, \neg A^{(i+1)}\} \subseteq t_{\mathsf{Cl}}(x)$ holds iff $x \in A^{\mathcal{I}_i} \setminus A^{\mathcal{I}_{i+1}}$, which is, by the semantics of actions, equivalent to $x \in (\mathsf{occ}_{i+1}(A))^{\mathcal{I}_i}$. It is not difficult to show that this holds iff $t, \mathfrak{T} \models \mathsf{occ}_{i+1}(A)$. The case $\{\neg A^{(i)}, A^{(i+1)}\} \subseteq t_{\mathsf{Cl}}(x)$ is similar.

(ii) Let r be a role and let $(\exists r.C)^{(i)} \in t$. Thus $x \in (\exists r.C)^{\mathcal{I}_i}$ and there is a $y \in \Delta$, such that $(x,y) \in r^{\mathcal{I}_i}$ and $y \in C^{\mathcal{I}_i}$. We define $t' := t_{\mathsf{Cl}}(y)$ and $\rho := \{i \mid (x,y) \in r^{\mathcal{I}_i}\}$. It is not difficult to check that t' and ρ satisfy Conditions (R1) to (R5) from Definition 5.2.3.

This finishes the proof of Claim 3. Let $\mathfrak{T}_0 := \mathfrak{T}_S \cup \mathfrak{T}_{\mathsf{ano}}$ and let $\mathfrak{T}'$ be the set of types which "survive" type elimination when it is started with $\mathfrak{T}_0$. By Claim 3 and since a type t being good in a type set $\mathfrak{T}$ implies that t is good in any set $\mathfrak{T}' \supseteq \mathfrak{T}$, we have that $\mathfrak{T}_S \subseteq \mathfrak{T} \subseteq \mathfrak{T}'$. Moreover, since $\mathcal{I}_n \models \neg C_0(a_0)$, we have that $\neg C_0^{(n)} \in t_{\mathsf{Cl}}(a_0^{\mathcal{I}_n})$, and thus $\{\{a_0\}, \neg C_0^{(n)}\} \subseteq t_{\mathsf{Cl}}(a_0^{\mathcal{I}_n}) \in \mathfrak{T}'$.

"$\Leftarrow$". For this direction, we also show the contrapositive. Assume that $\mathcal{ALCIO}$-elim returns false for the input $\mathcal{A}_0, \mathcal{T}, \alpha_1, \ldots, \alpha_n, \varphi_0$. We show that φ_0 is not a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. To this end, we construct $\mathcal{I}_0, \ldots, \mathcal{I}_n$ such that $\mathcal{I}_0 \models \mathcal{A}_0$, $\mathcal{I}_i \Rightarrow^{\mathcal{T}}_{\alpha_{i+1}} \mathcal{I}_{i+1}$ for all $i \leq n$, and $\mathcal{I}_n \not\models C_0(a_0)$.

Since the algorithm returns false, there exists a proper core type set $\mathfrak{T}_S$ and a type set $\mathfrak{T}$ such that:

- $\mathfrak{T}_S \subseteq \mathfrak{T}$ and $\mathfrak{T} \setminus \mathfrak{T}_S \subseteq \mathfrak{T}_{\mathsf{ano}}$
- for every $t \in \mathfrak{T}$, t is good in $\mathfrak{T}$
- there is a $t_0 \in \mathfrak{T}$ with $\{\{a_0\}, \neg C_0^{(n)}\} \subseteq t_0$

The types from $\mathfrak{T}$ will be the elements of the domains of $\mathcal{I}_i$. Let $t \in \mathfrak{T}$. Since t is good in $\mathfrak{T}$, for every $D = (\exists r.C)^{(i)} \in t$ we can choose a type t' and a set of indices ρ which satisfy Conditions (R1) to (R5) of Definition 5.2.3. Then, we call the chosen t' a (ρ, r)-successor of t. For $t, t' \in \mathfrak{T}$ and r a role, set

$$\mathcal{R}(r,t,t') := \bigcup \{\rho \mid t' \text{ is a } (\rho,r)\text{-successor of } t\}$$

Since $\mathfrak{T}_S$ is a core type set and $\mathfrak{T} \setminus \mathfrak{T}_S \subseteq \mathfrak{T}_{\mathsf{ano}}$, for every $a \in \mathsf{Ind}$ there is exactly one type $t \in \mathfrak{T}$ such that $\{a\} \in t$. For every $a \in \mathsf{Ind}$, we denote this type with t_a. Now we can define $\mathcal{I}_0, \ldots, \mathcal{I}_n$ as follows: for all $i \leq n$,

$$\begin{array}{rcl} \Delta^{\mathcal{I}_i} & := & \mathfrak{T} \\ A^{\mathcal{I}_i} & := & \{t \in \mathfrak{T} \mid A^{(i)} \in t\} \\ r^{\mathcal{I}_i} & := & \{(t,t') \in \mathfrak{T} \times \mathfrak{T} \mid i \in \mathcal{R}(r,t,t') \cup \mathcal{R}(r^-,t',t)\} \\ a^{\mathcal{I}_i} & := & t_a \end{array}$$

Note that the definition of type ensures that interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ respect UNA on the individuals from Ind. Indeed, for all $a, b \in \mathsf{Ind}$, $b^{\mathcal{I}_i} = a^{\mathcal{I}_i}$ implies that $\{\{a\}, \{b\}\} \subseteq t_a (= t_b)$. The definition of type implies that $a = b$.

Claim 4: For all $t \in \mathfrak{T}$ and $C^{(i)} \in \mathsf{Cl}$, we have $t \in C^{\mathcal{I}_i}$ iff $C^{(i)} \in t$.

Proof. We prove the claim by induction on the structure of C.

- $C = A$ and $C = \{a\}$: trivial by definition of $\mathcal{I}_0, \ldots, \mathcal{I}_n$.
- $C = \neg D$ and $C = D \sqcap E$: easy by definition of type.

- $C = \exists s.D$, where s is a (possibly inverse) role:

  "only if". $t \in (\exists s.D)^{\mathcal{I}_i}$ implies that there exists a t' such that $(t,t') \in s^{\mathcal{I}_i}$ and $t' \in D^{\mathcal{I}_i}$. By induction, we have that $D^{(i)} \in t'$. Assume that $(\exists s.D)^{(i)} \notin t$. Then $(\neg\exists s.D)^{(i)} \in t$. By definition of $\mathcal{I}_i$, $(t,t') \in s^{\mathcal{I}_i}$ implies that one of the following cases applies:

  (i) $i \in \mathcal{R}(s,t,t')$. Then $(\neg\exists s.D)^{(i)} \in t$ and Condition (R2) of Definition 5.2.3, imply $\neg D^{(i)} \in t'$, which is a contradiction to t' being a type.

  (ii) $i \in \mathcal{R}(\mathsf{Inv}(s),t',t)$. Then $(\neg\exists s.D)^{(i)} \in t$ and Condition (R3) of Definition 5.2.3, imply $\neg D^{(i)} \in t'$, which is a contradiction to t' being a type.

  Thus $(\exists s.D)^{(i)} \in t$.

  "if". Let $(\exists s.D)^{(i)} \in t$. Since t is a good type in $\mathfrak{T}$, there exists a type $t' \in \mathfrak{T}$ and a set $\rho \ni i$, such that $\mathcal{R}(s,t,t') \supseteq \rho$ and $D^{(i)} \in t$. By definition of $s^{\mathcal{I}_i}$, we have that $(t,t') \in s^{\mathcal{I}_i}$. Moreover, since $D^{(i)} \in t'$, by induction we have that $t' \in D^{\mathcal{I}_i}$. Thus, it holds that $t \in (\exists s.D)^{\mathcal{I}_i}$.

Using Claim 4, the next claim is easily shown.

Claim 5. Let $t,t' \in \mathfrak{T}$. For all $0 \leq i < n$ the following holds:

(i) for all concept literals B from the input, $t, \mathfrak{T} \models \mathsf{occ}_{i+1}(B)$ implies $t \in (\mathsf{occ}_{i+1}(B))^{\mathcal{I}_i}$.

(ii) for all role literals s from the input, $t,t', \mathfrak{T} \models \mathsf{occ}_{i+1}(s)$ implies $(t,t') \in (\mathsf{occ}_{i+1}(s))^{\mathcal{I}_i}$.

We use Claims 4 and 5 to show that $\mathcal{I}_0 \models \mathcal{A}_0$, $\mathcal{I}_i \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_{i+1}$ for all $i \leq n$, and that $\mathcal{I}_0 \models \neg C_0(a_0)$.

- $\mathcal{I}_0 \models \mathcal{A}_0$: Let $C(a) \in \mathcal{A}_0$. Since $\mathfrak{T}_S$ is proper, $C^{(0)} \in t_a$ and, by Claim 4, we have that $a^{\mathcal{I}_0} = t_a \in C^{\mathcal{I}_0}$.

- $\mathcal{I}_i \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_{i+1}$ for all $i \leq n$:

  - $\mathcal{I}_i$ are models of $\mathcal{T}$: Let $C \sqsubseteq D \in \mathcal{T}$ and $t \in C^{\mathcal{I}_i}$. By Claim 4, we have $C^{(i)} \in t$, and since t is a type for Cl, $C^{(i)} \in t$ implies $D^{(i)} \in t$. Hence, $t \in D^{\mathcal{I}_i}$.
  - Let $C(a)/D(b) \in \mathsf{post}_i$ and let $\mathcal{I}_i \models C(a)$. This means that $t_a = a^{\mathcal{I}_i} \in C^{\mathcal{I}_i}$, and thus $\{\{a\}, C^{(i)}\} \subseteq t_a$. Since $\mathfrak{T}_S$ is proper, there exists a $t \in \mathfrak{T}_S$, such that $\{\{b\}, D^{(i+1)}\} \subseteq t$. Obviously, $t = t_b$, and thus $t_b = b^{\mathcal{I}_i} \in D^{\mathcal{I}_{i+1}}$, i.e. $\mathcal{I}_{i+1} \models D(b)$.
  - for each $A \in \mathsf{N_C}$ and $i < n$, we have: $A^{\mathcal{I}_i} \setminus A^{\mathcal{I}_{i+1}} \subseteq (\mathsf{occ}_{i+1}(A))^{\mathcal{I}_i}$ and $A^{\mathcal{I}_{i+1}} \setminus A^{\mathcal{I}_i} \subseteq (\mathsf{occ}_{i+1}(\neg A))^{\mathcal{I}_i}$. We show only the former since the latter can be shown in a similar way. Let $t \in A^{\mathcal{I}_i}$ and $t \notin A^{\mathcal{I}_{i+1}}$. By Claim 4, this implies $A^{(i)} \in t$ and $(\neg A)^{(i+1)} \in t$. Since t is good in $\mathfrak{T}$, we have that $t, \mathfrak{T} \models \mathsf{occ}_{i+1}(A)$, and by Claim 5 we obtain that $t \in (\mathsf{occ}_{i+1}(A))^{\mathcal{I}_i}$.
  - for each $r \in \mathsf{N_R}$ and $i < n$, we have: $r^{\mathcal{I}_i} \setminus r^{\mathcal{I}_{i+1}} \subseteq (\mathsf{occ}_{i+1}(r))^{\mathcal{I}_i}$ and $r^{\mathcal{I}_{i+1}} \setminus r^{\mathcal{I}_i} \subseteq (\mathsf{occ}_{i+1}(\neg r))^{\mathcal{I}_i}$. Again, we show only the former as the latter can be shown in a similar way. Let $(t,t') \in r^{\mathcal{I}_i}$ and $(t,t') \notin r^{\mathcal{I}_{i+1}}$. By the definition of $r^{\mathcal{I}_i}$, $i \in \mathcal{R}(r,t,t') \cup \mathcal{R}(r^-,t',t)$ and $i+1 \notin \mathcal{R}(r,t,t') \cup \mathcal{R}(r^-,t',t)$.

(i) Let $i \in \mathcal{R}(r,t,t')$. By definition of $\mathcal{R}$, t' is a (ρ,r)-successor of t for a $\rho \ni i$. Since $i+1 \notin \mathcal{R}(r,t,t')$, we obtain that $i+1 \notin \rho$. Since t is good in $\mathfrak{T}$, by Condition (R4) of Definition 5.2.3, we have that $t,t',\mathfrak{T} \models \mathsf{occ}_{i+1}(r)$. By Claim 5, we obtain that $(t,t') \in (\mathsf{occ}_{i+1}(r))^{\mathcal{I}_i}$.

(ii) Let $i \in \mathcal{R}(r^-,t',t)$. Since $i+1 \notin \mathcal{R}(r^-,t',t)$ and t' is good in $\mathfrak{T}$, similarly as in (i), by Condition (R4) of Definition 5.2.3, we have that $t',t,\mathfrak{T} \models \mathsf{occ}_{i+1}(r^-)$, which is equivalent to $t,t',\mathfrak{T} \models \mathsf{occ}_{i+1}(r)$. By Claim 5, we obtain that $(t,t') \in (\mathsf{occ}_{i+1}(r))^{\mathcal{I}_i}$.

- $\mathcal{I}_n \models \neg C_0(a_0)$: Since $(C_0)^{(n)} \in t_{a_0}$, by Claim 4, it holds that $t_{a_0} = a_0^{\mathcal{I}_n} \in (C_0)^{\mathcal{I}_n}$

❑

The algorithm runs in exponential time: first, we have already argued that there are only exponentially many core type sets. Second, the number of elimination rounds is bounded by the number of types, of which there are only exponentially many. And third, it is easily seen that it can be checked in exponential time whether a type is good in a given type set.

Moreover, for an $\mathcal{ALC}$-concept C and a general TBox $\mathcal{T}$, it is easy to see that the following holds: C is satisfiable w.r.t. $\mathcal{T}$ iff $\neg C(a)$ is *not* a consequence of applying the empty action $(\emptyset,\emptyset,\emptyset)$ in $\emptyset$ w.r.t. $\mathcal{T}$. Thus, since concept satisfiability w.r.t. general TBoxes is ExpTime-hard in $\mathcal{ALC}$ [BCM+03] and concept satisfiability can be reduced to (non-)projection, we obtain the following result.

Theorem 5.2.5. *Projection, executability and weak consistency in $\mathfrak{A}_2$ are* ExpTime*-complete for $\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCI}$, and $\mathcal{ALCIO}$.*

It is not too difficult to adapt the algorithm given in this section to the DL $\mathcal{ALCQO}$. Therefore, the reasoning problems from Theorem 5.2.5 are also ExpTime-complete for $\mathcal{ALCQ}$ and $\mathcal{ALCQO}$.

5.2.2 Projection beyond ExpTime

In the previous section, we have identified a number of DLs for which both reasoning about actions and standard DL reasoning are ExpTime-complete. Another candidate for a DL with such a behaviour is $\mathcal{ALCQI}$, in which satisfiability and subsumption are ExpTime-complete as well [Tob00]. However, since the action formalism $\mathfrak{A}_2$ is a generalization of the action formalism $\mathfrak{A}_1$, Theorem 3.2.6 implies the following.

Lemma 5.2.6. *Projection and executability (weak consistency) in $\mathfrak{A}_2$ are co-*NExpTime*-hard (*NExpTime*-hard) for $\mathcal{ALCQI}$ even if occlusions for roles are disallowed and only nominals are allowed in the occlusions of concept names.*

In the following, we show that projection for $\mathcal{ALCQI}$ is in fact co-NExpTime-complete, and that the same holds for the DL $\mathcal{ALCQIO}$. Note that, for the latter DL, also concept subsumption is co-NExpTime-complete. The proof proceeds by reducing projection in $\mathcal{ALCQIO}$ to ABox consequence in $\mathcal{ALCQIO}^{\neg,\cap,\cup}$, i.e., the extension of $\mathcal{ALCQIO}$ with the Boolean role constructors $\neg$, $\cap$, and $\cup$ with the following semantics:

$$\begin{aligned}(\neg r)^{\mathcal{I}} &:= (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus r^{\mathcal{I}} \\ (r \cap s)^{\mathcal{I}} &:= r^{\mathcal{I}} \cap s^{\mathcal{I}} \\ (r \cup s)^{\mathcal{I}} &:= r^{\mathcal{I}} \cup s^{\mathcal{I}}\end{aligned}$$

Let $\alpha_1, \ldots, \alpha_n$ be a composite action with $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ for $i = 1, \ldots, n$, and let $\mathcal{T}$ be a general TBox, $\mathcal{A}_0$ an ABox and φ_0 an assertion. We are interested in deciding whether φ_0 is a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. In what follows, we call $\alpha_1, \ldots, \alpha_n$, $\mathcal{T}$, $\mathcal{A}_0$ and φ_0 the *input*. W.l.o.g., we make the following assumptions:

- φ_0 is of the form $\varphi_0 = C_0(a_0)$, where C_0 is a (possibly complex) concept.

  As in the previous section, this assumption can be made because an assertion $r(a, b)$ can be replaced with $(\exists r.\{b\})(a)$, and $\neg r(a, b)$ with $(\neg\exists r.\{b\})(a)$.

- Each occlusion pattern occ_i contains exactly one occlusion pattern that is unconditional (i.e., indexed by taut) and formulated in $\mathcal{ALCQIO}^{\neg,\sqcap,\sqcup}$.

  An occlusion pattern $\{\mathsf{occ}_{\varphi_1}, \ldots, \mathsf{occ}_{\varphi_n}\}$ can be converted into an occlusion pattern $\{\mathsf{occ}_{\mathsf{taut}}\}$ formulated in $\mathcal{ALCQIO}^{\neg,\sqcap,\sqcup}$ as follows. First, we may assume w.l.o.g. that φ_i is of the form $C_i(a_i)$ for $1 \leq i \leq n$ (see previous point). For $1 \leq i \leq n$, let P_i denote the concept $\forall U.(\{a_i\} \to C_i)$, where U denotes the universal role, i.e. $r \sqcup \neg r$ for some $r \in \mathsf{N_R}$. Then, define for each concept literal A

  $$\mathsf{occ}_{\mathsf{taut}}(A) := \bigsqcup_{1 \leq i \leq n} (P_i \sqcap \mathsf{occ}_{\varphi_i}(A))$$

  Likewise, for each role literal r, define

  $$\mathsf{occ}_{\mathsf{taut}}(r) := \{(P_i \sqcap C, P_i \sqcap D) \mid (C, D) \in \mathsf{occ}_{\varphi_i}\}.$$

  Having the occlusion pattern formulated in $\mathcal{ALCQIO}^{\neg,\sqcap,\sqcup}$ is unproblematic since our reduction is to $\mathcal{ALCQIO}^{\neg,\sqcap,\sqcup}$ anyway. In the following, we slightly abuse notation and confuse the singleton set occ_i with the (unconditional) occlusion mapping contained in it.

The idea of the reduction is to define an ABox $\mathcal{A}_{\mathsf{red}}$ and a TBox $\mathcal{T}_{\mathsf{red}}$ such that a single model of them encodes a sequence of interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ such that $\mathcal{I}_0 \models \mathcal{A}_0, \mathcal{T}$ and $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_i$ for $i = 1, \ldots, n$. As in the previous section, we use Sub to denote the set of subconcepts of the concepts which occur in the input and introduce concept names $A^{(i)}$ and role names $r^{(i)}$ for every concept name A and every role name r used in the input, for all $i \leq n$. For a complex concept $C \in \mathsf{Sub}$, we use $C^{(i)}$, for $i \leq n$, to denote the concept obtained by replacing all concept names A and role names r occurring in C by $A^{(i)}$ and $r^{(i)}$ respectively.

We start by assembling the reduction ABox $\mathcal{A}_{\mathsf{red}}$. First, define a "copy" $\mathcal{A}_{\mathsf{ini}}$ of the input ABox $\mathcal{A}_0$ as:

$$\begin{aligned}\mathcal{A}_{\mathsf{ini}} := \quad & \{C^{(0)}(a) \mid C(a) \in \mathcal{A}_0\} \cup \\ & \{r^{(0)}(a, b) \mid r(a, b) \in \mathcal{A}_0\} \cup \{\neg r^{(0)}(a, b) \mid \neg r(a, b) \in \mathcal{A}_0\}\end{aligned}$$

Then, introduce abbreviations, for $i \leq n$:

$$\begin{aligned}\mathsf{p}_i(C(a)) &:= \forall U.(\{a\} \to C^{(i)}), \\ \mathsf{p}_i(r(a, b)) &:= \forall U.(\{a\} \to \exists r^{(i)}.\{b\}), \\ \mathsf{p}_i(\neg r(a, b)) &:= \forall U.(\{a\} \to .\forall r^{(i)}.\neg\{b\}),\end{aligned}$$

Now we can define the components of $\mathcal{A}_{\mathsf{red}}$ that take care of post-condition satisfaction. For $1 \leq i \leq n$, we define:

$$\mathcal{A}^{(i)}_{\mathsf{post}} := \{\big(\mathsf{p}_{i-1}(\varphi) \rightarrow \mathsf{p}_i(\psi)\big)(a_0) \mid \varphi/\psi \in \mathsf{post}_i\}$$

We assemble $\mathcal{A}_{\mathsf{red}}$ as

$$\mathcal{A}_{\mathsf{red}} := \mathcal{A}_{\mathsf{ini}} \cup \bigcup_{1 \leq i \leq n} \mathcal{A}^{(i)}_{\mathsf{post}}.$$

Next, we define the components of the TBox $\mathcal{T}_{\mathsf{red}}$. Since all interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ have to be models of the input TBox $\mathcal{T}$, we define for each $i \leq n$, a copy $\mathcal{T}^{(i)}$ of $\mathcal{T}$ in the obvious way:

$$\mathcal{T}^{(i)} = \{C^{(i)} \sqsubseteq D^{(i)} \mid C \sqsubseteq D \in \mathcal{T}\}.$$

To deal with occlusions, we introduce auxiliary role names $r^{(i)}_{\mathsf{Dom}(C)}$ and $r^{(i)}_{\mathsf{Ran}(D)}$ for $0 \leq i < n$ and all concepts C, D such that $(C, D) \in \mathsf{occ}_i(s)$ for some role literal s. The following TBox $\mathcal{T}^{(i)}_{\mathsf{aux}}$ ensures that $r^{(i)}_{\mathsf{Dom}(C)}$ and $r^{(i)}_{\mathsf{Ran}(D)}$ are interpreted as $C^{(i)} \times \top$ and $\top \times D^{(i)}$, respectively. It contains the following axioms, for all concepts C, D as above:

$$\begin{array}{rcl@{\qquad}rcl} C^{(i)} & \sqsubseteq & \forall \neg r^{(i)}_{\mathsf{Dom}(C)}.\bot & \top & \sqsubseteq & \forall r^{(i)}_{\mathsf{Ran}(D)}.D^{(i)} \\ \neg C^{(i)} & \sqsubseteq & \forall r^{(i)}_{\mathsf{Dom}(C)}.\bot & \top & \sqsubseteq & \forall \neg r^{(i)}_{\mathsf{Ran}(D)}.\neg D^{(i)} \end{array}$$

The following TBox $\mathcal{T}^{(i)}_{\mathsf{fix}}$ ensures that concept and role names do not change unless this is allowed by the occlusion pattern:

- for every concept name A in the input,

$$\begin{array}{rcl} A^{(i)} \sqcap \neg A^{(i+1)} & \sqsubseteq & (\mathsf{occ}_{i+1}(A))^{(i)} \\ \neg A^{(i)} \sqcap A^{(i+1)} & \sqsubseteq & (\mathsf{occ}_{i+1}(\neg A))^{(i)} \end{array}$$

- for every role name r in the input,

$$\begin{array}{rcl} \top & \sqsubseteq & \forall \neg \Big(\bigcup_{(C,D) \in \mathsf{occ}_{i+1}(r)} (r^{(i)}_{\mathsf{Dom}(C)} \cap r^{(i)}_{\mathsf{Ran}(D)}) \Big) \cap (r^{(i)} \cap \neg r^{(i+1)}).\bot \\ \top & \sqsubseteq & \forall \neg \Big(\bigcup_{(C,D) \in \mathsf{occ}_{i+1}(\neg r)} (r^{(i)}_{\mathsf{Dom}(C)} \cap r^{(i)}_{\mathsf{Ran}(D)}) \Big) \cap (\neg r^{(i)} \cap r^{(i+1)}).\bot \end{array}$$

Finally, we can construct $\mathcal{T}_{\mathsf{red}}$ as

$$\mathcal{T}_{\mathsf{red}} := \bigcup_{0 \leq i \leq n} \mathcal{T}^{(i)} \cup \bigcup_{0 \leq i < n} \mathcal{T}^{(i)}_{\mathsf{aux}} \cup \bigcup_{0 \leq i < n} \mathcal{T}^{(i)}_{\mathsf{fix}}.$$

Then the following lemma holds:

Lemma 5.2.7. *$C_0(a_0)$ is a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$ iff $C^{(n)}_0(a_0)$ is a consequence of $\mathcal{A}_{\mathsf{red}}$ w.r.t. $\mathcal{T}_{\mathsf{red}}$ (under UNA).*

Proof. "⇒": We prove this direction by contraposition. Assume that $C_0^{(n)}(a_0)$ is not a consequence of $\mathcal{A}_{\mathsf{red}}$ w.r.t. $\mathcal{T}_{\mathsf{red}}$. Thus, there exists a $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$, $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$, and $\mathcal{J} \models \neg C_0^{(n)}(a_0)$. In order to prove that $C_0(a_0)$ is not a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$ we show that there are $\mathcal{I}_0, \ldots, \mathcal{I}_n$ such that $\mathcal{I}_{i-1} \Rightarrow_{\alpha_i}^{\mathcal{T}} \mathcal{I}_i$ for all $1 \leq i \leq n$, $\mathcal{I}_0 \models \mathcal{A}_0$, and $\mathcal{I}_n \not\models C_0(a_0)$.

We construct $\mathcal{I}_i$ for $i \leq n$ as following:

- $\Delta^{\mathcal{I}_i} := \Delta^{\mathcal{J}}$;
- $A^{\mathcal{I}_i} := (A^{(i)})^{\mathcal{J}}$ for every concept name A;
- $r^{\mathcal{I}_i} := (r^{(i)})^{\mathcal{J}}$ for every role name r;
- $a^{\mathcal{I}_i} := a^{\mathcal{J}}$ for every individual name a.

By definition of $C^{(i)}$, it is obvious that for all $x \in \Delta^{\mathcal{J}}$, $C \in \mathsf{Sub}$ and $i \leq n$:

$$x \in C^{\mathcal{I}_i} \text{ iff } x \in (C^{(i)})^{\mathcal{J}} \qquad (*)$$

We have that $\mathcal{I}_{i-1} \Rightarrow_{\alpha_i}^{\mathcal{T}} \mathcal{I}_i$ for all $1 \leq i \leq n$ since

- $\mathcal{I}_i \models \mathcal{T}$: for all $C \sqsubseteq D \in \mathcal{T}$, we have $C^{(i)} \sqsubseteq D^{(i)} \in \mathcal{T}_{\mathsf{red}}$. $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$ implies $\mathcal{J} \models C^{(i)} \sqsubseteq D^{(i)}$. For all $x \in \Delta^{\mathcal{I}_i} = \Delta^{\mathcal{J}}$, we have $x \in (C^{(i)})^{\mathcal{J}}$ implies $x \in (D^{(i)})^{\mathcal{J}}$. By $(*)$, this yields that $x \in C^{\mathcal{I}_i}$ implies $x \in D^{\mathcal{I}_i}$.
- $\mathcal{I}_{i-1}, \mathcal{I}_i \models \mathsf{post}_i$: It follows from the definition of p_i that for all ABox assertions φ and for all $i \leq n$, we have $(\mathsf{p}_i(\varphi))^{\mathcal{J}} = \Delta^{\mathcal{J}}$ if $\mathcal{I}_i \models \varphi$ and $(\mathsf{p}_i(\varphi))^{\mathcal{J}} = \emptyset$ otherwise. For all $\varphi/\psi \in \mathsf{post}_i$, we have $(\mathsf{p}_{i-1}(\varphi) \rightarrow \mathsf{p}_i(\psi))(a_0) \in \mathcal{A}_{\mathsf{red}}$. Assume $\mathcal{I}_{i-1} \models \varphi$. Then $(\mathsf{p}_{i-1}(\varphi))^{\mathcal{J}} = \Delta^{\mathcal{I}_{i-1}}$ by $(*)$. Thus, $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$ yields $(\mathsf{p}_{i-1}(\psi))^{\mathcal{J}} = \Delta^{\mathcal{J}}$, which implies $\mathcal{I}_i \models \psi$ by $(*)$.
- $A^{\mathcal{I}_{i-1}} \setminus A^{\mathcal{I}_i} \subseteq (\mathsf{occ}_i(A))^{\mathcal{I}_{i-1}}$ and $A^{\mathcal{I}_i} \setminus A^{\mathcal{I}_{i-1}} \subseteq (\mathsf{occ}_i(\neg A))^{\mathcal{I}_{i-1}}$ follow from $\mathcal{J} \models \mathcal{T}_{\mathsf{fix}}^{(i)}$.
- $r^{\mathcal{I}_{i-1}} \setminus r^{\mathcal{I}_i} \subseteq (\mathsf{occ}_i(r))^{\mathcal{I}_{i-1}}$: Let $(x, y) \in r^{\mathcal{I}_{i-1}} \setminus r^{\mathcal{I}_i}$. By the construction of $\mathcal{I}_{i-1}$ and $\mathcal{I}_i$, we have $(x, y) \in (r^{(i-1)} \cap \neg r^{(i)})^{\mathcal{J}}$. Then $\mathcal{J} \models \mathcal{T}_{\mathsf{fix}}^{(i)}$ implies

$$(x, y) \in \Big(\bigcup_{(C,D) \in \mathsf{occ}_i(r)} (r^{(i-1)}_{\mathsf{Dom}(C)} \cap r^{(i-1)}_{\mathsf{Ran}(D)}) \Big)^{\mathcal{J}}.$$

  Hence, there exists a pair $(C, D) \in \mathsf{occ}_i(r)$ such that $(x, y) \in (r^{(i-1)}_{\mathsf{Dom}(C)} \cap r^{(i-1)}_{\mathsf{Ran}(D)})^{\mathcal{J}}$. Moreover, $\mathcal{J} \models \mathcal{T}_{\mathsf{aux}}^{(i)}$ implies $x \in (C^{(i-1)})^{\mathcal{J}}$ and $y \in (D^{(i-1)})^{\mathcal{J}}$.

  Thus, by $(*)$ we have $x \in C^{\mathcal{I}_{i-1}}$ and $y \in D^{\mathcal{I}_{i-1}}$ which implies $(x, y) \in (\mathsf{occ}_i(r))^{\mathcal{I}_{i-1}}$. Analogously, it can be shown that $r^{\mathcal{I}_i} \setminus r^{\mathcal{I}_{i-1}} \subseteq (\mathsf{occ}_i(\neg r))^{\mathcal{I}_{i-1}}$ holds.

$\mathcal{I}_0 \models \mathcal{A}_0$: for all concept assertions $C(a) \in \mathcal{A}$, we have $C^{(0)}(a) \in \mathcal{A}_{\mathsf{red}}$. $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$ implies $a^{\mathcal{J}} \in (C^{(0)})^{\mathcal{J}}$. Then, by $(*)$ we know $a^{\mathcal{I}_0} \in C^{\mathcal{I}_0}$. We can prove the same result for all role assertions in $\mathcal{A}$ from the definition of $r^{\mathcal{I}_0}$ and $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$.

$\mathcal{I}_n \not\models C_0(a_0)$: $\mathcal{J} \models \neg C_0^{(n)}(a_0)$ implies $a_0^{\mathcal{J}} \in (\neg C_0^{(n)})^{\mathcal{J}}$. Thus, by $(*)$ we know $a_0^{\mathcal{J}} = a_0^{\mathcal{I}_n} \in (\neg C_0)^{\mathcal{I}_n}$.

"$\Leftarrow$": This direction can also be proved by contraposition. Assume that $C_0(a_0)$ is not a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. Thus, there are $\mathcal{I}_0, \ldots, \mathcal{I}_n$ such that $\mathcal{I}_{i-1} \Rightarrow_{\alpha_i}^{\mathcal{T}} \mathcal{I}_i$ for all $1 \leq i \leq n$, $\mathcal{I}_0 \models \mathcal{A}_0$, and $\mathcal{I}_n \not\models C_0(a_0)$. We define an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ as follows:

- $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}_0} (= \Delta^{\mathcal{I}_1} = \cdots = \Delta^{\mathcal{I}_n})$;
- $(A^{(i)})^{\mathcal{J}} := A^{\mathcal{I}_i}$ for all concept names A and for all $i \leq n$;
- $(r^{(i)})^{\mathcal{J}} := r^{\mathcal{I}_i}$ for all role names r and for all $i \leq n$;
- $a^{\mathcal{J}} := a^{\mathcal{I}_0} (= a^{\mathcal{I}_1} = \cdots = a^{\mathcal{I}_n})$ for all individual names a;
- $(r^{(i)}_{\mathsf{Dom}(C)})^{\mathcal{J}} := \{C^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}\}$ and $(r^{(i)}_{\mathsf{Ran}(D)})^{\mathcal{J}} := \{\Delta^{\mathcal{I}_i} \times D^{\mathcal{I}_i}\}$ for all $i < n$.

By definition of $C^{(i)}$, it obvious that for all $x \in \Delta^{\mathcal{J}}$, $C \in \mathsf{Sub}$ and $i \leq n$:

$$x \in C^{\mathcal{I}_i} \text{ iff } x \in (C^{(i)})^{\mathcal{J}}$$

Using this observation and the semantics of actions, it is not difficult to show that indeed $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$, $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$, and $\mathcal{J} \models \neg C_0^{(n)}(a_0)$. Thus, $C_0^{(n)}(a_0)$ is not a consequence of $\mathcal{A}_{\mathsf{red}}$ w.r.t. $\mathcal{T}_{\mathsf{red}}$. ❑

Since $\mathcal{ALCQIO}^{\neg,\sqcap,\sqcup}$ is a fragment of $\mathcal{C}^2$ (the 2-variable fragment of first-order logic with counting), ABox inconsistency in $\mathcal{ALCQIO}^{\neg,\sqcap,\sqcup}$ is in co-NEXPTIME even if numbers are coded in binary [PH05]. Since $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ are polynomial in the size of the input, Lemma 5.2.7 gives us a co-NEXPTIME upper bound for projection in $\mathcal{ALCQIO}$ and $\mathcal{ALCQI}$. Together with Lemma 5.2.6 we obtain the following:

Theorem 5.2.8. *Projection and executability in $\mathfrak{A}_2$ are co-*NEXPTIME*-complete, while weak consistency is* NEXPTIME*-complete for $\mathcal{ALCQIO}$ and $\mathcal{ALCQI}$.*

5.3 Undecidability of Strong Consistency

Unlike projection, executability, or weak consistency, strong consistency is undecidable. We show that it is undecidable already in $\mathcal{ALC}$. The proof consists of a reduction of the undecidable *semantic consequence problem* from modal logic. Before formulating the DL version of this problem, we need some preliminaries. We use $\mathcal{ALC}$ concepts with only one fixed role name r, which we call $\mathcal{ALC}_r$-concepts. Accordingly, we also assume that interpretations interpret only concept names and the role name r. An *r-frame* is a structure $\mathcal{F} = (\Delta^{\mathcal{F}}, r^{\mathcal{F}})$ where $\Delta^{\mathcal{F}}$ is a non-empty set and $r^{\mathcal{F}} \subseteq \Delta^{\mathcal{F}} \times \Delta^{\mathcal{F}}$. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is *based* on an r-frame $\mathcal{F}$ iff $\Delta^{\mathcal{I}} = \Delta^{\mathcal{F}}$ and $r^{\mathcal{I}} = r^{\mathcal{F}}$. We say that a concept C is *valid* on $\mathcal{F}$ (written $\mathcal{F} \models C$) iff $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ for every interpretation $\mathcal{I}$ based on $\mathcal{F}$.

Definition 5.3.1 (Semantic consequence problem). Let D and E be $\mathcal{ALC}_r$-concepts. We say that E is a *semantic consequence* of D iff for every r-frame $\mathcal{F} = (\Delta^{\mathcal{F}}, r^{\mathcal{F}})$ such that $\mathcal{F} \models D$, it holds that $\mathcal{F} \models E$. △

In [Tho75], it is proved that for $\mathcal{ALC}_r$-concepts D and E, the problem "Is E a semantic consequence of D?" is undecidable. We now show that the semantic consequence problem can be reduced to strong consistency. For $\mathcal{ALC}_r$-concepts D and E, we define the action $\alpha_D = (\mathsf{pre}, \{\mathsf{occ}_{\mathsf{taut}}\}, \mathsf{post})$ where $\mathsf{pre} := \{\neg E(a)\}$, $\mathsf{post} := \{\top(a)/(\exists u.\neg D)(a)\}$ (u a role name), and $\mathsf{occ}_{\mathsf{taut}}$ maps r and $\neg r$ to $\{(\bot, \bot)\}$, all other role literals to $\{(\top, \top)\}$, and all concept literals to $\top$. Then the following holds.

Lemma 5.3.2. *The action α_D is strongly consistent with the empty TBox iff E is a semantic consequence of D.*

Proof. "⇒" We show the contraposition. Assume that E is not a semantic consequence of D. Then there exists an r-frame $\mathcal{F} = (\Delta^{\mathcal{F}}, r^{\mathcal{F}})$ such that $\mathcal{F} \models D$ and there is an interpretation $\mathcal{I}$ based on $\mathcal{F}$ such that $E^{\mathcal{I}} \neq \Delta^{\mathcal{I}}$. We take $\mathcal{I}$ based on $\mathcal{F}$ such that $a^{\mathcal{I}} \notin E^{\mathcal{I}}$, thus $\mathcal{I} \models \mathsf{pre}$. But every $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow^{\emptyset}_{\alpha_D} \mathcal{I}'$ must be based on $\mathcal{F}$ (since $r^{\mathcal{I}'} = r^{\mathcal{I}} = r^{\mathcal{F}}$) and must satisfy $D^{\mathcal{I}'} \neq \Delta^{\mathcal{I}'}$ (by the post-condition of α). Since $\mathcal{F} \models D$, there is no such $\mathcal{I}'$. Thus, α_D is not strongly consistent with the empty TBox.

"⇐" Assume that E is a semantic consequence of D. Let $\mathcal{I} \models \mathsf{pre}$. By definition of pre, we have that $a^{\mathcal{I}} \notin E^{\mathcal{I}}$, and thus $\mathcal{I}$ is not based on an r-frame $\mathcal{F} = (\Delta^{\mathcal{F}}, r^{\mathcal{F}})$ validating E. Since E is a semantic consequence of D, $\mathcal{F}$ is not validating D either, and there is an interpretation $\mathcal{I}'$ based on $\mathcal{F}$ such that $D^{\mathcal{I}'} \neq \Delta^{\mathcal{I}'}$. Take $y \in \Delta^{\mathcal{I}'}$ such that $y \notin D^{\mathcal{I}'}$. Since D is an $\mathcal{ALC}_r$-concept, we may assume that $u^{\mathcal{I}'} = \{(a^{\mathcal{I}'}, y)\}$. Obviously, we have that $\mathcal{I} \Rightarrow^{\emptyset}_{\alpha_D} \mathcal{I}'$, and, consequently, α_D is strongly consistent with the empty TBox. ❏

As an immediate consequence, we obtain the following theorem.

Theorem 5.3.3. *Strong consistency of $\mathcal{ALC}$-actions in $\mathfrak{A}_2$ is undecidable, even with the empty TBox.*

5.4 Practicability

Practical Projection Algorithms

Together with a reasoner that is capable of deciding ABox consistency in the description logic $\mathcal{ALCQIO}^{\neg,\cap,\cup}$, the reduction developed in Section 5.2.2 can be used for practical reasoning with $\mathcal{ALCQIO}$ actions. Unfortunately, to the best of our knowledge there exists no available DL reasoner that supports the Boolean operators on roles.

Here we identify a (natural) restricted version of the action formalism $\mathfrak{A}_2$ in which the reduction of projection to ABox consistency does not need the Boolean role constructors an thus projection can be reduced to standard reasoning problems in DLs that are implemented in DL reasoners such as RacerPro and FaCT++.

We call an $\mathfrak{A}_2$-action $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ *restricted* iff:

(i) for every role name r, and $\mathsf{occ}_\varphi \in \mathsf{occ}$ such that $\varphi \neq \mathsf{taut}$, $\mathsf{occ}_\varphi(r)$ and $\mathsf{occ}_\varphi(\neg r)$ are subsets of $\{(\{a\}, \{b\}) \mid a, b \in \mathsf{N_I}\}$;

(ii) if $\mathsf{occ}_{\mathsf{taut}} \in \mathsf{occ}$, then $\mathsf{occ}_{\mathsf{taut}}(r) = \mathsf{occ}_{\mathsf{taut}}(\neg r)$ for all role names r; and

(iii) for every role name r, $\mathsf{occ}_{\mathsf{taut}}(r)$ is a subset of

$$\{(\{a\}, \{b\}), (\{a\}, \top) \mid a, b \in \mathsf{N_I}\} \cup \{(\top, \top), (\bot, \bot)\}$$

For this restricted fragment, projection in $\mathcal{L}$ can be reduced to ABox (in)consistency in $\mathcal{LO}$, the extension of $\mathcal{L}$ with nominals. We restrain ourselves from presenting the reduction in detail, as it is a combination of (adapted parts of) the reductions from Section 3.2 and Section 5.2.2. Since the minimization of changes for concept names can be achieved with GCIs without introducing Boolean role constructors, as shown in Section 5.2.2, we just show that the technique from Section 3.2 can be used in order to ensure the minimization of changes in role interpretations in the non-occluded part of the domain.

Assume that we want to decide projection with respect to a composite action $\alpha_1, \ldots, \alpha_n$ with all $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ restricted. As in previous reductions, we use a role name $r^{(i)}$ for every role name r used in the input and every $i \leq n$. Moreover, we make a distinction between *named* (i.e. d such that $d = a^{\mathcal{I}}$ for some individual a used in the input) and *unnamed* domain elements. Intuitively, the extension of r in the i-th interpretation can be assembled from the extensions of $r^{(0)}, \ldots, r^{(n)}$ as follows:

- regarding pairs (x, y) with both x, y named, we consider $r^{(i)}$;

  Occlusions of role names on named elements are formalized by the reduction ABox $\mathcal{A}_{\mathsf{red}}$ by means of assertions similar to those in $\mathcal{A}^{(i)}_{\mathsf{min}}$ in the reduction presented in Section 3.2.

- regarding pairs (x, y) with one of x, y not named, we consider $r^{(j)}$, where $j \in \{1, \ldots, i\}$ is maximal such that $\mathsf{occ}_{j\,\mathsf{taut}}(r)$ contains the global occlusion $(\top, \top)$ if x is unnamed; and where where $j \in \{1, \ldots, i\}$ is maximal such that $\mathsf{occ}_{j\,\mathsf{taut}}(r)$ contains either $(\{a\}, \top)$ or $(\top, \top)$ if $x = a^{\mathcal{I}_i}$ (and $j = 0$ if there is no such j).

  Thus, in order to check the membership in $r^{\mathcal{I}_i}$ of pairs (x, y) with x or y unnamed, we "go back" to the last interpretation before which (x, y) was occluded. This can easily be ensured by acyclic concept definitions similar to those in the TBox $\mathcal{T}_{\mathsf{sub}}$ from Section 3.2. The concept name $T^{(i)}_{(\geqslant m\ r\ C)}$ stands for $(\geqslant m\ r\ C)$ in the i-th interpretation:

$$T^{(i)}_{(\geqslant m\ r\ C)} \doteq \bigsqcup_{a \in \mathsf{Ind}} \Big(\{a\} \sqcap \bigsqcup_{0 \leq j \leq m} \big((\geqslant j\ r^{(i)}\ (N \sqcap T^{(i)}_C)) \\ \sqcap (\geqslant (m-j)\ r^{(\mathsf{ol}^i_{a,r})}\ (\neg N \sqcap T^{(i)}_C)) \big) \Big) \\ \sqcup \big(\neg N \sqcap (\geqslant m\ r^{(\mathsf{ol}^i_r)}\ T^{(i)}_C) \big)$$

  where

$$\mathsf{lo}^i_{a,r} := \max\{\ \ell \mid 1 \leq \ell \leq i \ \wedge\ \{(\{a\}, \top), (\top, \top)\} \cap \mathsf{occ}_{\ell\,\mathsf{taut}}(r) \neq \emptyset\ \}$$
$$\mathsf{lo}^i_r := \max\{\ \ell \mid 1 \leq \ell \leq i \ \wedge\ (\top, \top) \in \mathsf{occ}_{\ell\,\mathsf{taut}}(r)\ \}$$

The reduction works for $\mathcal{ALC}$ extended with any combination of inverses, nominals, and number restrictions.

Strong Consistency

Our only negative result concerns the undecidability of strong consistency. To discuss the impact of this result, let us briefly review the relevance of strong consistency for the action designer and for the user of the action (the person who applies the action).

For the action *designer*, an algorithm for checking strong consistency would be useful for fine-tuning the ramifications of his action. However, it is worth noting that deciding strong consistency could not replace manual inspection of the ramifications. For example, occluding all concept names with $\top$ and all role names with $\{(\top, \top)\}$ usually ensures strong consistency but does not lead to an intuitive behaviour of the action. With weak consistency, we offer at least some automatic support to the action designer for detecting ramification problems.

For the *user* of the action, strong consistency is required to ensure that the execution of an action whose preconditions are satisfied will not fail. If the action is such that failure cannot be tolerated (because executing the action is expensive, dangerous, etc), strong consistency is thus indispensable and should already be guaranteed by the action designer. Also when working with composite actions, strong consistency has to be required: if an action execution fails after previous actions in the sequence have been successfully executed, then we have already changed the state of affairs and it may not be possible to revert these changes to use a different composite action for reaching the desired goal. However, in the case of atomic actions it is conceivable that an execution failure does not have any negative effects. If this is the case, the action user only needs to check that the action is executable, and strong consistency is not strictly required.

Chapter 6

Planning

In this chapter, we complete our work on description logic actions by investigating planning in $\mathfrak{A}_1$. For possible applications of DL actions, planning is a very important reasoning task. For example, in the context of semantic web services, planning can support web service discovery, which is needed for an automatic service execution.

We assume that in planning tasks, available operators are parameterized $\mathfrak{A}_1$-actions, i.e., $\mathfrak{A}_1$-actions where variables may occur in place of individuals. Moreover, we assume that these operators can be instantiated with individuals from a finite set of individuals Ind. Intuitively, the *plan existence problem* is: given an acyclic TBox $\mathcal{T}$ which defines abbreviations for complex concepts, ABoxes $\mathcal{A}$ and Γ giving incomplete descriptions of the initial and the goal state, and a set of operators Op, is there a plan (sequence of actions obtained by instantiating operators from Op) which "transforms" the state described by $\mathcal{A}$ into a state where Γ is satisfied? For example, given an incomplete knowledge about Dirk in the initial state, and a set of operators for obtaining a lease, a letter from the employer and a bank account, one might want to know which actions, and in which order, have to be executed in order to achieve that Dirk obtains a bank account with a credit card. We require that plans are *uniform*, i.e., that the same plan achieves the goal, independently of the model of the ABox describing the initial world state. Hence one may say that we investigate conformant DL planning.

It is known that, already in the propositional case, planning is a hard problem. For example, the plan existence problem for propositional STRIPS-style actions with complete state descriptions is PSpace-complete [Byl94], while it is ExpSpace-complete for conformant planning (incomplete state descriptions) where actions have conditional post-conditions [HJ99, Rin04].

In the largest part of this chapter we investigate the computational complexity of the plan existence problem for the propositionally closed fragments of $\mathcal{ALCQIO}$. If we allow only for actions with unconditional post-conditions, we show that, in these logics, the plan existence problem is decidable and of the same computational complexity as projection, ranging from PSpace-complete to co-NExpTime-complete. This is done by using a compact representation of possible states obtained by action application: every state of the search space can implicitly be described by means of the initial ABox $\mathcal{A}$ and the set of accumulated action post-conditions. If conditional post-conditions are allowed, such a compact representataion of the states of the search space is no longer posssible, since different models of the initial ABox $\mathcal{A}$ may cause different post-conditions to be triggered. Using a combinatorial method we show that the plan existence problem in this case is in 2-ExpSpace. Since the only known lower complexity bound is inherited from propositional logic (ExpSpace-hard), we leave the

exact computational complexity of conditional PLANEX as an open problem.

Finally, we investigate the complexity of planning in the lightweight description logic $\mathcal{EL}$. We show that hardness results for propositional planning carry over to planning in $\mathcal{EL}$, by adapting reductions from [Byl94] and [Rin04].

The rest of this chapter is organized as follows. In Section 6.1 we introduce the notion of a planning task and define the plan existence problem. In Section 6.2 we present the complexity results for planning if only unconditional post-conditions are allowed. Section 6.3 is dedicated to planning where we admit conditional post-conditions. Finally, in Section 6.4 we present complexity results for planning in $\mathcal{EL}$.

6.1 Planning Problem

In this section, we formally introduce the notion of the planing task based on the action formalism $\mathfrak{A}_1$, as well as the plan existence problem. Since so far we focused on the reasoning problems projection and executability, it sufficed to deal with ground $\mathfrak{A}_1$-actions, as introduced in Definition 3.1.1. In order to address planning, we need to introduce *parameterized actions*, called *operators*:

Definition 6.1.1 (Operator). Let $\mathsf{N_X}$ be a countably infinite set of *variables*, disjoint with the set of individuals $\mathsf{N_I}$. Let $\mathcal{T}$ be an acyclic TBox. An *operator for* $\mathcal{T}$ is a parameterized atomic $\mathfrak{A}_1$-action for $\mathcal{T}$, i.e., an action in whose definition variables from $\mathsf{N_X}$ may occur in place of individual names. △

If o is an operator (for a TBox $\mathcal{T}$), we use $\mathsf{var}(o)$ to denote the set of variables in o. A substitution v for o is a mapping $v : \mathsf{var}(o) \to \mathsf{N_I}$. The action α that is obtained by applying a substitution v to o is denoted with $\alpha := o[v]$. We assume that operators can be instantiated with individuals from a finite set $\mathsf{Ind} \subset \mathsf{N_I}$. Moreover, we assume that $\mathcal{T}$, $\mathcal{A}$ and Γ contain only individuals from Ind (we say that they are *based on* Ind). For an operator o, we set $o[\mathsf{Ind}] := \{o[v] \mid v : \mathsf{var}(o) \to \mathsf{Ind}\}$. Moreover, for a set of operators Op, we set $\mathsf{Op}[\mathsf{Ind}] := \{o[\mathsf{Ind}] \mid o \in \mathsf{Op}\}$, i.e. $\mathsf{Op}[\mathsf{Ind}]$ is the set of all actions obtained by instantiating operators from Op with individuals from Ind. The following definition formally introduces the notion of a planing task: [1]

Definition 6.1.2 (Planning task). A planning task is a tuple $\Pi = (\mathsf{Ind}, \mathcal{T}, \mathsf{Op}, \mathcal{A}, \Gamma)$, where

- Ind is a finite set of individual names;
- $\mathcal{T}$ is an acyclic TBox based on Ind;
- Op is a finite set of operators for $\mathcal{T}$;
- $\mathcal{A}$ (initial state) is an ABox based on Ind;
- Γ (goal) is an ABox based on Ind.

A *plan* in Π is a composite action $\alpha = \alpha_1, \ldots, \alpha_k$, such that $\alpha_i \in \mathsf{Op}[\mathsf{Ind}]$ and α_i is consistent with $\mathcal{T}$ [2], $i = 1..k$. A plan $\alpha = \alpha_1, \ldots, \alpha_k$ in Π is a *solution* to the planning task Π iff:

[1] The notion of a planing task is defined along the lines with standard STRIPS definitions. One difference is that our planning task has an additional component – an acyclic TBox.

[2] Recall that α is consistent with $\mathcal{T}$ iff for all models $\mathcal{I}$ of $\mathcal{T}$, there exists $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow_\alpha^\mathcal{T} \mathcal{I}'$; c.f. Section 3.1.2.

1. α is executable in $\mathcal{A}$ w.r.t. $\mathcal{T}$; and
2. for all interpretations $\mathcal{I}$ and $\mathcal{I}'$ such that $\mathcal{I} \models \mathcal{A}, \mathcal{T}$ and $\mathcal{I} \Rightarrow_\alpha^\mathcal{T} \mathcal{I}'$, it holds that $\mathcal{I}' \models \Gamma$.

△

A planning task $\Pi = (\mathsf{Ind}, \mathcal{T}, \mathsf{Op}, \mathcal{A}, \Gamma)$ is called *unconditional* if all operators in Op have only unconditional post-conditions, and *conditional*, otherwise.

Example 6.1.3. *We illustrate the previous definition by the following example describing a (simplified) process of opening a bank account in the UK.*

Let the set of individuals be defined as

$$\mathsf{Ind} = \{\mathsf{dirk}, \mathsf{uni\_liv}, \mathsf{yoga\_center}, \mathsf{UK}, \mathsf{el}, \mathsf{el}', \mathsf{l}, \mathsf{ba}\}.$$

Here, el *and* el' *denote employers' letters,* l *stands for a lease and* ba *for a bank account.*

The initial state - ABox $\mathcal{A}$ states that Dirk is a resident of the UK who has gotten two jobs – at the University of Liverpool and in the Yoga Center, but still does not hold a bank account in the UK.

$$\begin{aligned}\mathcal{A} \;:=\; \{&\mathsf{resident}(\mathsf{dirk}, \mathsf{UK}), \mathsf{employs}(\mathsf{uni\_liv}, \mathsf{dirk}), \mathsf{employs}(\mathsf{yoga\_center}, \mathsf{dirk}),\\ &\mathsf{University}(\mathsf{uni\_liv}), \neg\exists\mathsf{holds}.(\mathsf{B\_acc} \sqcap \exists\mathsf{in}.\{\mathsf{UK}\})(\mathsf{dirk})\}\end{aligned}$$

Moreover, the set Op *contains the operators for obtaining a lease, a letter from employer, and a bank account. The set of occlusions* occ *is empty for all three operators, so we will state only their sets of pre- and post-conditions* pre *and* post*.*

- *Suppose the pre-condition of obtaining a lease is that the customer x holds a letter from his employer. This is formalized by the operator* $\mathsf{get\_Lease}$*:*

$$\begin{aligned}\mathsf{pre}: &\quad \{\exists\mathsf{holds}.\mathsf{EmployerLetter}(x)\}\\ \mathsf{post}: &\quad \{\mathsf{holds}(x, y), \mathsf{Lease}(y)\}\end{aligned}$$

- *The operator* $\mathsf{get\_Letter}$ *describes the process of an employee x getting a letter y from his employer z:*

$$\begin{aligned}\mathsf{pre}: &\quad \{\mathsf{employs}(z, x)\}\\ \mathsf{post}: &\quad \{\mathsf{holds}(x, y), \mathsf{EmployerLetter}(y), \mathsf{signed}(z, y)\}\end{aligned}$$

- *Suppose the pre-condition of opening a bank account is that the customer x is a resident in the UK and holds a proof of address. Moreover, suppose that, if x is rated as "reliable", then the bank account comes with a credit card, otherwise not. This service can be formalized by the following operator* $\mathsf{get\_B\_acc}$*:*

$$\begin{aligned}\mathsf{pre}: &\quad \{\exists\mathsf{resident}.\{\mathsf{UK}\}(x), \exists\mathsf{holds}.\mathsf{Proof\_address}(x)\}\\ \mathsf{post}: &\quad \{\mathsf{holds}(x, y), \mathsf{in}(y, \mathsf{UK}),\\ &\qquad \mathsf{Reliable}(x)/\mathsf{B\_acc\_credit}(y),\\ &\qquad \neg\mathsf{Reliable}(x)/\mathsf{B\_acc\_no\_credit}(y)\}\end{aligned}$$

The meaning of the concepts used in $\mathcal{A}$ *and* Op *is defined in the following acyclic TBox* $\mathcal{T}$*:*

$$\begin{aligned}
\mathsf{Reliable} &\doteq \exists \mathsf{holds}.(\mathsf{B_acc} \sqcap \mathsf{Good_credit_rating} \sqcap \exists \mathsf{in}.\{\mathsf{UK}\}) \\
&\quad \sqcup \exists \mathsf{holds}.(\mathsf{EmployerLetter} \sqcap \exists \mathsf{signed}^{-}.\mathsf{University}) \\
\mathsf{Proof_address} &\doteq \mathsf{Electricity_contract} \sqcup \mathsf{Lease} \\
\mathsf{B_acc} &\doteq \mathsf{B_acc_credit} \sqcup \mathtt{B_acc_no_credit}
\end{aligned}$$

The first concept definition tells us that a person is rated as reliable if and only if he already holds a bank account in the UK with a good credit rating, or holds a letter stating that he is employed at the university. The second definition defines a proof of the address to be either an electricity contract or a lease, while the last one states that a bank account can come either with or without a credit card.

Finally, we have two goals, $\Gamma_1 = \{\exists \mathsf{holds}.(\mathsf{B\_acc} \sqcap \exists \mathsf{in}.\{\mathsf{UK}\})(\mathsf{dirk})\}$*, requiring that Dirk holds a bank account in the UK, and a more ambitious one,* $\Gamma_2 = \{\exists \mathsf{holds}.(\mathsf{B\_acc\_credit} \sqcap \exists \mathsf{in}.\{\mathsf{UK}\})(\mathsf{dirk})\}$*, namely that Dirk holds a bank account in the UK with a credit card. We define corresponding planing tasks* Π_1 *and* Π_2 *as* $\Pi_1 = (\mathsf{Ind}, \mathcal{T}, \mathsf{Op}, \mathcal{A}, \Gamma_1)$ *and* $\Pi_2 = (\mathsf{Ind}, \mathcal{T}, \mathsf{Op}, \mathcal{A}, \Gamma_2)$*. It is not difficult to see that the plan:*

$$\mathsf{get\_Letter}[x/\mathsf{dirk}, y/\mathsf{el}, z/\mathsf{yoga\_center}], \mathsf{get\_Lease}[x/\mathsf{dirk}, y/\mathsf{l}], \mathsf{get\_B\_acc}[x/\mathsf{dirk}, y/\mathsf{ba}]$$

is a solution to Π_1*, but not* Π_2*, while the plan:*

$$\mathsf{get\_Letter}[x/\mathsf{dirk}, y/\mathsf{el}', z/\mathsf{uni\_liv}], \mathsf{get\_Lease}[x/\mathsf{dirk}, y/\mathsf{l}], \mathsf{get\_B\_acc}[x/\mathsf{dirk}, y/\mathsf{ba}]$$

is a solution both to Π_1 *and* Π_2*.*

The most common planning problem, PLANEX, c.f. [ENS95], is defined below:

Definition 6.1.4 (PLANEX). The *plan existence problem (PLANEX)*, is the problem of whether a given planning task Π has a solution.

If Π is an unconditional planning task, we call it *unconditional PLANEX*, and otherwise *conditional*[3] *PLANEX.* △

In the next two sections, we will present algorithms for deciding unconditional and conditional PLANEX. In order to be able to determine the computational complexity of the algorithms (and PLANEX), we define the size of a planning task as follows.

Definition 6.1.5 (Size of Planning Task). Let $\Pi = (\mathsf{Ind}, \mathcal{T}, \mathsf{Op}, \mathcal{A}, \Gamma)$ be a planning task. The *size* of Π (written $|\Pi|$) is defined as:

$$|\Pi| := |\mathsf{Ind}| + |\mathcal{T}| + \sum_{o \in \mathsf{Op}} |o| + |\mathcal{A}| + |\Gamma|$$

△

[3]Note that, in the literature on planning, the term "conditional planning" is usually used in a different context: it describes the existence of a precedence relation on actions which determines in which order they have to be executed.

6.2 Unconditional PLANEX

In this section, we focus on the plan existence problem in the case operators have only unconditional post-conditions. It turns out that unconditional PLANEX is not harder, at least in theory, than projection in the propositionally closed fragments of $\mathcal{ALCQIO}$.

Obviously, the plan existence problem is closely related to projection and executability. We start by introducing some notation. Let $\mathcal{A}$ and $\mathcal{B}$ be ABoxes, $\mathcal{T}$ an acyclic TBox, and α a (possibly composite) action. We will write $\mathcal{T}, \mathcal{A}^\alpha \models \mathcal{B}$ iff $\mathcal{T}, \mathcal{A}^\alpha \models \varphi$ (φ is a consequence of applying α in $\mathcal{A}$ w.r.t. $\mathcal{T}$) for all $\varphi \in \mathcal{B}$.

Let $\Pi = (\mathsf{Ind}, \mathcal{T}, \mathsf{Op}, \mathcal{A}, \Gamma)$ be a planning task for which we want to decide if it has a solution. This means that we want to check if there is a finite sequence of actions from $\mathsf{Op}[\mathsf{Ind}]$ which transform the initial state (described by $\mathcal{A}$) into a state where goal Γ holds.

Intuitively, planning is related to a step-wise computation of the next state – which corresponds to computing updated ABoxes (c.f. Section 1.3.3). However, in [LLMW06c], it is shown that an updated ABox may be exponentially large in the size of the initial ABox and the update, which makes this approach unsuitable. We base our approach on the following observation: instead of computing a sequence of (exponentially large) updated ABoxes, it suffices to compute a sequence of *updates* which are applied to the initial ABox $\mathcal{A}$. Intuitively, these updates are lists of accumulated triggered post-conditions of action sequences. Since post-conditions are unconditional, every chosen action sequence can be represented by the same update, independently of the model of the initial ABox. Similarly, we keep track of accumulated occlusions. Thus, *states of the search space* can compactly be described as pairs: (occlusion, update).

We define the set of possible atomic changes as:

$$\mathcal{L}_{\mathsf{post}} := \{\psi \mid \psi \in \mathsf{post},\ \alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post}),\ \alpha \in \mathsf{Op}[\mathsf{Ind}]\}$$

and the set of possible occlusions:

$$\mathcal{L}_{\mathsf{occ}} := \{\psi \mid \psi \in \mathsf{occ},\ \alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post}),\ \alpha \in \mathsf{Op}[\mathsf{Ind}]\}$$

An *update* for Π is a consistent subset of $\mathcal{L}_{\mathsf{post}}$. Let $\mathfrak{U}$ be a set of all updates for Π. Moreover, let $\mathfrak{O} := 2^{\mathcal{L}_{\mathsf{occ}}}$. Then $\mathfrak{O} \times \mathfrak{U}$ is our search space, the size of which is exponential in the size of Π. Let $\dot{\neg} l$ denote $\neg l$ if l is a positive assertion, and l' if $l = \neg l'$. For a $\mathcal{U} \in \mathfrak{U}$, we set

$$\begin{aligned} \neg\mathcal{U} &:= \{\dot{\neg} l \mid l \in \mathcal{U}\}; \quad \text{and} \\ \overline{\mathcal{U}} &:= \{l \mid l \in \mathcal{U} \cup \neg\mathcal{U} \text{ and } l \text{ positive}\}. \end{aligned}$$

Intuitively, $(\emptyset, \emptyset)$ represents the initial state of the search space, and all tuples $(\mathcal{O}, \mathcal{U}) \in \mathfrak{O} \times \mathfrak{U}$ such that $\mathcal{T}, \mathcal{A}^{(\emptyset, \mathcal{O}, \mathcal{U})} \models \Gamma$ represent goal states.

In the next step, we define the transition relation "$\xrightarrow{\alpha}$" on $\mathfrak{O} \times \mathfrak{U}$. Let $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ be a consistent action from $\mathsf{Op}[\mathsf{Ind}]$ [4] and let $(\mathcal{O}, \mathcal{U}), (\mathcal{O}', \mathcal{U}') \in \mathfrak{O} \times \mathfrak{U}$. We say that α transforms $(\mathcal{O}, \mathcal{U})$ into $(\mathcal{O}', \mathcal{U}')$ (written $(\mathcal{O}, \mathcal{U}) \xrightarrow{\alpha} (\mathcal{O}', \mathcal{U}')$) iff:

(i) $\mathcal{O}' = (\mathcal{O} \cup \mathsf{occ}) \setminus \overline{\mathsf{post}}$

(ii) $\mathcal{U}' = (\mathcal{U} \setminus (\mathsf{occ} \cup \neg\mathsf{occ} \cup \neg\mathsf{post})) \cup \mathsf{post}$

[4] Since post is a set of *unconditional* post-conditions, α is consistent iff post is consistent.

Obviously, the relation "$\xrightarrow{\alpha}$" is functional for every α. In the following lemma, we show that "$\xrightarrow{\alpha}$" simulates "$\Rightarrow^{\mathcal{T}}_{\alpha}$" on the set $\mathfrak{O} \times \mathfrak{U}$.

Lemma 6.2.1. *Let* $\Pi = (\mathsf{Ind}, \mathcal{T}, \mathsf{Op}, \mathcal{A}, \Gamma)$ *be a planning task and let* $\alpha = \alpha_1, \ldots, \alpha_k$ *with* $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ *be a plan in* Π. *Let* $\mathcal{U}_0 = \mathcal{O}_0 := \emptyset$ *and let* $(\mathcal{O}_1, \mathcal{U}_1), \ldots, (\mathcal{O}_k, \mathcal{U}_k)$ *be such that*

$$(\mathcal{O}_0, \mathcal{U}_0) \xrightarrow{\alpha_1} (\mathcal{O}_1, \mathcal{U}_1) \cdots \xrightarrow{\alpha_k} (\mathcal{O}_k, \mathcal{U}_k)$$

Then the following holds:

(a) $(\mathcal{O}_i, \mathcal{U}_i) \in \mathfrak{O} \times \mathfrak{U}$ *for all* $i \leq k$*:*

(b) For all interpretations $\mathcal{I}, \mathcal{I}'$ *such that* $\mathcal{I} \models \mathcal{A}$ *and for all* $i \leq k$*, we have that* $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_i} \mathcal{I}'$ *iff* $\mathcal{I} \Rightarrow^{\mathcal{T}}_{(\emptyset,\mathcal{O}_i,\mathcal{U}_i)} \mathcal{I}'$.

(c) $\mathcal{T}, \mathcal{A}^{(\emptyset,\mathcal{O}_i,\mathcal{U}_i)} \models \mathsf{pre}_{i+1}$ *for all* $i < k$ *iff* $\alpha_1, \ldots, \alpha_k$ *is executable in* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$.

Proof. (a) Since $\alpha_1, \ldots, \alpha_k$ are consistent by definition of a plan, it is easy to see that the definition of $\xrightarrow{\alpha_i}$ implies that $(\emptyset, \mathcal{O}_i, \mathcal{U}_i)$ are consistent for all $i \leq k$. Thus, $(\mathcal{O}_i, \mathcal{U}_i) \in \mathfrak{O} \times \mathfrak{U}$ for all $i \leq k$.

(b) Proof is by induction on i. For $i = 0$, trivially true. Assume that the claim holds for $i = m$, and let us prove that it implies the same for $i = m + 1$. Let $\mathcal{I} \models \mathcal{A}$ and let $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_{m+1}} \mathcal{I}'$. The latter holds iff there exists $\mathcal{I}''$ such that $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_m} \mathcal{I}''$ and $\mathcal{I}'' \Rightarrow^{\mathcal{T}}_{\alpha_{m+1}} \mathcal{I}'$. By I.H., we have that for $\mathcal{I} \models \mathcal{A}$ it holds that $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_m} \mathcal{I}''$ iff $\mathcal{I} \Rightarrow^{\mathcal{T}}_{(\emptyset,\mathcal{O}_m,\mathcal{U}_m)} \mathcal{I}''$. Finally, since $(\emptyset, \mathcal{O}_m, \mathcal{U}_m)$ is consistent, the points (i) and (ii) of the definition of $\xrightarrow{\alpha_{m+1}}$ imply that there exists $\mathcal{I}''$ such that $\mathcal{I} \Rightarrow^{\mathcal{T}}_{(\emptyset,\mathcal{O}_m,\mathcal{U}_m)} \mathcal{I}''$ and $\mathcal{I}'' \Rightarrow^{\mathcal{T}}_{(\emptyset,\mathsf{occ}_{m+1},\mathsf{post}_{m+1})} \mathcal{I}'$ iff it holds that $\mathcal{I} \Rightarrow^{\mathcal{T}}_{(\emptyset,\mathcal{O}_{m+1},\mathcal{U}_{m+1})} \mathcal{I}'$.

(c) follows directly from (b) and the definition of executability. ❏

A non-deterministic procedure which decides whether the planning task Π has a solution is presented in Figure 6.1. The procedure searches for an executable sequence of consistent actions from $\mathsf{Op}[\mathsf{Ind}]$ which transforms the initial state $S_0 = (\emptyset, \emptyset)$ into a state $\mathcal{S}_\Gamma = (\mathcal{O}_\Gamma, \mathcal{U}_\Gamma) \in \mathfrak{O} \times \mathfrak{U}$ such that $\mathcal{T}, \mathcal{A}^{\mathcal{S}_\Gamma} \models \Gamma$ [5] (goal state). We use ϵ do denote the empty action $(\emptyset, \emptyset, \emptyset)$. Since the size of the search space $\mathfrak{O} \times \mathfrak{U}$ is bounded by $2^{|\mathcal{L}_{\mathsf{occ}}|+|\mathcal{L}_{\mathsf{post}}|}$, there is no need to search for longer sequences than $n := 2^{|\mathcal{L}_{\mathsf{occ}}|+|\mathcal{L}_{\mathsf{post}}|}$.

Lemma 6.2.2. *Let* $\Pi = (\mathsf{Ind}, \mathcal{T}, \mathsf{Op}, \mathcal{A}, \Gamma)$ *be an unconditional planning task. Then* Π *has a solution iff* **PLANEX**(Π) *returns "TRUE".*

Proof. "$\Rightarrow$" Let the plan $\alpha_1, \ldots, \alpha_k$ be a solution to Π such that $k \leq n$. Let $\alpha_1, \ldots, \alpha_n = \alpha_1, \ldots, \alpha_k, \epsilon, \ldots, \epsilon$ be obtained by appending $n - k$ empty actions to $\alpha_1, \ldots, \alpha_k$. Since $\alpha_1, \ldots, \alpha_k$ is a solution to Π, we have that (i) $\alpha_1, \ldots, \alpha_n$ is executable in $\mathcal{A}$ w.r.t. $\mathcal{T}$; (ii) $\mathcal{T}, \mathcal{A}^{\alpha_1,\ldots,\alpha_n} \models \Gamma$; and (iii) α_i is consistent for $1 \leq i \leq n$. By Lemma 6.2.1, we have that the run of **PLANEX**(Π) which guesses the sequence of actions $\alpha_1, \ldots, \alpha_n$, returns "TRUE".

"$\Leftarrow$" Let **PLANEX**(Π) returns "TRUE". Then there exists a sequence of consistent actions $\alpha_1, \ldots, \alpha_n$ such that $\mathcal{S}_0 \xrightarrow{\alpha_1} \mathcal{S}_1 \cdots \xrightarrow{\alpha_m} \mathcal{S}_n$ and (i) $\mathcal{T}, \mathcal{A}^{\mathcal{S}_{i-1}} \models \mathsf{pre}_i$ for $1 \leq i < n$; (ii) $\mathcal{T}, \mathcal{A}^{\mathcal{S}_n} \models \Gamma$. Let $i_1 < \cdots < i_k$ be such that $\{i_1, \ldots, i_k\}$ is maximal subset of $\{1, \ldots, n\}$ with $\alpha_i \in \mathsf{Op}[\mathsf{Ind}]$ (or alternatively $\alpha_i \neq \epsilon$, if $\epsilon \notin \mathsf{Op}[\mathsf{Ind}]$). By Lemma 6.2.1, we have that $\alpha_{i_1}, \ldots, \alpha_{i_k}$ is a solution to Π. ❏

[5] From now on, if $\mathcal{S} = (\mathcal{O}, \mathcal{U})$, we write $\mathcal{S}$ as an abbreviation for the action $(\emptyset, \mathcal{O}, \mathcal{U})$.

PLANEX(Π)
  $i := 0$; $\mathcal{S}_0 := (\emptyset, \emptyset)$; $n := 2^{|\mathcal{L}_{\mathsf{occ}}|+|\mathcal{L}_{\mathsf{post}}|}$;
  while $i < n$
    guess $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post}) \in \mathsf{Op}[\mathsf{Ind}] \cup \{\epsilon\}$
    if post inconsistent or $\mathcal{T}, \mathcal{A}^{\mathcal{S}_i} \not\models \mathsf{pre}$
      then return "FALSE"
    compute $\mathcal{S}_{i+1}$ such that $\mathcal{S}_i \xrightarrow{\alpha} \mathcal{S}_{i+1}$
    $i := i + 1$
  if $\mathcal{T}, \mathcal{A}^{\mathcal{S}_n} \not\models \Gamma$
    then return "FALSE"
  return "TRUE"

Figure 6.1: (Non-deterministic) Unconditional PLANEX Algorithm

Clearly, **PLANEX**(Π) works in NPSpace with a "projection oracle". If projection is in PSpace, then PLANEX is obviously in NPSpace. By using Savitch's result [Sav70] that PSpace = NPSpace, we obtain that PLANEX is then in PSpace. Similarly, if projection is in ExpTime, since NPSpace ⊆ ExpTime, we have that PLANEX can be decided in ExpTime.

Finally, we will show the less straightforward result that PLANEX is in co-NExpTime if projection is in co-NExpTime. To this end, we develop an alternative NExpTime algorithm which returns "TRUE" iff the planning task Π has *no solution*. Note that Π has no solution iff in every run of the **PLANEX**(Π) algorithm (corresponding to a different plan $\alpha_1, \ldots, \alpha_n$ in Π), at least one of the tests $\mathcal{T}, \mathcal{A}^{\mathcal{S}_i} \models \mathsf{pre}_{i+1}$ or $\mathcal{T}, \mathcal{A}^{\mathcal{S}_n} \models \Gamma$ fails.

Let $\mathfrak{S} := \mathfrak{O} \times \mathfrak{U}$ and $\mathfrak{Q} := \{\mathsf{pre} \mid \alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post}) \in \mathsf{Op}[\mathsf{Ind}]$ and α consistent$\} \cup \{\Gamma\}$. The alternative (non-deterministic) algorithm **noPLANEX**(Π) has three steps:

(i) guess a subset T of $\mathfrak{S} \times \mathfrak{Q}$;

(ii) check whether for all tuples $(\mathcal{S}, \mathcal{Q}) \in T$ it holds that $\mathcal{T}, \mathcal{A}^{\mathcal{S}} \not\models \mathcal{Q}$;

(iii) check if the following holds: in every run of the original **PLANEX**(Π) procedure, there is at least one projection test $\mathcal{T}, \mathcal{A}^{\mathcal{S}} \models \mathcal{Q}?$ such that $(\mathcal{S}, \mathcal{Q}) \in T$.

If (ii) and (iii) give positive answers, **noPLANEX**(Π) returns "TRUE", and otherwise "FALSE". It is not difficult to see that (i) and (ii) can be executed in NExpTime and that (iii) can be checked in ExpTime. Thus, **noPLANEX**(Π) can be executed in NExpTime. We obtained the following lemma:

Lemma 6.2.3. *Let* $\mathcal{L} \in \{\mathcal{ALC}, \mathcal{ALCO}, \mathcal{ALCI}, \mathcal{ALCQ}, \mathcal{ALCIO}, \mathcal{ALCQO}, \mathcal{ALCQI}, \mathcal{ALCQIO}\}$. *Unconditional PLANEX in* $\mathcal{L}$ *has the same upper complexity bound as projection in* $\mathcal{L}$.

We show that the upper complexity bounds established in Lemma 3.2.1 are tight by the following easy reduction of projection to PLANEX. Let $\mathcal{A}$ be an ABox, α an update (i.e., an action with empty pre-conditions and empty occlusions and only with unconditional post-conditions), and φ an assertion, such that $\mathcal{A}^{\alpha} \not\models \varphi$. We define the planning task $\Gamma_{\mathcal{A},\alpha,\varphi}$ as

$$\Gamma_{\mathcal{A},\alpha,\varphi} := (\emptyset, \emptyset, \{\alpha\}, \mathcal{A}, \{\varphi\}).$$

Note that, in case of an update, it does not make a difference whether it is applied one or more times. It is not difficult to see that $\mathcal{A}^\alpha \models \varphi$ iff $\Gamma_{\mathcal{A},\alpha,\varphi}$ has a solution.

Since the lower bounds for projection from Theorem 3.2.1 hold already in the case of the empty TBox and an update, we conclude that the complexity bounds from Lemma 6.2.3 are optimal, i.e., the unconditional plan existence problem is of exactly the same computational complexity as projection.

Theorem 6.2.4. *The unconditional plan existence problem is:*

(a) PSPACE*-complete in* $\mathcal{ALC}$*,* $\mathcal{ALCO}$*,* $\mathcal{ALCQ}$*, and* $\mathcal{ALCQO}$*;*

(b) EXPTIME*-complete in* $\mathcal{ALCI}$ *and* $\mathcal{ALCIO}$*;*

*(c) co-*NEXPTIME*-complete in* $\mathcal{ALCQI}$ *and* $\mathcal{ALCQIO}$*.*

6.3 Conditional PLANEX

If we allow for conditional post-conditions in operators, the complexity results from the previous section do not hold anymore. With conditional post-conditions, already in the propositional case, conformant PLANEX is EXPSPACE-hard [HJ99, Rin04]. In this section we will show that conditional PLANEX is decidable for DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$. Decidability will be shown by a 2-EXPSPACE algorithm.

Let $\Pi = (\mathsf{Ind}, \mathcal{T}, \mathsf{Op}, \mathcal{A}, \Gamma)$ be a conditional planning task for which we want to decide if it has a solution. For the sake of simplicity, we assume that occlusions in operators from Op are empty, i.e. operators are of the form $(\mathsf{pre}, \mathsf{post})$. Non-empty occlusions can be treated similarly as in the previous section. We will also use abbreviations introduced in the previous section.

In case of *unconditional* planning, which was treated in the previous section, the states of a search space could compactly be described as updates (assuming that occlusions are empty). Intuitively, a state (update) $\mathcal{U}$ stands for the interpretation $\mathcal{I}^{\mathcal{U}}_{\mathcal{T}}$[6], for *all* initial models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$. If $\mathcal{U}$ describes the state obtained by applying a sequence of actions $\alpha_1, \ldots, \alpha_k$ in $\mathcal{I}$, then $\mathcal{U}$ is simply obtained by "merging" the unconditional post-conditions of $\alpha_1, \ldots, \alpha_k$. Thus, to every fixed sequence of actions $\alpha_1, \ldots, \alpha_k$, the same $\mathcal{U}$ corresponds, independently of the initial interpretation $\mathcal{I}$. However, in *conditional* planning, this is not the case: different initial models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ cause different combinations of α_i-post-conditions to be triggered, and thus there is *no single* update which can compactly describe the effect of applying $\alpha_1, \ldots, \alpha_k$ in all initial models $\mathcal{I}$. Instead, we partition the set of all initial $\mathcal{I}$ into finitely (double-exponentially) many classes $\mathfrak{I}_i$, such that for all $\mathcal{I}$ from the same class $\mathfrak{I}_i$, the same update $\mathcal{U}_i$ describes the effects of $\alpha_1, \ldots, \alpha_k$. Thus, the search space will be a mapping, assigning an update $\mathcal{U}_i$ to every class $\mathfrak{I}_i$.

Formally, we set

$$\mathcal{L}_{\mathsf{post}} := \{\psi \mid \varphi/\psi \in \mathsf{post}, \alpha = (\mathsf{pre}, \mathsf{post}), \alpha \in \mathsf{Op}[\mathsf{Ind}]\}$$

and

$$\mathcal{C}_{\mathsf{post}} := \{\varphi \mid \varphi/\psi \in \mathsf{post}, \alpha = (\mathsf{pre}, \mathsf{post}), \alpha \in \mathsf{Op}[\mathsf{Ind}]\}.$$

[6]Recall that $\mathcal{I}^{\mathcal{U}}_{\mathcal{T}}$ denotes the unique interpretation $\mathcal{I}'$ such that $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\mathcal{U}} \mathcal{I}'$.

An *update* in Π is a consistent subset of $\mathcal{L}_{\mathsf{post}}$. Let $\mathfrak{U}$ be the set of all updates in Π. A *context* $\mathcal{C}$ in Π is a consistent subset of $\mathcal{C}_{\mathsf{post}} \cup \neg\mathcal{C}_{\mathsf{post}}$ such that for every $\varphi \in \mathcal{C}_{\mathsf{post}}$, it is the case that either $\varphi \in \mathcal{C}$ or $\neg\varphi \in \mathcal{C}$. Moreover, let $\mathfrak{C}$ be the set of all contexts in Π.

Let m be a mapping $m : \mathfrak{U} \to \mathfrak{C}$. With m we associate a set $\mathfrak{I}(m)$ of models of $\mathcal{A}$ and $\mathcal{T}$ defined with:

$$\mathfrak{I}(m) := \{\mathcal{I} \mid \mathcal{I} \models \mathcal{T}, \mathcal{A} \text{ and } \mathcal{I}_{\mathcal{T}}^{\mathcal{U}} \models m(\mathcal{U}) \text{ for all } \mathcal{U} \in \mathfrak{U}\}$$

We say that m is *admissible* iff $\mathfrak{I}(m) \neq \emptyset$, i.e. m stands for a non-empty class of models of $\mathcal{A}$ and $\mathcal{T}$. Intuitively, if $m(\mathcal{U}) = \mathcal{C}$, it means that after updating an $\mathcal{I} \in \mathfrak{I}(m)$ with $\mathcal{U}$, all assertions from $\mathcal{C}$ will hold. Let $\mathfrak{M}$ be the set of all admissible mappings $m : \mathfrak{U} \to \mathfrak{C}$.

Lemma 6.3.1. *Let $\mathcal{I}$ be a model of $\mathcal{T}$ and $\mathcal{A}$. Then there exists a* unique *$m \in \mathfrak{M}$ such that $\mathcal{I} \in \mathfrak{I}(m)$.*

Proof. For $\mathcal{U} \in \mathfrak{U}$, we set: $m(\mathcal{U}) = \{\varphi \mid \varphi \in \mathcal{C}_{\mathsf{post}} \cup \neg\mathcal{C}_{\mathsf{post}} \text{ and } \mathcal{I}_{\mathcal{T}}^{\mathcal{U}} \models \varphi\}$. Then $m(\mathcal{U})$ is obviously a context in Π for all $\mathcal{U} \in \mathfrak{U}$, and thus m maps $\mathfrak{U}$ to $\mathfrak{C}$. Moreover, it is obvious that $\mathcal{I} \in \mathfrak{I}(m)$, which implies that m is admissible, i.e., $m \in \mathfrak{M}$.

It remains to show that m is unique. Assume, to the contrary, that there exists an $m' \in \mathfrak{M}$, $m \neq m'$, such that $\mathcal{I} \in \mathfrak{I}(m')$. Let $\mathcal{U} \in \mathfrak{U}$ be such that $m(\mathcal{U}) \neq m'(\mathcal{U})$. Since $m(\mathcal{U})$ and $m'(\mathcal{U})$ are both contexts in Π, this means that there is a $\hat{\varphi} \in \mathcal{C}_{\mathsf{post}} \cup \neg\mathcal{C}_{\mathsf{post}}$ such that $\hat{\varphi} \in m(\mathcal{U})$ and $\neg\hat{\varphi} \in m'(\mathcal{U})$. But then $\mathcal{I} \in \mathfrak{I}(m) \cap \mathfrak{I}(m')$ implies that $\mathcal{I}_{\mathcal{T}}^{\mathcal{U}} \models \hat{\varphi}$ and $\mathcal{I}_{\mathcal{T}}^{\mathcal{U}} \models \neg\hat{\varphi}$, which is a contradiction. ❏

The search space $\mathfrak{S}$ is the set of all mappings $\mathcal{S} : \mathfrak{M} \to \mathfrak{U}$. Every $\mathcal{S} \in \mathfrak{S}$ should be understood in the following way: if the "initial model" $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ is in $\mathfrak{I}(m)$, then the state described by $\mathcal{S}$ corresponds to $\mathcal{I}_{\mathcal{T}}^{\mathcal{U}}$, where $\mathcal{U} = \mathcal{S}(m)$.

If $\mathcal{B}$ is an ABox, we write $\mathcal{T}, \mathcal{A}^{\mathcal{S}} \models^* \mathcal{B}$ iff for all $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{A}, \mathcal{T}$ the following holds: if m is (the unique) element of $\mathfrak{M}$ such that $\mathcal{I} \in \mathfrak{I}(m)$, then $\mathcal{I}_{\mathcal{T}}^{\mathcal{S}(m)} \models \mathcal{B}$.

Intuitively, $\mathcal{S}_0 \in \mathfrak{S}$ such that $\mathcal{S}_0(m) = \emptyset$ for all $m \in \mathfrak{M}$ represents the initial state of the search space, and all $\mathcal{S} \in \mathfrak{S}$ such that $\mathcal{T}, \mathcal{A}^{\mathcal{S}} \models^* \Gamma$ represent goal states. Similarly as in the previous section, we define the transition relation $\xrightarrow{\alpha}$ on $\mathfrak{S} \times \mathfrak{S}$. For $\mathcal{S}, \mathcal{S}' \in \mathfrak{S}$, $m \in \mathfrak{M}$, and $\alpha = (\mathsf{pre}, \mathsf{post})$ we set

$$\mathsf{post}_{\alpha,\mathcal{S},m} := \{\psi \mid \varphi/\psi \in \mathsf{post}, \varphi \in m(\mathcal{S}(m))\}.$$

We say that α transforms $\mathcal{S}$ into $\mathcal{S}'$ (written $\mathcal{S} \xrightarrow{\alpha} \mathcal{S}'$) iff

$$\mathcal{S}'(m) = (\mathcal{S}(m) \setminus \neg\mathsf{post}_{\alpha,\mathcal{S},m}) \cup \mathsf{post}_{\alpha,\mathcal{S},m} \quad \text{for all } m \in \mathfrak{M}$$

Then the following lemma holds:

Lemma 6.3.2. *Let $\Pi = (\mathsf{Ind}, \mathcal{T}, \mathsf{Op}, \mathcal{A}, \Gamma)$ be a planning task and let $\alpha = \alpha_1, \ldots, \alpha_k$ with $\alpha_i = (\mathsf{pre}_i, \mathsf{post}_i)$ be a plan in Π. Let $\mathcal{S}_0 \in \mathfrak{S}$ be defined with $\mathcal{S}_0(m) = \emptyset$ for all $m \in \mathfrak{M}$. Moreover, let $\mathcal{S}_1, \ldots, \mathcal{S}_k$ be such such that*

$$\mathcal{S}_0 \xrightarrow{\alpha_1} \mathcal{S}_1 \cdots \xrightarrow{\alpha_k} \mathcal{S}_k.$$

Then the following holds:

(a) $\mathcal{S}_i \in \mathfrak{S}$ for all $i \leq k$:

condPLANEX(Π)
  $i := 0$; $\mathcal{S}_0(m) = \emptyset$ for all $m \in \mathfrak{M}$;
  while $i < 2^{|\mathcal{L}_{\mathsf{post}}| \cdot 2^{|\mathcal{C}_{\mathsf{post}}|} \cdot 2^{|\mathcal{L}_{\mathsf{post}}|}}$
    if $\mathcal{T}, \mathcal{A}^{\mathcal{S}_i} \models^* \Gamma$
      then return "TRUE"
    guess a $\alpha = (\mathsf{pre}, \mathsf{post}) \in \mathsf{Op}[\mathsf{Ind}]$
    if α inconsistent or $\mathcal{T}, \mathcal{A}^{\mathcal{S}_i} \not\models^* \mathsf{pre}_{i+1}$
      then return "FALSE"
    compute $\mathcal{S}_{i+1}$ such that $\mathcal{S}_i \stackrel{\alpha}{\rightarrow} \mathcal{S}_{i+1}$
    $i := i + 1$
  return "FALSE"

Figure 6.2: Conditional PLANEX Algorithm

(b) Let $\mathcal{I}$ be an interpretation such that $\mathcal{I} \models \mathcal{A}, \mathcal{T}$, and let $m \in \mathfrak{M}$ be such that $\mathcal{I} \in \mathfrak{I}(m)$. Then for all interpretations $\mathcal{I}'$ and for all $1 \leq i \leq k$, we have that $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_i} \mathcal{I}'$ iff $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\mathcal{S}_i(m)} \mathcal{I}'$.

(c) $\mathcal{T}, \mathcal{A}^{\mathcal{S}_i} \models^* \mathcal{B}$ iff $\mathcal{T}, \mathcal{A}^{\alpha_1,\ldots,\alpha_i} \models \mathcal{B}$ for an ABox $\mathcal{B}$, for all $i \leq k$.

(d) $\mathcal{T}, \mathcal{A}^{\mathcal{S}_i} \models^* \mathsf{pre}_{i+1}$ for all $i < k$ iff $\alpha_1, \ldots, \alpha_k$ is executable in $\mathcal{A}$ w.r.t. $\mathcal{T}$.

Proof. (a) Since actions $\alpha_1, \ldots, \alpha_k$ are consistent with $\mathcal{T}$ (by definition of a plan in Π), and $m \in \mathfrak{M}$ are admissible we have that $\mathcal{S}_i(m)$ is consistent for all $i \leq k$ and $m \in \mathfrak{M}$, and thus $\mathcal{S}_i(m) \in \mathfrak{U}$. Hence $\mathcal{S}_i \in \mathfrak{S}$ for all $i \leq k$.

(b) Proof by induction on i. For $i = 0$ trivial since $\mathcal{S}_0(m) = \emptyset$ for all $m \in \mathfrak{M}$. Assume that the claim holds for $i = l$. We show that then the same for $i = l + 1$ is implied. Let $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_l,\alpha_{l+1}} \mathcal{I}'$. This holds iff there exists an $\mathcal{I}''$ such that $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_l} \mathcal{I}''$ and $\mathcal{I}'' \Rightarrow^{\mathcal{T}}_{\alpha_{l+1}} \mathcal{I}'$. By I.H., this is further equivalent to $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\mathcal{S}_l(m)} \mathcal{I}''$ and $\mathcal{I}'' \Rightarrow^{\mathcal{T}}_{\alpha_{l+1}} \mathcal{I}'$ (i.e. $\mathcal{I}^{\mathcal{S}_l(m)}_{\mathcal{T}} \Rightarrow^{\mathcal{T}}_{\alpha_{l+1}} \mathcal{I}'$). Since $\mathcal{I} \in \mathfrak{I}(m)$, it holds that $\mathcal{I}^{\mathcal{S}_l(m)}_{\mathcal{T}} \models m(\mathcal{S}_l(m))$. This means that for all $\varphi/\psi \in \mathsf{post}_{l+1}$, it holds that $\mathcal{I}^{\mathcal{S}_l(m)}_{\mathcal{T}} \models \varphi$ iff $\varphi \in m(\mathcal{S}_l(m))$. This implies that $\mathsf{post}_{\alpha_{l+1},\mathcal{S}_l,m} := \{\psi \mid \varphi/\psi \in \mathsf{post}_{l+1},\ \mathcal{I}^{\mathcal{S}_l(m)}_{\mathcal{T}} \models \varphi\}$. By definition of $\stackrel{\alpha_{l+1}}{\rightarrow}$ we have that $\mathcal{I}^{\mathcal{S}_l(m)}_{\mathcal{T}} \Rightarrow^{\mathcal{T}}_{\alpha_{l+1}} \mathcal{I}'$ iff $\mathcal{I}' = \mathcal{I}^{\mathcal{S}_{l+1}(m)}_{\mathcal{T}}$.

(c) is a direct consequence of (b) and the definition of $\models^*$;

(d) is a direct consequence of (c) and the definition of executability. ❑

The number of different mappings $m : \mathfrak{U} \rightarrow \mathfrak{C}$ is at most $2^{|\mathcal{C}_{\mathsf{post}}| \cdot 2^{|\mathcal{L}_{\mathsf{post}}|}}$, and thus $|\mathfrak{M}| \leq 2^{|\mathcal{C}_{\mathsf{post}}| \cdot 2^{|\mathcal{L}_{\mathsf{post}}|}}$. Thus, for the search space $\mathfrak{S}$, we have that $|\mathfrak{S}| \leq 2^{|\mathcal{L}_{\mathsf{post}}| \cdot 2^{|\mathcal{C}_{\mathsf{post}}| \cdot 2^{|\mathcal{L}_{\mathsf{post}}|}}}$. The (non-deterministic) algorithm **condPLANEX**(Π) for conditional PLANEX is presented in Figure 6.2.

Lemma 6.3.3. condPLANEX(Π) *returns "TRUE" iff the conditional planning task Π has a solution.*

Proof. The claim of this lemma holds as an easy consequence of Lemma 6.3.2 and the fact that the size of the search space $|\mathfrak{S}|$ is at most $2^{|\mathcal{L}_{\mathsf{post}}| \cdot 2^{|\mathcal{C}_{\mathsf{post}}| \cdot 2^{|\mathcal{L}_{\mathsf{post}}|}}}$: ❑

Lemma 6.3.4. *For DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$:*

(a) it can be decided in a double exponential space (in the size of the planning task Π) whether a mapping $m : \mathcal{U} \to \mathfrak{C}$ is admissible;

(b) it can be decided in a double exponential space (in the size of Π) whether $\mathcal{T}, \mathcal{A}^{\mathcal{S}} \models^* \mathcal{B}$, where $\mathcal{B}$ is an ABox from the set $\{\mathsf{pre} \mid \alpha = (\mathsf{pre}, \mathsf{post}) \in \mathsf{Op}[\mathsf{Ind}]\} \cup \{\Gamma\}$

Proof. (a) Let $m : \mathfrak{U} \to \mathfrak{C}$. The problem of whether m is admissible can be reduced to ABox consistency. A reduction is analogous to the one presented in Section 3.2.1 and we restrain ourselves from presenting it in detail. The idea is to introduce concept names $A^{(\mathcal{U})}$ and $T_C^{(\mathcal{U})}$ and role names $r^{(\mathcal{U})}$ for all $\mathcal{U} \in \mathfrak{U}$ and for all concept names A, complex concepts C and role names r from Π respectively. They stand for extensions of A, C and r in the interpretation $\mathcal{I}_{\mathcal{T}}^{\mathcal{U}}$, respectively. The reduction acyclic TBox $\mathcal{T}_m$, similarly to $\mathcal{T}_{\mathsf{red}}$ from Section 3.2.1, ensures that $T_C^{(\mathcal{U})}$ is indeed interpreted as C in the interpretation $\mathcal{I}_{\mathcal{T}}^{\mathcal{U}}$. The reduction ABox $\mathcal{A}_m$, analogously to $\mathcal{A}_{\mathsf{red}}$ from Section 3.2.1, ensures that the assertions from the initial ABox hold in the interpretation $\mathcal{I}_{\mathcal{T}}^{\emptyset}$, and that $A^{(\emptyset)}$ and $A^{(\mathcal{U})}$ (and $r^{(\emptyset)}$ and $r^{(\mathcal{U})}$) are related in the correct way for every $\mathcal{U} \in \mathfrak{U} \setminus \{\emptyset\}$. Finally, $\mathcal{A}_m$ contains the following assertions for every $\mathcal{U} \in \mathfrak{U}$:

$$\{T_C^{(\mathcal{U})}(a) \mid C(a) \in m(\mathcal{U})\} \cup \{r^{(\mathcal{U})}(a,b) \mid r(a,b) \in m(\mathcal{U})\} \cup \{\neg r^{(\mathcal{U})}(a,b) \mid \neg r(a,b) \in m(\mathcal{U})\}$$

Then $\mathcal{A}_m$ is consistent w.r.t. $\mathcal{T}_m$ iff m is admissible. Moreover, the size of $\mathcal{A}_m$ and $\mathcal{T}_m$ is exponential in the size of Π (since $|\mathfrak{U}|$ is exponential in $|\Pi|$). Since in $\mathcal{ALCQIO}$ and its sublogics, ABox consistency can be decided in a space exponential in the size of input ABox and TBox, we conclude that admissibility of m can be decided in a space double exponential in $|\Pi|$.

(b) It can be decided whether $\mathcal{T}, \mathcal{A}^{\mathcal{S}} \models^* \mathcal{B}$ in the following way: For every $m \in \mathfrak{M}$, we construct an ABox $\mathcal{A}_m$ and an acyclic TBox $\mathcal{T}_m$ as described in (a). Moreover, let $T_{\mathcal{B}}^{\mathcal{S}(m)}$ be defined analogously to $T_{\mathcal{B}}^{(n)}$ from Observation 3.2.4. It is not difficult to see that $\mathcal{T}, \mathcal{A}^{\mathcal{S}} \models^* \mathcal{B}$ iff $\mathcal{T}_m, \mathcal{A}_m \models T_{\mathcal{B}}^{\mathcal{S}(m)}$ for all $m \in \mathfrak{M}$. Obviously, this boils down to double-exponentially many ABox consequence checks (each of them can be performed in a space double exponential in $|\Pi|$). We have thus shown that (b) holds. ❑

Lemma 6.3.4 (a) implies the set $\mathfrak{M}$ of admissible mappings $m : \mathfrak{U} \to \mathfrak{C}$ can be constructed and stored in 2-ExpSpace. Moreover, the search space $\mathfrak{S}$ is 3-exponential in the size of Π, and thus **condPLANEX**(Π) requires 2-NExpSpace with a "$\models^*$ oracle". Since $\models^*$ can be decided in 2-ExpSpace by Lemma 6.3.4 (b), conditional PLANEX is in 2-NExpSpace. Since 2-NExpSpace =2-ExpSpace [Sav70], we obtain the following theorem:

Theorem 6.3.5. *The conditional plan existence problem is in 2-*ExpSpace *in the DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$.*

The obtained 2-ExpSpace upper complexity bound for conditional PLANEX does not match the ExpSpace lower complexity bound inherited from the propositional logic. We leave the exact computational complexity of conditional PLANEX in $\mathcal{ALC}$ and its extensions as open problem.

6.4 Results on Planning in $\mathcal{EL}$

In this section we will present adaptations of hardness reductions for PLANEX in the propositional logic by Bylander [Byl94] and Rintanen [Rin04] to PLANEX in $\mathcal{EL}$. Thus we show

that (PSPACE and EXPSPACE) hardness results from propositional planning carry over to (conditional and unconditional) planning in $\mathcal{EL}$.

6.4.1 Hardness Results for Unconditional PLANEX in $\mathcal{EL}$

Bylander [Byl94] showed that PLANEX in basic (propositional) STRIPS is PSPACE-hard by reducing the acceptance problem of a deterministic Turing machine with a polynomial space bound to PLANEX. We will present a variation of his reduction, using DL syntax. Since the reduction uses negated assertions only in (unconditional) post-conditions, we obtain that unconditional PLANEX in $\mathcal{EL}$ without TBoxes is also PSPACE-hard.

Recall that a deterministic Turing machine is a tuple $(Q, \Sigma, \delta, q_0, F)$, where

- $Q = \{q_0, \ldots, q_n\}$ a finite set of states;
- $\Sigma = \{\mathsf{blank}, a_1, \ldots, a_l\}$ a finite alphabet;
- $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R\}$ is a transition function;
- q_0 is the initial state;
- $F \subseteq Q$ is the set of final states.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a deterministic Turing machine with a polynomial space bound $p(x)$, and let $a = a_{i_0}, \ldots a_{i_k} \in \Sigma^*$ be an input word. This means that M will use $m = p(k)$ tape cells in its run.

We define a planning task $\Pi_{M,a} = (\mathsf{Ind}_{M,a}, \emptyset, \mathsf{Op}_{M,a}, \mathcal{A}_{M,a}, \Gamma_{M,a})$ such that a planner for Π simulates moves of the Turing Machine M. In the reduction, we use concept names $Q_0, \ldots, Q_n$ to represent the states $q_0, \ldots, q_n$, concept names $\mathsf{Blank}, A_1, \ldots, A_l$, to represent elements of alphabet Σ, concept name Accept which denotes the acceptance of the Turing machine and the role name right denoting the next tape cell on the right. Moreover, individuals $t_0, t_1, \ldots, t_m$ denote the tape cells of M.

The set of individuals $\mathsf{Ind}_{M,a}$, the initial state $\mathcal{A}_{M,a}$, the goal $\Gamma_{M,a}$, and the set of operators $\mathsf{Op}_{M,a}$ are defined with:

$$\begin{aligned}
\mathsf{Ind}_{M,a} &:= \{t_0, t_1, \ldots, t_m\} \\
\mathcal{A}_{M,a} &:= \{Q_0(t_0), A_{i_0}(t_0), \ldots, A_{i_k}(t_k), \mathsf{Blank}(t_{k+1}), \ldots, \mathsf{Blank}(t_m)\} \\
&\quad \cup \{\mathsf{right}(t_0, t_1), \ldots, \mathsf{right}(t_{m-1}, t_m)\} \\
\Gamma_{M,a} &:= \{\mathsf{Accept}(t_0)\}
\end{aligned}$$

$$\mathsf{Op}_{M,a} := \bigcup_{q_f \in F} \{\mathsf{accept}_{q_f}(x)\} \cup \bigcup_{\delta(q,a)=(q',b,R)} \{\mathsf{right}_{q,a,q',b}(x,y)\} \cup \bigcup_{\delta(q,a)=(q',b,L)} \{\mathsf{left}_{q,a,q',b}(x,y)\}$$

where the single operators (of the form $(\mathsf{pre}, \mathsf{post})$, $\mathsf{occ} = \emptyset$ for all operators) are defined as follows:[7]

$$\begin{aligned}
\mathsf{right}_{q,a,q',b}(x,y) &:= (\{Q(x), A(x), \mathsf{right}(x,y)\}, \{\neg Q(x), \neg A(x), B(x), Q'(y)\}) \\
\mathsf{left}_{q,a,q',b}(x,y) &:= (\{Q(x), A(x), \mathsf{right}(y,x)\}, \{\neg Q(x), \neg A(x), B(x), Q'(y)\}) \\
\mathsf{accept}_{q_f}(x) &:= (\{Q_f(x)\}, \{\mathsf{Accept}(t_0)\})
\end{aligned}$$

Although the reduction we presented is syntactically very similar to the one from [Byl94], the explanation of why it is correct is slightly different. Unlike in [Byl94], the ABox $\mathcal{A}_{M,a}$ is interpreted w.r.t. open world semantics, which means that it also admits models in which the Turing machine may initially be in more than one state, and in which tape cells can be multiply labelled. However, the potential plan has to achieve the goal starting from *any* model of $\mathcal{A}_{M,a}$, including also its minimal model $\mathcal{I}_0$ satisfying only atomic assertions listed in $\mathcal{A}_{M,a}$. It is not difficult to see that $\mathcal{I}_0$ can be transformed by a sequence of $\mathsf{Op}_{M,a}$ operators into $\mathcal{I}$ such that $\mathcal{I} \models \mathsf{Accept}(t_0)$ iff the Turing machine M accepts the input a. Moreover, a plan successful for $\mathcal{I}_0$ is successful for any other model of $\mathcal{A}_{M,a}$. Hence the Turing machine M accepts the input a iff there is a solution to the planning task $\Pi_{M,a} = (\mathsf{Ind}_{M,a}, \emptyset, \mathsf{Op}_{M,a}, \mathcal{A}_{M,a}, \Gamma_{M,a})$.

As a consequence we obtain the following lemma.

Lemma 6.4.1. *The unconditional plan existence problem in $\mathcal{EL}$ is* PSPACE*-hard, even if the TBox in the planning task is empty.*

The matching upper complexity bound for unconditional PLANEX in $\mathcal{EL}$ with TBoxes follows from results for $\mathcal{ALC}$ in Theorem 6.2.4.

Theorem 6.4.2. *The unconditional plan existence problem in $\mathcal{EL}$ is* PSPACE*-complete, both with empty and non-empty (acyclic) TBoxes.*

6.4.2 Hardness Results for Conditional PLANEX in $\mathcal{EL}$

Rintanen [Rin04] showed that conformant PLANEX in propositional logic is EXPSPACE-hard by reducing the acceptance problem of a deterministic Turing machine with an exponential space bound to conformant PLANEX. Since Rintanen's reduction makes a heavy use of disjunction in order to express the uncertainty about the initial state, as well as negation, we have to find a way to simulate both negation and disjunction in the initial ABox and action pre-conditions. Our adaption of Rintanen's reduction shows that conditional PLANEX in $\mathcal{EL}$ without TBoxes is also EXPSPACE-hard.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a deterministic Turing machine with an exponential space bound $2^{p(x)}$, and let $a = a_{i_0}, \ldots a_{i_k} \in \Sigma^*$ be an input word. This means that M will use $m = 2^{p(k)}$ tape cells in its run. The idea od Rintanen's reduction is to keep track of only one tape cell, called the *watched tape cell*, rather than encoding all tape cells in the planing task. The uncertainty in the initial state of Rintanen's reduction is about which tape cell is watched.

[7] Operators $\mathsf{right}_{q,a,q',b}(x,y)$ and $\mathsf{left}_{q,a,q',b}(x,y)$ are defined under assumption that $a \neq b$. In case $a = b$, the post-condition of these operators should simply be $\{\neg Q(x), Q'(y)\}$.

Let $\Sigma = \{\mathsf{blank}, a_1, \ldots, a_l\}$ and $Q = \{q_0, \ldots, q_n\}$. As in the previous reduction, we use concept names $Q_0, \ldots, Q_n$ to represent the states of Q and the concept name Accept to denote the acceptance of the input. Concept names $H_0, H_1, \ldots, H_{p(k)-1}$ are used as the binary representation of the position of the *head* of the Turing machine, while concept names $W_0, W_1, \ldots, W_{p(k)-1}$ represent the position of the watched tape cell. Contents of the watched tape cell will be represented by concepts $\mathsf{C}_{\mathsf{blank}}, \mathsf{C}_{a_1}, \ldots, \mathsf{C}_{a_l}$. Moreover, we will use concept names $\mathsf{GOright}$ and GOleft to describe that the head can move right or left, respectively. Finally, we introduce auxiliary concept names Start_1, Start_2, Aux, HeqW and OK_c for $c \in \Sigma$, to be explained later. In order to simulate negation in action pre-conditions, we introduce concept names $\overline{H}_0, \overline{H}_1, \ldots, \overline{H}_{p(k)-1}$, and $\overline{W}_0, \overline{W}_1, \ldots, \overline{W}_{p(k)-1}$, such that $\overline{H}_i$ stands for $\neg H_i$, and similar for $\overline{W}_i$.

Let $m < 2^{p(k)}$, and let W_i^m denote W_i if i^{th} bit of m is 1, and $\overline{W}_i$, otherwise. Then the abbreviation $\mathsf{Watched}_{=m}$, defined with

$$\mathsf{Watched}_{=m} := \mathsf{W}_0^m \sqcap \cdots \sqcap \mathsf{W}_{p(k)-1}^m.$$

denotes that the position of the watched tape cell is m. Moreover, let $d = [\log_2(m+1)] + 1$. Then $\mathsf{Watched}_{>m}$ denotes the following set of concepts which describe that the position of the watched tape cell is greater than m:

$$\mathsf{Watched}_{>m} := \{\mathsf{Watched}_{=i} \mid m \leq i \leq 2^d - 1\} \cup \{W_i \mid d \leq i \leq p(k) - 1\}.$$

The reduction we present is essentially propositional, and all ABox assertions we use are of the form $C(m)$, where C is a conjunction of concept names (or a negated concept name), and m is a fixed individual name. For better readability we will abuse the notation and write only C instead of $C(m)$. Moreover, we will use $\varphi/\{\psi_1, \psi_2\}$ to abbreviate $\varphi/\psi_1, \varphi/\psi_2$ in the post-conditions of actions. We define Head^{++} to be a set of conditional post-conditions that increase the position of the head by one:

$$\begin{aligned}\mathsf{Head}^{++} \;:=\; & \{\overline{H}_0/\{H_0, \neg\overline{H}_0\},\ H_0/\{\overline{H}_0, \neg H_0\},\\ & \overline{H}_1 \sqcap H_0/\{H_1, \neg\overline{H}_1\},\ H_1 \sqcap H_0/\{\overline{H}_1, \neg H_1\},\\ & \ldots\\ & \overline{H}_{p(k)-1} \sqcap H_{p(k)-2} \sqcap \cdots \sqcap H_1 \sqcap H_0/\{H_{p(k)-1}, \neg\overline{H}_{p(k)-1}\}\}\end{aligned}$$

In a similar way, we can define Head^{--}, a set of conditional post-conditions that decrease the position of the head by one. Now we define a planning task

$$\Pi_{M,a} = (\emptyset, \emptyset, \mathsf{Op}_{M,a}, \mathcal{A}_{M,a}, \Gamma_{M,a})$$

such that a planner for Π simulates moves of the Turing Machine M. The initial state $\mathcal{A}_{M,a}$, the goal $\Gamma_{M,a}$, and the set of operators $\mathsf{Op}_{M,a}$ are defined with:

$$\begin{aligned}\mathcal{A}_{M,a} \;&:=\; \{\overline{H}_0, \ldots, \overline{H}_{p(k)-1}, \overline{W}_0, \overline{W}_1, \ldots, \overline{W}_{p(k)-1}, \mathsf{GOright}, \mathsf{Start}_1\}\\ \Gamma_{M,a} \;&:=\; \{\mathsf{Accept}\}\\ \mathsf{Op}_{M,a} \;&:=\; \{\mathsf{start}_1, \mathsf{start}_2, \mathsf{accept}\} \cup \bigcup_{a \in \Sigma} \mathsf{aux}_a\\ &\qquad \bigcup_{\delta(q,a)=(q',b,R)} \{\mathsf{right}_{q,a,q',b}\} \cup \bigcup_{\delta(q,a)=(q',b,L)} \{\mathsf{left}_{q,a,q',b}\}\end{aligned}$$

For all operators from $\mathsf{Op}_{M,a}$, sets of occlusions are empty, and we will write them in the form $(\mathsf{pre}, \mathsf{post})$. The operators start_1 and start_2 are designed in such a way that their pre-conditions ensure that $\mathsf{start}_1, \mathsf{start}_2$ is a prefix of every plan that solves $\Pi_{M,a}$. This is enforced by concepts Start_1 and Start_2 in their pre-conditions. Post-conditions of start_1 ensure that $\overline{H}_i$ is equivalent to $\neg H_i$ and $\overline{W}_i$ is equivalent to $\neg W_i$, for $i < p(k)$. Post-conditions of start_2 ensure that, after its execution:

- the machine is in the initial state q_0, and only in that state.
- if the watched cell position is m, then the contents of the cell is a_m, for $m \leq k$, and the contents is blank, if $m > k$.

Note that world states obtained after executing $\mathsf{start}_1, \mathsf{start}_2$ in $\mathcal{A}_{M,a}$ correspond to the initial state from Rintanen's [Rin04] reduction.

$$
\begin{aligned}
\mathsf{start}_1 \;:=\; & (\{\mathsf{Start}_1\}, \{\top/\neg H_i, W_i/\neg\overline{W}_i \mid i < p(k)\} \cup \{\neg\mathsf{Start}_1, \mathsf{Start}_2\}) \\
\mathsf{start}_2 \;:=\; & (\{\mathsf{Start}_2\}, \{Q_0, \neg Q_1, \ldots, \neg Q_n, \neg\mathsf{Start}_2 \\
& \qquad \neg\mathsf{OK}_{a_1}, \ldots, \neg\mathsf{OK}_{a_l}, \neg\mathsf{OK}_{\mathsf{blank}}, \\
& \qquad \mathsf{Watched}_{=0}/\mathsf{C}_{a_{i_0}}, \ldots, \mathsf{Watched}_{=k}/\mathsf{C}_{a_{i_k}}\} \cup \\
& \qquad\qquad \{X/\mathsf{C}_{\mathsf{blank}} \mid X \in \mathsf{Watched}_{>k}\})
\end{aligned}
$$

The auxiliary operators aux_a, for $a \in \Sigma$ are supposed to precede "transition" operators $\mathsf{right}_{q,a,q',b}$ and $\mathsf{left}_{q,a,q',b}$. Truth values of Aux and OK_a ensure that aux_a and $\mathsf{right}_{q,a,q',b}$ (or $\mathsf{left}_{q,a,q',b}$) are executed alternatingly. The purpose of aux_a is to evaluate concepts HeqW and OK_a, which appear in the pre-conditions and the premises of post-conditions of the transition operators. It is ensured that

$$
\mathsf{HeqW} \equiv \neg\big((H_0 \sqcap \overline{W}_0) \sqcup (\overline{H}_0 \sqcap W_0) \sqcup \cdots \sqcup (\overline{H}_{p(k)-1} \sqcap W_{p(k)-1})\big)
$$

i.e., HeqW holds if and only if the head position and the watched cell position coincide. Moreover, it is achieved that $\mathsf{OK}_a \equiv \mathsf{HeqW} \rightarrow \mathsf{C}_a$, i.e., OK_a is true iff the following holds: if the head points at the watched cell, then the content of the cell is a. Finally, $\mathsf{GOright}$ is set to false if the head is at the rightmost tape cell, and similar for GOleft.

$$
\begin{aligned}
\mathsf{aux}_a \;:=\; & (\{\mathsf{Aux}\}, \{\top/\neg\mathsf{Aux},\ \mathsf{C}_a/\mathsf{OK}_a, \\
& \qquad H_0 \sqcap \overline{W}_0/\{\mathsf{OK}_a, \neg\mathsf{HeqW}\}, \\
& \qquad \overline{H}_0 \sqcap W_0/\{\mathsf{OK}_a, \neg\mathsf{HeqW}\}, \\
& \qquad \ldots \\
& \qquad H_{p(k)-1} \sqcap \overline{W}_{p(k)-1}/\{\mathsf{OK}_a, \neg\mathsf{HeqW}\}, \\
& \qquad \overline{H}_{p(k)-1} \sqcap W_{p(k)-1}/\{\mathsf{OK}_a, \neg\mathsf{HeqW}\}, \\
& \qquad H_0 \sqcap H_1 \sqcap \cdots \sqcap H_{p(k)-1}/\neg\mathsf{GOright}, \\
& \qquad \overline{H}_0 \sqcap \overline{H}_1 \sqcap \cdots \sqcap \overline{H}_{p(k)-1}/\neg\mathsf{GOleft}\}),
\end{aligned}
$$

The transition operators and the accepting operator are defined as follows:[8]

$$\begin{aligned}
\mathsf{right}_{q,a,q',b} &:= (\{Q, \mathsf{OK}_a, \mathsf{GOright}\}, \{\neg Q,\ Q',\ \mathsf{HeqW}/\{\neg \mathsf{C}_a, \mathsf{C}_b\}, \\
&\qquad\qquad \mathsf{GOleft},\ \neg \mathsf{OK}_a,\ \mathsf{Aux},\ \mathsf{HeqW}\} \cup \mathsf{Head}^{++}) \\
\mathsf{left}_{q,a,q',b} &:= (\{Q, \mathsf{OK}_a, \mathsf{GOleft}\}, \{\neg Q,\ Q',\ \mathsf{HeqW}/\{\neg \mathsf{C}_a, \mathsf{C}_b\}, \\
&\qquad\qquad \mathsf{GOright},\ \neg \mathsf{OK}_a,\ \mathsf{Aux},\ \mathsf{HeqW}\} \cup \mathsf{Head}^{--}) \\
\mathsf{accept} &:= (\emptyset, \{Q_f/\mathsf{Accept} \mid q_f \in F\})
\end{aligned}$$

A correct simulation of the Turing Machine is obtained assuming that when operator $\mathsf{right}_{q,a,q',b}$ (or $\mathsf{left}_{q,a,q',b}$) is executed, the current tape symbol is a. Post-conditions $\mathsf{HeqW}/\{\neg \mathsf{C}_a, \mathsf{C}_b\}$ describe that the new symbol in the watched tape cell is b and not a anymore. It is not difficult to see that the Turing machine M accepts the input a iff there is a solution to the planning task $\Pi_{M,a} = (\emptyset, \emptyset, \mathsf{Op}_{M,a}, \mathcal{A}_{M,a}, \Gamma_{M,a})$. As a consequence we obtain the following result.

Theorem 6.4.3. *The conditional plan existence problem in $\mathcal{EL}$ is* ExpSpace*-hard, even if the TBox in the planning task is empty.*

Our adaption of the reduction from [Rin04] shows that the ExpSpace-hardness result holds for propositional logic even if we disallow disjunction and negation in operator pre-conditions and premises of post-conditions.

We conjecture that conditional PLANEX in $\mathcal{EL}$ is ExpSpace-complete. To show the matching upper complexity bound, one would have to modify the filtration method from Section 4.1.1 in such a way that the size of counter-models for projection is bounded by the size of the initial ABox and the sizes of concept assertions appearing in pre- and post-conditions of all ground instances of the operators.

[8] As in the previous section, we assume that $a \neq b$ in the definitions of operators $\mathsf{right}_{q,a,q',b}$ and $\mathsf{left}_{q,a,q',b}$. If $a = b$, $\mathsf{HeqW}/\{\neg \mathsf{C}_a, \mathsf{C}_b\}$ should be omitted from their post-conditions.

Chapter 7

Description Logics with Concrete Domains

This technical chapter is different from the previous ones in the sense that it is dedicated to decidable extensions of description logics. Its goal is to advance the knowledge about decidable DLs that include the expressive means *concrete domains* and *general TBoxes*. As already discussed in the introduction, the purpose of concrete domains is to enable the definition of concepts with reference to concrete qualities of their instances such as the weight, age, duration, and spatial extension. General TBoxes play an important role in modern DLs as they allow to represent background knowledge of application domains by stating via inclusions $C \sqsubseteq D$ that the extension of a concept C is included in the extension of a concept D.

Our contribution is two-fold: first, instead of focusing on particular concrete domains as in previous work [Lut04a, Lut02a], we identify a *general* property of concrete domains, called ω-admissibility, that is sufficient for proving decidability of DLs equipped with concrete domains and general TBoxes. For defining ω-admissibility, we concentrate on a particular kind of concrete domains: *constraint systems*. Constraint systems and ω-admissibility are introduced in Section 7.1. Roughly, a constraint system is a concrete domain that only has binary predicates, which are interpreted as jointly exhaustive and pairwise disjoint (JEPD) relations. We exhibit two example constraint systems that are ω-admissible: a temporal one based on the real line and the Allen relations [All83], and a spatial one based on the real plane and the RCC8 relations [EF91, RCC92]. In Section 7.2, we introduce the description logic $\mathcal{ALC}(\mathcal{C})$ that incorporates the constraint system $\mathcal{C}$ into the basic DL $\mathcal{ALC}$.

Second, in Section 7.3 we develop a *tableau algorithm* for $\mathcal{ALC}(\mathcal{C})$ with general TBoxes. This algorithm is used to establish a general decidability result for the concept satisfiability and subsumption problem in $\mathcal{ALC}$ equipped with general TBoxes and any ω-admissible concrete domain $\mathcal{C}$. Soundness, completeness and termination of the algorithm are proved in Section 7.4. As state-of-the-art DL reasoners are based on tableau algorithms similar to the one described in this chapter, we view our algorithm as a first step towards an efficient implementation of description logics with (ω-admissible) concrete domains and general TBoxes. In contrast to existing tableau algorithms [HMW01, HS01], we allow the concrete domain constructor to have feature chains of length greater than one. In particular, in Section 7.5 we identify an expressive fragment of our logic that should be easily integrated into existing DL reasoners.

7.1 Constraint Systems

We introduce a general notion of *constraint system* that is intended to capture standard constraint systems based on a set of jointly-exhaustive and pairwise-disjoint (JEPD) binary relations. Examples for such systems include spatial constraint networks based on the RCC8 relations [EF91, Ben97, RN99] or on cardinal direction relations [Fra96], and temporal constraint networks based on Allen's relations of time intervals [All83, VKvB90, NB95] or on relations between time points [VK86, VKvB90].

Definition 7.1.1 (Rel-network). Let Var be a countably infinite set of variables and Rel a finite set of binary relation symbols. A Rel-*constraint* is an expression $(x\ r\ y)$ with $x, y \in \mathsf{Var}$ and $r \in \mathsf{Rel}$. A Rel-*network* is a (finite or infinite) set of Rel-constraints. For N a Rel-network, we use V_N to denote the variables used in N. We say that N is *complete* if, for all $x, y \in V_N$, there is exactly one constraint $(x\ r\ y) \in N$. △

We define the semantics of Rel-network by using complete Rel-networks as models. Intuitively, the nodes in these complete networks should be viewed as concrete values rather than as variables. Equivalently to our network-based semantics, we could proceed as in constraint satisfaction problems, associate each variable with a set of values, and view relations as constraints on these values, see e.g. [RN95].

Definition 7.1.2 (Model, Constraint System). Let N be a Rel-network and N' a complete Rel-networks. We say that N' is a *model* of N if there is a mapping $\tau : V_N \to V_{N'}$ such that $(x\ r\ y) \in N$ implies $(\tau(x)\ r\ \tau(y)) \in N'$.

A *constraint system* $\mathcal{C} = \langle \mathsf{Rel}, \mathfrak{M} \rangle$ consists of a finite set of binary relation symbols Rel and a set $\mathfrak{M}$ of complete Rel-networks (the *models* of $\mathcal{C}$). A Rel-network N is *satisfiable* in $\mathcal{C}$ if $\mathfrak{M}$ contains a model of N. △

To emphasize the different role of variables in Rel-networks and in models, we denote variables in the former with $x, y, \ldots$ and in the latter with v, v', etc. Note that Rel-networks used as models have to be complete, which corresponds to the relations in Rel to be jointly exhaustive and mutually exclusive.

Equivalently to our network-based semantics, we could proceed as in constraint satisfaction problems, associate each variable with a set of values, and view relations as constraints on these values, see e.g. [RN95].

In the following two subsections, we introduce two example constraint systems: one for spatial reasoning based on the RCC8 topological relations in the real plane, and one for temporal reasoning based on the Allen relations in the real line.

7.1.1 RCC8

The RCC8 relations, which are illustrated in Figure 7.1, are intended to describe the relation between regions in topological spaces [RCC92]. In this chapter, we will use the standard topology of the real plane which is one of the most appropriate topologies for spatial reasoning. Let

$$\mathsf{RCC8} = \{\mathsf{eq}, \mathsf{dc}, \mathsf{ec}, \mathsf{po}, \mathsf{tpp}, \mathsf{ntpp}, \mathsf{tppi}, \mathsf{ntppi}\}$$

denote the RCC8 relations. Recall that a topological space is a pair $\mathfrak{T} = (U, \mathbb{I})$, where U is a

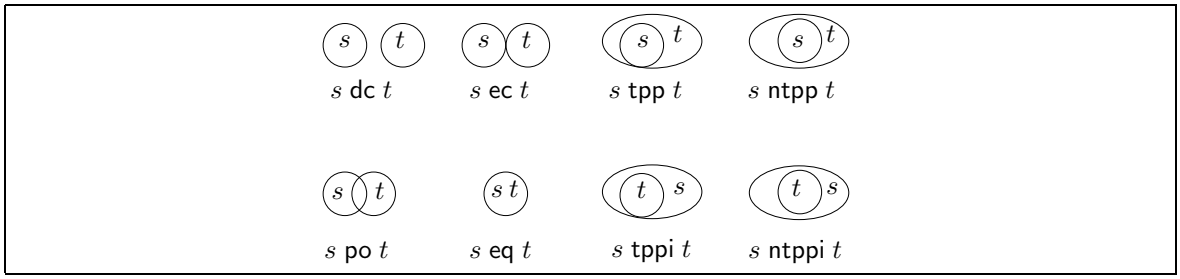

Figure 7.1: The eight RCC8 relations.

set and $\mathbb{I}$ is an *interior operator* on U, i.e., for all $s, t \subseteq U$, we have

$$\begin{array}{rclcrcl} \mathbb{I}(U) & = & U & & \mathbb{I}(s) & \subseteq & s \\ \mathbb{I}(s) \cap \mathbb{I}(t) & = & \mathbb{I}(s \cap t) & & \mathbb{II}(s) & = & \mathbb{I}(s). \end{array}$$

As usual, the closure operator $\mathbb{C}$ is defined as $\mathbb{C}(s) = \overline{\mathbb{I}(\overline{s})}$, where $\overline{t} = U \setminus t$, for $t \subseteq U$. As the *regions* of a topological space $\mathfrak{T} = (U, \mathbb{I})$, we use the set of non-empty, regular closed subsets of U, where a subset $s \subseteq U$ is called *regular closed* if $\mathbb{CI}(s) = s$. Given a topological space $\mathfrak{T}$ and a set of regions $U_{\mathfrak{T}}$, we define the extension of the RCC8 relations as the following subsets of $U_{\mathfrak{T}} \times U_{\mathfrak{T}}$:

$$\begin{array}{rcl} (s,t) \in \mathsf{dc}^{\mathfrak{T}} & \text{iff} & s \cap t = \emptyset \\ (s,t) \in \mathsf{ec}^{\mathfrak{T}} & \text{iff} & \mathbb{I}(s) \cap \mathbb{I}(t) = \emptyset \ \wedge \ s \cap t \neq \emptyset \\ (s,t) \in \mathsf{po}^{\mathfrak{T}} & \text{iff} & \mathbb{I}(s) \cap \mathbb{I}(t) \neq \emptyset \ \wedge \ s \setminus t \neq \emptyset \ \wedge t \setminus s \neq \emptyset \\ (s,t) \in \mathsf{eq}^{\mathfrak{T}} & \text{iff} & s = t \\ (s,t) \in \mathsf{tpp}^{\mathfrak{T}} & \text{iff} & s \cap \overline{t} = \emptyset \ \wedge \ s \cap \overline{\mathbb{I}(t)} \neq \emptyset \wedge s \neq t \\ (s,t) \in \mathsf{ntpp}^{\mathfrak{T}} & \text{iff} & s \cap \overline{\mathbb{I}(t)} = \emptyset \wedge s \neq t \\ (s,t) \in \mathsf{tppi}^{\mathfrak{T}} & \text{iff} & (t,s) \in \mathsf{tpp}^{\mathfrak{T}} \\ (s,t) \in \mathsf{ntppi}^{\mathfrak{T}} & \text{iff} & (t,s) \in \mathsf{ntpp}^{\mathfrak{T}}. \end{array}$$

Let $\mathfrak{T}_{\mathbb{R}^2}$ be the standard topology on ^{2} induced by the Euclidean metric, and let $\mathcal{RS}_{\mathbb{R}^2}$ be the set of all non-empty regular closed subsets of $\mathfrak{T}_{\mathbb{R}^2}$. Then we define the constraint system

$$\mathsf{RCC8}_{\mathbb{R}^2} = \langle \mathsf{RCC8}, \mathfrak{M}_{\mathbb{R}^2} \rangle$$

by setting $\mathfrak{M}_{\mathbb{R}^2} := \{N_{\mathbb{R}^2}\}$, where $N_{\mathbb{R}^2}$ is defined by fixing a variable $v_s \in \mathsf{Var}$ for every $s \in \mathcal{RS}_{\mathbb{R}^2}$ and setting

$$N_{\mathbb{R}^2} := \{(v_s \ r \ v_t) \mid r \in \mathsf{RCC8}, \ s,t \in \mathcal{RS}_{\mathbb{R}^2} \text{ and } (s,t) \in r^{\mathfrak{T}_{\mathbb{R}^2}}\}.$$

Note that using only regular closed sets excludes sub-dimensional regions such as points and lines. This is necessary for the RCC8 relations to be jointly exhaustive and pairwise disjoint.

7.1.2 Allen's Relations

In artificial intelligence, constraint systems based on Allen's interval relations are a popular tool for the representation of temporal knowledge [All83]. Let

$$\mathsf{Allen} = \{\mathsf{b}, \mathsf{a}, \mathsf{m}, \mathsf{mi}, \mathsf{o}, \mathsf{oi}, \mathsf{d}, \mathsf{di}, \mathsf{s}, \mathsf{si}, \mathsf{f}, \mathsf{fi}, =\}$$

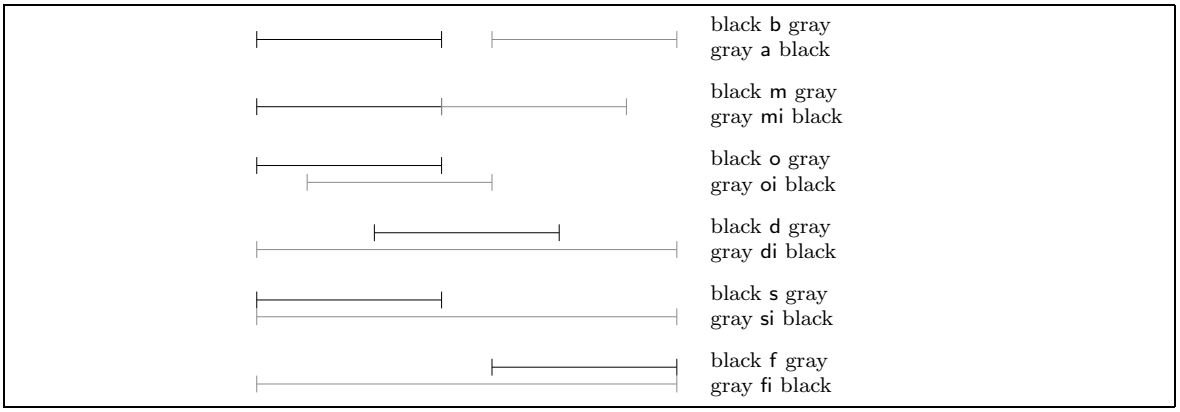

Figure 7.2: The thirteen Allen relations.

denote the thirteen Allen relations. Examples of these relations are given in Figure 7.2. As the flow of time, we use the real numbers with the usual ordering. Let $\mathsf{Int}_{\mathbb{R}}$ denote the set of all closed intervals $[r_1, r_2]$ over with $r_1 < r_2$, i.e., point-intervals are not admitted. The extension $\mathsf{r}^{\mathbb{R}}$ of each Allen relation r is a subset of $\mathsf{Int}_{\mathbb{R}} \times \mathsf{Int}_{\mathbb{R}}$. It is defined in terms of the relationships between endpoints in the obvious way, c.f. Figure 7.2. We define the constraint system

$$\mathsf{Allen}_{\mathbb{R}} = \langle \mathsf{Allen}, \mathfrak{M}_{\mathbb{R}} \rangle$$

by setting $\mathfrak{M}_{\mathbb{R}} := \{N_{\mathbb{R}}\}$, where $N_{\mathbb{R}}$ is defined by fixing a variable $v_i \in \mathsf{Var}$ for every $i \in \mathsf{Int}_{\mathbb{R}}$ and setting

$$N_{\mathbb{R}} := \{(v_i \; r \; v_j) \mid r \in \mathsf{Allen},\ i, j \in \mathsf{Int}_{\mathbb{R}} \text{ and } (i, j) \in r^{\mathbb{R}}\}.$$

We could also define the constraint system $\mathsf{Allen}_{\mathbb{Q}}$ based on the rationals rather than on the reals: this has no impact on the satisfiability of finite and infinite Allen-networks (which are countable by definition). If we use the natural numbers or the integers, this still holds for finite networks, but not for infinite ones: there are infinite Allen-networks that are satisfiable over the reals and rationals, but not over the natural number and integers.

7.1.3 Properties of Constraint Systems

We will use constraint systems as a concrete domain for description logics. To obtain sound and complete reasoning procedures for DLs with such concrete domains, we require that constraint systems satisfy certain properties. First, we need to ensure that satisfiable networks (satisfying some additional conditions) can be "patched" together to a joint network that is also satisfiable. This is ensured by the patchwork property.

Definition 7.1.3 (Patchwork Property). Let $\mathcal{C} = \langle \mathsf{Rel}, \mathfrak{M} \rangle$ be a constraint system, and let N, M be finite complete Rel-networks such that, for the intersection parts

$$\begin{aligned} I_{N,M} &:= \{(x \; r \; y) \mid x, y \in V_N \cap V_M \text{ and } (x \; r \; y) \in N\} \\ I_{M,N} &:= \{(x \; r \; y) \mid x, y \in V_N \cap V_M \text{ and } (x \; r \; y) \in M\} \end{aligned}$$

we have $I_{N,M} = I_{M,N}$. Then the *composition* of N and M is defined as $N \cup M$. We say that $\mathcal{C}$ has the *patchwork property* if the following holds: if N and M are satisfiable then $N \cup M$ is satisfiable. $\triangle$

The patchwork property is similar to the property of constraint networks formulated by Balbiani in [BC02], where constraint networks are combined with linear temporal logic.

For using constraint systems with the DL tableau algorithm presented in this chapter, we must be sure that, even if we patch together an infinite number of satisfiable networks, the resulting (infinite) network is still satisfiable. This is guaranteed by the compactness property.

Definition 7.1.4 (Compactness). Let $\mathcal{C} = \langle \mathsf{Rel}, \mathfrak{M} \rangle$ be a constraint system. If N is a Rel-network and $V \subseteq V_N$, we write $N|_V$ to denote the network $\{(x\, r\, y) \in N \mid x, y \in V\} \subseteq N$. Then $\mathcal{C}$ has the *compactness property* if the following holds: a Rel-network N with V_N infinite is satisfiable in $\mathcal{C}$ if and only if, for every finite $V \subseteq V_N$, the network $N|_V$ is satisfiable in $\mathcal{C}$. △

Finally, our tableau algorithm has to check satisfiability of certain $\mathcal{C}$-networks. Thus, we have to assume that $\mathcal{C}$-satisfiability is decidable. The properties of constraint systems we require are summarized in the following definition.

Definition 7.1.5 (ω-admissible). Let $\mathcal{C} = (\mathsf{Rel}, \mathfrak{M})$ be a constraint system. We say that $\mathcal{C}$ is *ω-admissible* iff the following holds:

1. satisfiability of finite $\mathcal{C}$-networks is decidable;
2. $\mathcal{C}$ has the patchwork property (c.f. Definition 7.1.3);
3. $\mathcal{C}$ has the compactness property (c.f. Definition 7.1.4).

△

In the sections 7.1.3 and 7.1.3, we prove that $\mathsf{RCC8}_{\mathbb{R}^2}$ and $\mathsf{Allen}_{\mathbb{R}}$ satisfy the patchwork property and the compactness property. Moreover, satisfiability of finite networks is decidable in both systems: it is tractable for $\mathsf{RCC8}_{\mathbb{R}^2}$ [RN99] and NP-complete for $\mathsf{Allen}_{\mathbb{R}}$ [VKvB90]. Thus, $\mathsf{RCC8}_{\mathbb{R}^2}$ and $\mathsf{Allen}_{\mathbb{R}}$ are ω-admissible.

Properties of RCC8

We show that $\mathsf{RCC8}_{\mathbb{R}^2}$ has the patchwork property and the compactness property. To this end, we consider a different variant of the constraint system $\mathsf{RCC8}_{\mathbb{R}^2}$. To introduce it, we need a couple of definitions. A *fork* F is a structure $\langle W_F, R_F, \pi_F \rangle$, where

- W_F is a set $\{b_F, r_F, \ell_F\}$ of cardinality three,
- R_F is the reflexive closure of $\{(b_F, r_F), (b_F, \ell_F)\}$, and
- $\pi_F : \mathsf{Var} \to 2^{W_F}$ is a valuation such that, for each $x \in \mathsf{Var}$, we have

$$b_F \in \pi_F(x) \text{ iff } \ell_F \in \pi_F(x) \text{ or } r_F \in \pi_F(x).$$

A *fork model* M is a (finite or infinite) disjoint union of forks $F_0, F_1, \ldots$. We write W_M for $\bigcup_{i\geq 0} W_{F_i}$, R_M for $\bigcup_{i\geq 0} R_{F_i}$, and $\pi_M(x)$ for $\bigcup_{i\geq 0} \pi_{F_i}(x)$. We may interpret the $\mathsf{RCC8}$ relations on a fork model M by associating a topological space $\mathfrak{T}_M$ with M: define an interior operator $\mathbb{I}_M$ by setting, for all $X \subseteq W_M$,

$$\mathbb{I}_M X := \{x \in \bigcup_{i\geq 0} W_M \mid \forall y\, (xR_M y \to y \in X)\}$$

(and thus $\mathbb{C}_M X = \{x \in W_M \mid \exists y\,(xR_M y \wedge y \in X)\}$). Let $\mathcal{RS}_M$ denote the set of non-empty regular closed subsets of W_M. We now define the constraint system

$$\mathsf{RCC8}_{\mathsf{Fork}} := \langle \mathsf{RCC8}, \mathfrak{M}_{\mathsf{Fork}} \rangle$$

by setting $\mathfrak{M}_{\mathsf{Fork}} := \{N_M \mid M \text{ a fork model}\}$, where N_M is defined by fixing a variable $v_X \in \mathsf{Var}$ for every $X \in \mathcal{RS}_M$ and setting

$$N_M := \{(v_X\ r\ v_{X'}) \mid r \in \mathsf{RCC8},\ X, X' \in \mathcal{RS}_M, \text{ and } (X, X') \in r^{\mathfrak{T}_M}\}.$$

It was shown by Renz and Nebel that satisfiability of finite constraint networks in $\mathsf{RCC8}_{\mathbb{R}^2}$ coincides with satisfiability in $\mathsf{RCC8}_{\mathsf{Fork}}$ [RN99]. This was extended to infinite networks in [LW06]:

Theorem 7.1.6. *An* $\mathsf{RCC8}$*-network is satisfiable in* $\mathsf{RCC8}_{\mathbb{R}^2}$ *iff it is satisfiable in* $\mathsf{RCC8}_{\mathsf{Fork}}$.

Due to Theorem 7.1.6, it suffices to prove the patchwork property and compactness for $\mathsf{RCC8}_{\mathsf{Fork}}$. This is what we do in the following. Our proof of the patchwork property is based on a result of Gabbay et al. [GKWZ03]. To formulate it, we need to introduce the standard translation [Ben97, RN99] of $\mathsf{RCC8}$-networks to the modal logic $\mathsf{S4}_u$, i.e., Lewis' (uni-modal) $\mathsf{S4}$ enriched with the universal modality. We refrain from giving the syntax and semantics of $\mathsf{S4}_u$ and refer, e.g., to [GKWZ03] for more information. Note, however, that formulas of $\mathsf{S4}_u$ can be interpreted in fork models.

We use $\mathbb{I}$ to denote the $\mathsf{S4}$ box operator, $\Box_u$ to denote the universal box, and write $\Diamond_u\varphi$ for $\neg\Box_u\neg\varphi$ as usual. Given an $\mathsf{RCC8}$-constraint $(x\,r\,y)$, we define a corresponding $\mathsf{S4}_u$-formula $(x\,r\,y)^{\bowtie}$ as follows:

$$\begin{aligned}
(x\,\mathsf{eq}\,y)^{\bowtie} &= \Box_u(x \leftrightarrow y)\\
(x\,\mathsf{dc}\,y)^{\bowtie} &= \Box_u(\neg x \vee \neg y)\\
(x\,\mathsf{ec}\,y)^{\bowtie} &= \Diamond_u(x \wedge y) \wedge \Box_u(\neg\mathbb{I}x \vee \neg\mathbb{I}y)\\
(x\,\mathsf{po}\,y)^{\bowtie} &= \Diamond_u(\mathbb{I}x \wedge \mathbb{I}y) \wedge \Diamond_u(x \wedge \neg y) \wedge \Diamond_u(\neg x \wedge y)\\
(x\,\mathsf{tpp}\,y)^{\bowtie} &= \Box_u(x \rightarrow y) \wedge \Diamond_u(x \wedge \neg\mathbb{I}y) \wedge \Diamond_u(\neg x \wedge y)\\
(x\,\mathsf{ntpp}\,y)^{\bowtie} &= \Box_u(x \rightarrow \mathbb{I}y) \wedge \Diamond_u(\neg x \wedge y)
\end{aligned}$$

Constraints $(x\,\mathsf{tppi}\,y)$ and $(x\,\mathsf{ntppi}\,y)$ are converted into $(y\,\mathsf{tpp}\,x)$ and $(y\,\mathsf{ntpp}\,x)$, respectively, and then translated as above. Observe that variables of the network are translated into propositional variables of $\mathsf{S4}_u$. For every $\mathsf{RCC8}$-constraint network N, we define a corresponding set of $\mathsf{S4}_u$ formulas $N^{\bowtie}$ by setting $N^{\bowtie} := \{(x\,\mathsf{r}\,y)^{\bowtie} \mid (x\,\mathsf{r}\,y) \in N\}$. The most important property of the translation $\cdot^{\bowtie}$ is the following, as established in [RN99]:

Theorem 7.1.7. *Let* N *be a finite* $\mathsf{RCC8}$*-network. Then* N *is satisfiable in* $\mathsf{RCC8}_{\mathsf{Fork}}$ *iff the set of* $\mathsf{S4}_u$ *formulas* $N^{\bowtie}$ *is satisfiable in a fork model.*

For a constraint $(x\,r\,y)$, we use $(x\,r\,y)^{\forall}$ to denote the formula obtained from $(x\,r\,y)^{\bowtie}$ by dropping all conjuncts starting with $\Diamond_u$ (assuming that $(x\,r\,y)^{\forall}$ is the constant true if all conjuncts are dropped), and likewise for $(x\,r\,y)^{\exists}$ and $\Box_u$. For networks, the notions $N^{\forall}$ and $N^{\exists}$ are defined in the obvious way.

For what follows, it will be important to identify a particular class of forks induced by a constraint network. Intuitively, this class of forks can be viewed as a canonical model for the inducing network, if this network is satisfiable. For N an $\mathsf{RCC8}$-network, we set

$$\mathsf{Fork}_N := \{F \text{ a fork} \mid F \text{ satisfies } N^{\forall}\}.$$

We say that two forks F and F' are *V-equivalent*, for V a set of variables, when for all $x \in V$, we have that (i) $r_F \in \pi_F(x)$ iff $r_{F'} \in \pi_{F'}(x)$ and (ii) $\ell_F \in \pi_F(x)$ iff $\ell_{F'} \in \pi_{F'}(x)$ (recall that by definition of forks, the value of b_F is determined by those of r_F and ℓ_F). The following theorem forms the basis for our proof that $\mathsf{RCC8}_{\mathsf{Fork}}$ has the patchwork property. It is a reformulation of Theorem 16.17 in [GKWZ03]. For $r \in \mathsf{RCC8}$, we use $\mathsf{Inv}(r)$ to denote the inverse of the relation r, e.g. $\mathsf{Inv}(\mathsf{po}) = \mathsf{po}$.

Theorem 7.1.8 (Gabbay et al.). *Let N be a finite, complete, satisfiable* RCC8*-network, $x \notin V_N$, and*

$$N' = N \cup \{(x\ r_y\ y),\ (y\ \mathsf{Inv}(r_y)\ x) \mid y \in V_N\}$$

for some family of relations $(r_y)_{y \in V_N}$, such that N' is satisfiable. Then, for each $F \in \mathsf{Fork}_N$, there exists an $F' \in \mathsf{Fork}_{N'}$ such that F and F' are V_N-equivalent.

The following corollary is easily proved by induction on the cardinality of $V_M \setminus V_N$.

Corollary 7.1.9. *Let N and M be two finite complete satisfiable* RCC8*-networks, such that $N \subseteq M$. Then, for each $F \in \mathsf{Fork}_N$, there exists an $F' \in \mathsf{Fork}_M$ such that F and F' are V_N-equivalent.*

We may now establish the patchwork property.

Lemma 7.1.10. $\mathsf{RCC8}_{\mathbb{R}^2}$ *has the patchwork property.*

Proof. By Theorem 7.1.6, it suffices to show that $\mathsf{RCC8}_{\mathsf{Fork}}$ has the patchwork property. Let N and M be finite and complete RCC8-networks that are satisfiable in $\mathsf{RCC8}_{\mathsf{Fork}}$ and whose intersection parts $I_{N,M}$ and $I_{M,N}$ (as defined in Definition 7.1.3) are identical. We have to prove that $N \cup M$ is also satisfiable in $\mathsf{RCC8}_{\mathsf{Fork}}$. By Theorem 7.1.7, it suffices to show that $(N \cup M)^{\bowtie}$ is satisfiable in a fork model. We show that a satisfying model is provided by $\mathcal{F}_{N,M} := \mathsf{Fork}_N \cap \mathsf{Fork}_M$. We distinguish between the universal and existential part of $(N \cup M)^{\bowtie}$.

(i) $\mathcal{F}_{N,M}$ satisfies $(N \cup M)^{\forall} = N^{\forall} \cup M^{\forall}$. It suffices to show that every $F \in \mathcal{F}_{N,M}$ satisfies $N^{\forall}$ and $M^{\forall}$. The former is an immediate consequence of $\mathcal{F}_{N,M} \subseteq \mathsf{Fork}_N$ and the definition of Fork_N. The argument for the latter is analogous.

(ii) $\mathcal{F}_{N,M}$ satisfies $(N \cup M)^{\exists} = N^{\exists} \cup M^{\exists}$. To show this, it is sufficient to show that (a) for every $F \in \mathsf{Fork}_N$, there is an $F' \in \mathcal{F}_{M,N}$ which is V_N-equivalent to F and (b) for every $F \in \mathsf{Fork}_M$, there is an $F' \in \mathcal{F}_{M,N}$ which is V_M-equivalent to F. Then, since Fork_N satisfies $N^{\bowtie}$, all $\Diamond_u \varphi \in N^{\exists}$ will be satisfied by $\mathcal{F}_{M,N}$, and likewise for M. We only show (a) as (b) is analogous. For brevity, let I denote $I_{N,M}$ $(=I_{M,N})$. Take an $F \in \mathsf{Fork}_N$. Clearly, since $I \subseteq N$, we have that $F \in \mathsf{Fork}_I$. Moreover, I is finite, complete, and satisfiable since N and M are. Thus, by Corollary 7.1.9 there exists an $F' \in \mathsf{Fork}_M$ that is V_I-equivalent to F. Now define a fork $F'' = (W_{F''}, R_{F''}, \pi_{F''})$ as follows:

$$\pi_{F''}(x) := \begin{cases} \pi_F(x) & \text{if } x \in V_N \\ \pi_{F'}(x) & \text{otherwise} \end{cases}$$

It is not difficult to see that F'' is V_N-equivalent to F and V_M-equivalent to F'. Since V_N is clearly closed under V_N-equivalence (and likewise for V_M), this yields $F'' \in \mathsf{Fork}_N \cap \mathsf{Fork}_M = \mathcal{F}_{M,N}$.

❑

It remains to treat compactness.

Lemma 7.1.11. $\mathsf{RCC8}_{\mathbb{R}^2}$ *has the compactness property.*

Proof. It is easily seen that satisfiability of an infinite $\mathsf{RCC8}$-network N implies satisfiability of $N|_V$, for every finite $V \subseteq V_N$. To show the converse, we give a satisfiability preserving translation of $\mathsf{RCC8}$-networks N to a set $\Gamma(N)$ of first-order sentences in the following signature: a binary predicate R representing the partial order in fork frames and unary predicates $(P_x)_{x\in\mathsf{Var}}$ for variables. We then use compactness of first-order logic to deduce that $\mathsf{RCC8}_{\mathsf{Fork}}$ has the compactness property. By Theorem 7.1.6, it follows that $\mathsf{RCC8}_{\mathbb{R}^2}$ has the compactness property. Let N be a (possibly infinite) $\mathsf{RCC8}$-network. The set of first-order sentences $\Gamma(N)$ consists of the following:

- a formula stating that R is a disjoint union of forks:

$$\begin{aligned}\forall w \exists x, y, z (&xRx \wedge yRy \wedge zRz \wedge xRy \wedge xRz \wedge \\ &\forall u(xRu \rightarrow (u = x \vee u = y \vee u = z)) \wedge \\ &\forall u(yRu \rightarrow u = y) \wedge \\ &\forall u(zRu \rightarrow u = z) \wedge \\ &\forall u(uRx \rightarrow u = x) \wedge \\ &\forall u(uRy \rightarrow (u = x \vee u = y)) \wedge \\ &\forall u(uRz \rightarrow (u = x \vee u = z)) \wedge \\ &x \neq y \wedge x \neq z \wedge y \neq z \wedge \\ &(w = x \vee w = y \vee w = z))\end{aligned}$$

- to ensure the restriction that is imposed on valuations of fork models, for each unary predicate P, we add the following formula:

$$\forall x(\mathsf{root}(x) \rightarrow (P(x) \leftrightarrow \exists y(xRy \wedge x \neq y \wedge P(y))))$$

  where $\mathsf{root}(x) := \forall y(yRx \rightarrow x = y)$ expresses that x is the root of a fork.

- the translation of each constraint in N. We only treat the case $(x\ \mathsf{ec}\ y)$ explicitly:

$$\exists z(P_x(z) \wedge P_y(z)) \wedge \neg\exists z(\mathsf{Int}_x(z) \wedge \mathsf{Int}_y(z))$$

  where $\mathsf{Int}_x(z) := P_x(z) \wedge \forall z'(zRz' \rightarrow P_y(z'))$ describes the interior points of P_x (to see this, consider the way in which fork frames induce topologies). The other cases are easily obtained by referring to the semantics of the $\mathsf{RCC8}$ relations.

Now let N be an infinite $\mathsf{RCC8}$-network such that $N|_V$ is satisfiable in $\mathsf{RCC8}_{\mathsf{Fork}}$ for every finite $V \subseteq V_N$. We have to show that N is satisfiable. Let Ψ be a finite subset of $\Gamma(N)$, and let N' be the fragment of N that contains precisely those constraints whose translation is in Ψ. By Theorem 7.1.7, N' has a model that is the topology of a fork model M. Define a first-order structure $\mathfrak{M}$ with domain W_M by setting $R^{\mathfrak{M}} := R_M$ and $P_x^{\mathfrak{M}} := \pi_M(x)$ for all $x \in V$. It is readily checked that $\mathfrak{M}$ is a model of Ψ. Thus, every finite subset of $\Gamma(N)$ is satisfiable and compactness of first-oder logic implies that $\Gamma(N)$ is satisfiable. Take a model $\mathfrak{N}$ of $\Gamma(N)$ with domain $\mathfrak{A}$. Clearly, $M' = (\mathfrak{A}, R^{\mathfrak{N}}, \{x \mapsto P_x^{\mathfrak{N}}\})$ is a fork model. It is readily checked that the topology $\mathfrak{T}_{M'}$ is a model of N. ❏

Properties of Allen

We prove that the constraint system $\mathsf{Allen}_{\mathbb{R}}$ has both the patchwork property and the compactness property.

Lemma 7.1.12. $\mathsf{Allen}_{\mathbb{R}}$ *has the patchwork property.*

Proof. Let N and M be finite complete Allen-networks that are satisfiable in $\mathsf{Allen}_{\mathbb{R}}$ and whose intersection parts $I_{N,M}$ and $I_{M,N}$ (defined as in Definition 7.1.3) are identical. We have to prove that $N \cup M$ is also satisfiable. Satisfiability of N means that there exists a mapping $\tau_N : V_N \to \mathsf{Int}_{\mathbb{R}}$ such that $(x \,\mathsf{r}\, y) \in N$ implies $(\tau_N(x), \tau_N(y)) \in \mathsf{r}^{\mathbb{R}}$, and an analogous mapping τ_M for M. Define

$$\begin{aligned} S_N \;:=\; & \{(x, L, r) \mid x \in V_{I_{N,M}} \text{ and } \tau_N(v) = [r, r'] \text{ for some } r' \in \quad\} \cup \\ & \{(x, R, r) \mid x \in V_{I_{N,M}} \text{ and } \tau_N(v) = [r', r] \text{ for some } r' \in \quad\} \end{aligned}$$

Now arrange the elements of S_N in a sequence $(v_0, D_0, r_0), \ldots, (v_k, D_k, r_k)$ such that $i < j$ implies $r_i \leq r_j$. Define a corresponding sequence $(v_0, D_0, r'_0), \ldots, (v_k, D_k, r'_k)$ for M by setting, for $i \leq k$,

$$r'_i := \begin{cases} r & \text{if } D_i = L \text{ and } \tau_M(x_i) = (r, r') \text{ for some } r' \in \\ r & \text{if } D_i = R \text{ and } \tau_M(x_i) = (r', r) \text{ for some } r' \in \quad . \end{cases}$$

Since $I_{N,M} = I_{M,N}$, we have that $i < j$ implies $r'_i \leq r'_j$. Fix, for each $i < k$, a bijection π_i from the interval $[r'_i, r'_{i+1})$ to the interval $[r_i, r_{i+1})$ that is an isomorphism w.r.t. "<". Moreover, fix additional isomorphisms $\pi^* : (-\infty, r'_0)$ to $(-\infty, r_0)$ and $\pi^\dagger : [r'_k, \infty)$ to $[r_k, \infty)$. For $r \in \quad$, set

$$\pi(r) := \begin{cases} \pi^*(r) & \text{if } r < r'_0 \\ \pi_i(r) & \text{if } r_i \leq r < r'_{i+1} \\ \pi^\dagger(r) & \text{if } r \geq r_k \end{cases}$$

Now define a mapping $\tau'_M : V_M \to \mathsf{Int}_{\mathbb{R}}$ by setting $\tau'_M(x) := [\pi(r), \pi(r')]$ if $\tau_M(x) = [r, r']$. It is readily checked that τ_N and τ'_M agree on $V_{I_{N,M}}$, and that $\tau_N \cup \tau'_M$ witnesses satisfaction of $N \cup M$ in $\mathsf{Allen}_{\mathbb{R}}$. ❏

Lemma 7.1.13. $\mathsf{Allen}_{\mathbb{R}}$ *has the compactness property.*

Proof. As in the case of RCC8, it is easily seen that satisfiability of an infinite Allen-network N implies satisfiability of $N|_V$, for every finite $V \subseteq V_N$. To show the converse, we give a satisfiability preserving translation of Allen-networks N to a set $\Gamma(N)$ of first-order sentences in the following signature: a binary predicate $<$ representing the ordering on , and constants $(b_x)_{x \in \mathsf{Var}}$ and $(e_x)_{x \in \mathsf{Var}}$ denoting the begin and end points of intervals. Let N be a (possibly infinite) constraint network. The set of first-order sentences $\Gamma(N)$ consists of the following:

- one sentence for each constraint in N. The translation is easily read off from the definition of the Allen relations. E.g., $(x \,\mathsf{m}\, y)$ translates to $e_x = b_y$;
- for each $x \in V_N$, a sentence ensuring the correct ordering of endpoints: $b_x < e_x$.

It is easily seen that each finite or infinite Allen-network N is satisfiable in $\mathsf{Allen}_{\mathbb{R}}$ iff $\Gamma(N)$ is satisfiable in a structure $(\;, <, P_1^{\mathfrak{M}}, P_2^{\mathfrak{M}}, \ldots)$. Thus, compactness of first-order logic on such structures implies that $\mathsf{Allen}_{\mathbb{R}}$ has the compactness property. ❏

7.2 Description Logic $\mathcal{ALC}(\mathcal{C})$

We introduce the description logic $\mathcal{ALC}(\mathcal{C})$, an extension of the basic DL $\mathcal{ALC}$, that allows to define concepts with reference to the constraint system $\mathcal{C}$. Different incarnations of $\mathcal{ALC}(\mathcal{C})$ are obtained by instantiating it with different constraint systems.

Definition 7.2.1 ($\mathcal{ALC}(\mathcal{C})$ Syntax and Semantics). Let $\mathcal{C} = (\mathsf{Rel}, \mathfrak{M})$ be a constraint system, and let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_{cF}}$ be mutually disjoint and countably infinite sets of *concept names*, *role names*, and *concrete features*. We assume that $\mathsf{N_R}$ is partitioned into two countably infinite subsets $\mathsf{N_{aF}}$ and $\mathsf{N_{rR}}$. The elements of $\mathsf{N_{aF}}$ are called *abstract features* and the elements of $\mathsf{N_{rR}}$ *standard roles*. A *path* of length $k+1$ with $k \geq 0$ is a sequence $R_1 \cdots R_k g$ consisting of roles $R_1, \ldots, R_k \in \mathsf{N_R}$ and a concrete feature $g \in \mathsf{N_{cF}}$. A path $R_1 \cdots R_k g$ with $\{R_1, \ldots, R_k\} \subseteq \mathsf{N_{aF}}$ is called *feature path*. The set of $\mathcal{ALC}(\mathcal{C})$-concepts is the smallest set such that

1. every concept name $A \in \mathsf{N_C}$ is a concept,
2. if C and D are concepts and $R \in \mathsf{N_R}$, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\forall R.C$, and $\exists R.C$ are concepts;
3. if u_1 and u_2 are feature paths and $r_1, \ldots, r_k \in \mathsf{Rel}$, then the following are also concepts:
$$\exists u_1, u_2.(r_1 \vee \cdots \vee r_k) \text{ and } \forall u_1, u_2.(r_1 \vee \cdots \vee r_k);$$
4. if U_1 and U_2 are paths of length at most two and $r_1, \ldots, r_k \in \mathsf{Rel}$, then the following are also concepts:
$$\exists U_1, U_2.(r_1 \vee \cdots \vee r_k) \text{ and } \forall U_1, U_2.(r_1 \vee \cdots \vee r_k);$$

Observe that we restrict the length of paths inside the constraint-based constructor to two only if at least one of the paths contains a standard role.

Interpretations for $\mathcal{ALC}(\mathcal{C})$, relative to those defined in Section 2, have as an additional argument a network $M_\mathcal{I}$ from the set of models $\mathfrak{M}$ of $\mathcal{C}$. Thus, an $\mathcal{ALC}(\mathcal{C})$-*interpretation* $\mathcal{I}$ is a tuple $(\Delta^\mathcal{I}, \cdot^\mathcal{I}, M_\mathcal{I})$, where $\Delta^\mathcal{I}$ is the *interpretation domain*, $\cdot^\mathcal{I}$ is the *interpretation function*, and $M_\mathcal{I} \in \mathfrak{M}$. The interpretation function maps concept and role names as defined in Section 2, and:

- each abstract feature f to a partial function $f^\mathcal{I}$ from $\Delta^\mathcal{I}$ to $\Delta^\mathcal{I}$;
- each concrete feature g to a partial function $g^\mathcal{I}$ from $\Delta^\mathcal{I}$ to the set of variables $V_{M_\mathcal{I}}$ of $M_\mathcal{I}$.

If $\mathbf{r} = r_1 \vee \cdots \vee r_k$, where $r_1, \ldots, r_k \in \mathsf{Rel}$, we write $M_\mathcal{I} \models (x\ \mathbf{r}\ y)$ iff there exists an $i \in \{1, \ldots, k\}$ such that $(x\ r_i\ y) \in M_\mathcal{I}$. The interpretation function is then extended to the concepts of the form $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$ and $\forall R.C$ as defined in Section 2, and additionally:

$$\begin{aligned}
(\exists U_1, U_2.\mathbf{r})^\mathcal{I} &:= \{d \in \Delta^\mathcal{I} \mid \exists v_1 \in U_1^\mathcal{I}(d) \text{ and } v_2 \in U_2^\mathcal{I}(d) \\
&\qquad \text{with } M_\mathcal{I} \models (v_1\ \mathbf{r}\ v_2)\}, \\
(\forall U_1, U_2.\mathbf{r})^\mathcal{I} &:= \{d \in \Delta^\mathcal{I} \mid \forall v_1 \in U_1^\mathcal{I}(d) \text{ and } v_2 \in U_2^\mathcal{I}(d), \\
&\qquad \text{we have } M_\mathcal{I} \models (v_1\ \mathbf{r}\ v_2)\}
\end{aligned}$$

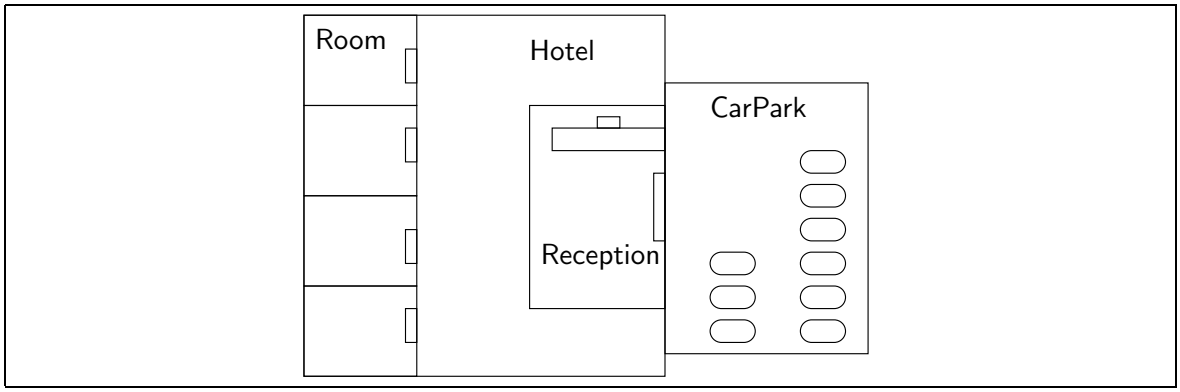

Figure 7.3: An example of a CarFriendlyHotel.

where for a path $U = R_1 \cdots R_k g$ and $d \in \Delta^{\mathcal{I}}$, $U^{\mathcal{I}}(d)$ is defined as

$$\{v \in V_{M_{\mathcal{I}}} \mid \exists e_1, \ldots, e_{k+1} : d = e_1, \\ (e_i, e_{i+1}) \in R_i^{\mathcal{I}} \text{ for } 1 \leq i \leq k, \text{ and } g^{\mathcal{I}}(e_{k+1}) = v\}.$$

The reasoning problems concept satisfiability and subsumption (w.r.t. TBoxes) in $\mathcal{ALC}(\mathcal{C})$ are defined as in Chapter 2, assuming that models are $\mathcal{ALC}(\mathcal{C})$-interpretations. △

Observe that the network $M_{\mathcal{I}}$ in the previous definitionis a model of the constraint system $\mathcal{C}$, whence variables in this network correspond to values in $\mathcal{C}$ and are denoted with v, v' rather than x, y.

Example 7.2.2. *The following example general TBox formulated in $\mathcal{ALC}(\mathcal{C})$ describes some properties of a hotel using the constraint system* $\mathsf{RCC8}_{\mathbb{R}^2}$*, where* has-room *is a role,* has-reception *and* has-carpark *are abstract features (assuming that a hotel has at most a single reception and car park),* loc *is a concrete feature, and all capitalized words are concept names.*

$$\begin{aligned}
\mathsf{Hotel} &\sqsubseteq \forall \mathsf{has\text{-}room}.\mathsf{Room} \sqcap \forall \mathsf{has\text{-}reception}.\mathsf{Reception} \\
&\quad \sqcap \forall \mathsf{has\text{-}carpark}.\mathsf{CarPark} \\
\mathsf{Hotel} &\sqsubseteq \forall(\mathsf{has\text{-}room}\ \mathsf{loc}), (\mathsf{loc}).\mathsf{tpp} \vee \mathsf{ntpp} \\
&\quad \sqcap \forall(\mathsf{has\text{-}room}\ \mathsf{loc}), (\mathsf{has\text{-}room}\ \mathsf{loc}).\mathsf{dc} \vee \mathsf{ec} \vee \mathsf{eq} \\
\mathsf{CarFriendlyHotel} &\doteq \mathsf{Hotel} \sqcap \exists(\mathsf{has\text{-}reception}\ \mathsf{loc}), (\mathsf{loc}).\mathsf{tpp} \\
&\quad \sqcap \exists(\mathsf{has\text{-}carpark}\ \mathsf{loc}), (\mathsf{loc}).\mathsf{ec} \\
&\quad \sqcap \exists(\mathsf{has\text{-}carpark}\ \mathsf{loc}), (\mathsf{has\text{-}reception}\ \mathsf{loc}).\mathsf{ec}
\end{aligned}$$

The first concept inclusion expresses that hotels are related via the three roles to objects of the proper type. The second concept inclusion says that the rooms of a hotel are spatially contained in the hotel, and that rooms do not overlap. Finally, the last concept inclusion describes hotels that are convenient for car owners: they have a car park that is directly next to the reception. This situation is illustrated in Figure 7.3.

7.3 A Tableau Algorithm for $\mathcal{ALC}(\mathcal{C})$

We present a tableau algorithm that decides satisfiability of $\mathcal{ALC}(\mathcal{C})$-concepts w.r.t. general TBoxes. It is well-known that subsumption can be reduced to (un)satisfiability: $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{T}$. This allows us to concentrate on concept satisfiability when devising reasoning procedures. Tableau algorithms are among the most popular decision procedures for description logics since they are amenable to various optimization techniques and often can be efficiently implemented. Therefore, we view the algorithm presented in this chapter as a first step towards practicable reasoning with concrete domains and general TBoxes. On the flipside, algorithms such as the one developed in this section usually do not yield tight upper complexity bounds.

The algorithm developed in the following is independent of the constraint system $\mathcal{C}$. This is achieved by delegating reasoning in $\mathcal{C}$ to an external reasoner that decides satisfiability of $\mathcal{C}$-networks. Throughout this section, we assume $\mathcal{C}$ to be ω-admissible.

7.3.1 Normal Forms

It is convenient to first convert the input concept and general TBox into an appropriate syntactic form. More precisely, we convert concepts and TBoxes into negation normal form (NNF) and restrict the length of paths that appear inside the constraint-based concept constructors. We start with describing NNF conversion. A concept is said to be in *negation normal form* if negation occurs only in front of concept names. The following lemma shows that NNF can be assumed without loss of generality. For a path $U = R_1 \cdots R_k g$, we write $\mathsf{ud}(U)$ to denote the concept $\forall R_1. \cdots \forall R_k.(\forall g, g.r \sqcap \forall g, g.r')$ where $r, r' \in \mathsf{Rel}$ are arbitrary such that $r \neq r'$.[1]

Lemma 7.3.1 (NNF Conversion). *Exhaustive application of the following rewrite rules translates $\mathcal{ALC}(\mathcal{C})$-concepts to equivalent ones in NNF.*

$$\begin{array}{rcll} \neg\neg C & \rightsquigarrow & C & \\ \neg(C \sqcap D) & \rightsquigarrow & \neg C \sqcup \neg D & \neg(C \sqcup D) \rightsquigarrow \neg C \sqcap \neg D \\ \neg(\exists R.C) & \rightsquigarrow & (\forall R.\neg C) & \neg(\forall R.C) \rightsquigarrow (\exists R.\neg C) \end{array}$$

$$\neg(\forall U_1, U_2.(r_1 \vee \cdots \vee r_k)) \rightsquigarrow \begin{cases} \bot & \text{if } \mathsf{Rel} = \{r_1, \ldots, r_k\} \\ \exists U_1, U_2.(\bigvee_{r \in \mathsf{Rel} \setminus \{r_1, \ldots, r_k\}} r) & \text{otherwise} \end{cases}$$

$$\neg(\exists U_1, U_2.(r_1 \vee \cdots \vee r_k)) \rightsquigarrow \begin{cases} \mathsf{ud}(U_1) \sqcup \mathsf{ud}(U_2) & \text{if } \mathsf{Rel} = \{r_1, \ldots, r_k\} \\ \forall U_1, U_2.(\bigvee_{r \in \mathsf{Rel} \setminus \{r_1, \ldots, r_k\}} r) & \text{otherwise} \end{cases}$$

By $\mathsf{nnf}(C)$, we denote the result of converting C into NNF using the above rules.

[1] This presupposes the natural assumptions that Rel has cardinality at least two.

In Lemma 7.3.1, the last two transformations are equivalence preserving since the Rel-networks used as models in $\mathcal{C}$ are complete.

We now show how to restrict the length of paths by converting concepts and TBoxes into path normal form. This normal form was first considered in [Lut04a] in the context of the description logic $\mathcal{TDL}$ and in [Lut02a] in the context of -$\mathcal{SHIQ}$.

Definition 7.3.2 (Path Normal Form). An $\mathcal{ALC}(\mathcal{C})$-concept C is in *path normal form (PNF)* if it is in NNF and for all subconcepts

$$\exists U_1, U_2.(r_1 \vee \cdots \vee r_k) \text{ and } \forall U_1, U_2.(r_1 \vee \cdots \vee r_k)$$

of C, the length of U_1 and U_2 is at most two. An $\mathcal{ALC}(\mathcal{C})$-TBox $\mathcal{T}$ is in path normal form iff $\mathcal{T}$ is of the form $\{\top \sqsubseteq C\}$, with C in PNF. △

The following lemma shows that we can w.l.o.g. assume $\mathcal{ALC}(\mathcal{C})$-concepts and TBoxes to be in PNF.

Lemma 7.3.3. *Satisfiability of $\mathcal{ALC}(\mathcal{C})$-concepts w.r.t. TBoxes can be reduced in polynomial time to satisfiability of $\mathcal{ALC}(\mathcal{C})$-concepts in PNF w.r.t. TBoxes in PNF.*

Proof. We first define an auxiliary mapping and then use this mapping to translate $\mathcal{ALC}(\mathcal{C})$-concepts into equivalent ones in PNF. Let C be an $\mathcal{ALC}(\mathcal{C})$-concept. By Lemma 7.3.1, we may assume w.l.o.g. that C is in NNF. For every feature path $u = f_1 \cdots f_n g$ used in C, we assume that $[g], [f_n g], \ldots, [f_1 \cdots f_n g]$ are fresh concrete features. We inductively define a mapping λ from feature paths u in C to concepts as follows:

$$\begin{aligned} \lambda(g) &= \top \\ \lambda(fu) &= (\exists f[u], [fu].=) \sqcap \exists f.\lambda(u) \end{aligned}$$

For every $\mathcal{ALC}(\mathcal{C})$-concept C, a corresponding concept $\rho(C)$ is obtained as follows: first replace all subconcepts $\forall u_1, u_2.(r_1 \vee \cdots \vee r_k)$ (with u_1, u_2 feature paths) with

$$\mathsf{ud}(u_1) \sqcup \mathsf{ud}(u_2) \sqcup \exists u_1, u_2.(r_1 \vee \cdots \vee r_k)$$

Then replace all subconcepts $\exists u_1, u_2.(r_1 \vee \cdots \vee r_k)$ with

$$\exists [u_1], [u_2].(r_1 \vee \cdots \vee r_k) \sqcap \lambda(u_1) \sqcap \lambda(u_2).$$

We extend the mapping ρ to TBoxes. For a TBox $\mathcal{T}$ we define

$$D_{\mathcal{T}} := \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \mathsf{nnf}(C \to D).$$

and set

$$\rho(\mathcal{T}) = \{\top \sqsubseteq \rho(D_{\mathcal{T}})\}.$$

Clearly, $\rho(C)$ and $\rho(\mathcal{T})$ are in PNF and the translation can be done in polynomial time. Moreover, it is easy to check that C is satisfiable w.r.t. $\mathcal{T}$ iff $\rho(C)$ is satisfiable w.r.t. $\rho(\mathcal{T})$: if $\mathcal{I}$ is a model of $\rho(C)$ and $\rho(\mathcal{T})$, then it can be seen that $\mathcal{I}$ is also a model of C and $\mathcal{T}$ as well. For the other direction, let $\mathcal{I}$ be a model of C and $\mathcal{T}$. A model $\mathcal{J}$ of $\rho(C)$ and $\rho(\mathcal{T})$ is obtained by extending $\mathcal{I}$ with the interpretation of freshly introduced concrete features in the following way:

$$[f_1 \ldots f_n g]^{\mathcal{J}} := f_1^{\mathcal{J}} \circ \cdots \circ f_n^{\mathcal{J}} \circ g^{\mathcal{J}}.$$

❏

The previous lemma shows that, in what follows, we may assume w.l.o.g. that all concepts and TBoxes are in PNF.

7.3.2 Data Structures

We introduce the data structures underlying the tableau algorithm, an operation for extending this data structure, and a cycle detection mechanism that is needed to ensure termination of the algorithm. As already said, we assume that the input concept C_0 is in PNF, and that the input TBox $\mathcal{T}$ is of the form $\mathcal{T} = \{\top \sqsubseteq C_\mathcal{T}\}$, where $C_\mathcal{T}$ is in PNF.

The main ingredient of the data structure underlying our algorithm is a tree that, in case of a successful run of the algorithm, represents a single model of the input concept and TBox. Due to the presence of the constraint system $\mathcal{C}$, this tree has two types of nodes: abstract ones that represent individuals of the logic domain $\Delta^\mathcal{I}$ and concrete ones that represent values of the concrete domain. We use $\mathsf{sub}(C)$ to denote the set of subconcepts of the concept C and set $\mathsf{sub}(C_0, \mathcal{T}) := \mathsf{sub}(C_0) \cup \mathsf{sub}(C_\mathcal{T})$.

Definition 7.3.4 (Completion system). Let $\mathsf{O_a}$ and $\mathsf{O_c}$ be disjoint and countably infinite sets of *abstract nodes* and *concrete nodes*. A *completion tree* for an $\mathcal{ALC}(\mathcal{C})$-concept C and a TBox $\mathcal{T}$ is a finite, labelled tree $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$ with nodes $\mathsf{V_a} \cup \mathsf{V_c}$ and edges $E \subseteq (\mathsf{V_a} \times (\mathsf{V_a} \cup \mathsf{V_c}))$ such that $\mathsf{V_a} \subseteq \mathsf{O_a}$ and $\mathsf{V_c} \subseteq \mathsf{O_c}$. The tree is labelled as follows:

1. each node $a \in \mathsf{V_a}$ is labelled with a subset $\mathcal{L}(a)$ of $\mathsf{sub}(C, \mathcal{T})$,
2. each edge $(a, b) \in E$ with $a, b \in \mathsf{V_a}$ is labelled with a role name $\mathcal{L}(a, b)$ occurring in C or $\mathcal{T}$;
3. each edge $(a, x) \in E$ with $a \in \mathsf{V_a}$ and $x \in \mathsf{V_c}$ is labelled with a concrete feature $\mathcal{L}(a, x)$ occurring in C or $\mathcal{T}$.

A node $b \in \mathsf{V_a}$ is an *R-successor* of a node $a \in \mathsf{V_a}$ if $(a, b) \in E$ and $\mathcal{L}(a, b) = R$, while $x \in \mathsf{V_c}$ is a *g-successor* of a if $(a, x) \in E$ and $\mathcal{L}(a, x) = g$. The notion *U-successor* for a path U is defined in the obvious way.

A *completion system* for an $\mathcal{ALC}(\mathcal{C})$-concept C and a TBox $\mathcal{T}$ is a pair $S = (T, \mathcal{N})$ where $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$ is a completion tree for C and $\mathcal{T}$ and $\mathcal{N}$ is a Rel-network with $V_\mathcal{N} = \mathsf{V_c}$. △

We now define an operation that is used by the tableau algorithm to add new nodes to completion trees. The operation respects the functionality of abstract and concrete features.

Definition 7.3.5 ($\oplus$ Operation). An abstract or concrete node is called *fresh* in a completion tree T if it does not appear in T. Let $S = (T, \mathcal{N})$ be a completion system with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$. We use the following operations:

- if $a \in \mathsf{V_a}$, $b \in \mathsf{O_a}$ fresh in T, and $R \in \mathsf{N_R}$, then $S \oplus aRb$ yields the completion system obtained from S in the following way:
  - if $R \notin \mathsf{N_{aF}}$ or $R \in \mathsf{N_{aF}}$ and a has no R-successors, then add b to $\mathsf{V_a}$, (a, b) to E and set $\mathcal{L}(a, b) = R$, $\mathcal{L}(b) = \emptyset$.
  - if $R \in \mathsf{N_{aF}}$ and there is a $c \in \mathsf{V_a}$ such that $(a, c) \in E$ and $\mathcal{L}(a, c) = R$ then rename c in T with b.

- if $a \in \mathsf{V_a}$, $x \in \mathsf{O_c}$ fresh in T, and $g \in \mathsf{N_{cF}}$, then $S \oplus agx$ yields the completion system obtained from S in the following way:
  - if a has no g-successors, then add x to $\mathsf{V_c}$, (a, x) to E and set $\mathcal{L}(a, x) = g$;
  - if a has a g-successor y, then rename y in T and $\mathcal{N}$ with x.

Let $U = R_1 \cdots R_n g$ be a path. With $S \oplus aUx$, where $a \in \mathsf{V_a}$ and $x \in \mathsf{O_c}$ is fresh in T, we denote the completion system obtained from S by taking distinct nodes $b_1, ..., b_n \in \mathsf{O_a}$ which are fresh w.r.t. T and setting

$$S' := S \oplus aR_1b_1 \oplus \cdots \oplus b_{n-1}R_nb_n \oplus b_ngx$$

△

The tableau algorithm works by starting with an initial completion system that is then successively expanded with the goal of constructing a model of the input concept and TBox. To ensure termination, we need a mechanism for detecting cyclic expansions, which is commonly called *blocking*. Informally, we detect nodes in the completion tree that are similar to previously created ones and then block them, i.e., stop further expansion at such nodes. To introduce blocking, we start with some preliminaries. For $a \in \mathsf{V_a}$, we define the set of features of a as

$$\mathsf{feat}(a) := \{\ g \in \mathsf{N_{cF}} \mid a \text{ has a } g\text{-successor }\}.$$

Next, we define the *concrete neighborhood* of a as the constraint network

$$\begin{array}{rcll} \mathcal{N}(a) & := & \{\ (x\ r\ y) \mid & \text{there exist } g, g' \in \mathsf{feat}(a) \text{ s.t. } x \text{ is a } g\text{-succ.} \\ & & & \text{of } a,\ y \text{ is a } g'\text{-succ. of } a, \text{ and } (x\ r\ y) \in \mathcal{N}\ \} \end{array}$$

Finally, if $a, b \in \mathsf{V_a}$ and $\mathsf{feat}(a) = \mathsf{feat}(b)$, we write $\mathcal{N}(a) \sim \mathcal{N}(b)$ to express that $\mathcal{N}(a)$ and $\mathcal{N}(b)$ are isomorphic, i.e., that the mapping $\pi : V_{\mathcal{N}(a)} \to V_{\mathcal{N}(b)}$ defined by mapping the g-successor of a to the g-successor of b for all $g \in \mathsf{feat}(a)$ is an isomorphism.

If T is a completion tree and a and b are abstract nodes in T, then we say that a is an *ancestor* of b if b is reachable from a in the tree T.

Definition 7.3.6 (Blocking). Let $S = (T, \mathcal{N})$ be a completion system for a concept C_0 and a TBox $\mathcal{T}$ with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$, and let $a, b \in \mathsf{V_a}$. We say that $a \in \mathsf{V_a}$ is *potentially blocked by* b if the following holds:

1. b is an ancestor of a in T,
2. $\mathcal{L}(a) \subseteq \mathcal{L}(b)$,
3. $\mathsf{feat}(a) = \mathsf{feat}(b)$.

We say that a is *directly blocked* by b if the following holds:

1. a is potentially blocked by b,
2. $\mathcal{N}(a)$ and $\mathcal{N}(b)$ are complete, and
3. $\mathcal{N}(a) \sim \mathcal{N}(b)$.

Finally, a is *blocked* if it or one of its ancestors is directly blocked. △

$R\sqcap$ if $C_1 \sqcap C_2 \in \mathcal{L}(a)$, a is not blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(a)$,
then set $\mathcal{L}(a) := \mathcal{L}(a) \cup \{C_1, C_2\}$

$R\sqcup$ if $C_1 \sqcup C_2 \in \mathcal{L}(a)$, a is not blocked, and $\{C_1, C_2\} \cap \mathcal{L}(a) = \emptyset$,
then set $\mathcal{L}(a) := \mathcal{L}(a) \cup \{C\}$ for some $C \in \{C_1, C_2\}$

$R\exists$ if $\exists R.C \in \mathcal{L}(a)$, a is not blocked, and there is no R-successor b of a such that $C \in \mathcal{L}(b)$
then set $S := S \oplus aRb$ for a fresh $b \in O_a$ and $\mathcal{L}(b) := \mathcal{L}(b) \cup \{C\}$

$R\forall$ if $\forall R.C \in \mathcal{L}(a)$, a is not blocked, and b is an R-successor of a such that $C \notin \mathcal{L}(b)$
then set $\mathcal{L}(b) := \mathcal{L}(b) \cup \{C\}$

$R\exists_c$ if $\exists U_1, U_2.(r_1 \vee \cdots \vee r_k) \in \mathcal{L}(a)$, a is not blocked, and there exist no $x_1, x_2 \in V_c$ such that x_i is a U_i-successor of a for $i = 1, 2$ and $(x_1\ r_i\ x_2) \in \mathcal{N}$ for some i with $1 \leq i \leq k$
then set $S := S \oplus aU_1x_1 \oplus aU_2x_2$ with $x_1, x_2 \in \mathsf{O_c}$ fresh and $\mathcal{N} := \mathcal{N} \cup \{(x_1\ r_i\ x_2)\}$ for some i with $1 \leq i \leq k$

$R\forall_c$ if $\forall U_1, U_2.(r_1 \vee \cdots \vee r_k) \in \mathcal{L}(a)$, a is not blocked, and there are $x_1, x_2 \in \mathsf{V_c}$ such that x_i is a U_i-successor of a for $i = 1, 2$ and $(x_1\ r_i\ x_2) \notin \mathcal{N}$ for all i with $1 \leq i \leq k$
then set $\mathcal{N} := \mathcal{N} \cup \{(x_1\ r_i\ x_2)\}$ for some i with $1 \leq i \leq k$

$R\mathsf{net}$ if a is potentially blocked by b or vice versa and $\mathcal{N}(a)$ is not complete
then non-deterministically guess a completion $\mathcal{N}'$ of $\mathcal{N}(a)$ and set $\mathcal{N} := \mathcal{N} \cup \mathcal{N}'$

$R\mathsf{tbox}$ if $C_\mathcal{T} \notin \mathcal{L}(a)$
then set $\mathcal{L}(a) := \mathcal{L}(a) \cup \{C_\mathcal{T}\}$

Figure 7.4: The completion rules.

7.3.3 The Tableau Algorithm

To decide the satisfiability of an $\mathcal{ALC}(\mathcal{C})$-concept C_0 w.r.t. a TBox $\mathcal{T}$, the tableau algorithm is started with the initial completion system $S_{C_0} = (T_{C_0}, \emptyset)$, where the initial completion tree T_{C_0} is defined by setting

$$T_{C_0} := (\{a_0\}, \emptyset, \emptyset, \{a_0 \mapsto \{C_0\}\}).$$

The algorithm then repeatedly applies the completion rules given in Figure 7.4. In the formulation of $R\mathsf{net}$, a *completion* of a Rel-network N is a satisfiable and complete Rel-network N' such that $V_N = V_{N'}$ and $N \subseteq N'$. Later on, we will argue that the completion to be guessed always exists.

As has already been noted above, rule application can be understood as the step-wise construction of a model of C_0 and $\mathcal{T}$. Among the rules, there are four non-deterministic ones: $R\sqcup$, $R\exists_c$, $R\forall_c$, and $R\mathsf{net}$.[2] Rules are applied until an obvious inconsistency (as defined below) is detected or the completion system becomes *complete*, i.e., no more rules are applicable. The algorithm returns "satisfiable" if there is a way to apply the rules such that a complete completion system is found that does not contain a contradiction. Otherwise, it returns "unsatisfiable".

All rules except $R\mathsf{net}$ are rather standard, see for example [BH91, Lut02b].[3] The purpose

[2] By disallowing disjunctions of relations in the constraint-based concept constructors, $R\exists_c$ and $R\forall_c$ can easily be made deterministic.

[3] Note that our version of the $R\exists$ rule uses the operation $S \oplus aRb$ which initializes the label $\mathcal{L}(b)$, and thus

```
procedure sat(S)
 if S contains a clash then return unsatisfiable
 if S is complete then return satisfiable
 if Rnet is applicable
  then S' := application of Rnet to S
  else S' := application of any applicable completion rule to S
 return sat(S')
```

Figure 7.5: The (non-deterministic) algorithm for satisfiability in $\mathcal{ALC}(\mathcal{C})$.

of $R\mathsf{net}$ is to resolve a potential blocking situation between two nodes a and b into either an actual blocking situation or a non-blocking situation. This is achieved by completing the networks $\mathcal{N}(a)$ and $\mathcal{N}(b)$. For ensuring termination, an appropriate interplay between this rule and the blocking condition is crucial. Namely, we have to apply $R\mathsf{net}$ with highest precedence. It can be seen that the blocking mechanism obtained in this way is a refinement of pairwise blocking as known from [HST99]. In particular, the conditions $\mathcal{L}(a) \subseteq \mathcal{L}(b)$ and $\mathsf{feat}(a) = \mathsf{feat}(b)$ are implied by the standard definition of pairwise blocking due to path normal form.

We now define what we mean by an obvious inconsistency. As soon as such an inconsistency is encountered, the tableau algorithm returns "unsatisfiable".

Definition 7.3.7 (Clash). Let $S = (T, \mathcal{N})$ be a completion system for a concept C and a TBox $\mathcal{T}$ with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$. S contains a *clash* if one of the following conditions holds:

1. there is an $a \in \mathsf{V_a}$ and an $A \in \mathsf{N_C}$ such that $\{A, \neg A\} \subseteq \mathcal{L}(a)$;
2. $\mathcal{N}$ is not satisfiable in $\mathcal{C}$.

If S does not contain a clash, S is called *clash-free*. △

We present the tableau algorithm in pseudo-code notation in Figure 7.5. It is started with the initial completion system as argument, i.e., by calling $\mathsf{sat}(S_{C_0})$.

Note that checking for clashes before rule application is crucial for $R\mathsf{net}$ to be well-defined: if $R\mathsf{net}$ is applied to a node a, we must be sure that there indeed exists a completion $\mathcal{N}'$ of $\mathcal{N}(a)$ to be guessed, i.e., a *satisfiable* network $\mathcal{N}'$ such that $V'_{\mathcal{N}} = V_{\mathcal{N}(a)}$ and $\mathcal{N}(a) \subseteq \mathcal{N}'$. Clash checking before rule application ensures that the network $\mathcal{N}$ is satisfiable when $R\mathsf{net}$ is applied. Clearly, this implies the existence of the required completion.

7.4 Correctness

We prove termination, soundness and completeness of the presented tableau algorithm. In the following, we use $|M|$ to denote the cardinality of a set M. With $\mathsf{N_C}^{C_0,\mathcal{T}}$, $\mathsf{N_R}^{C_0,\mathcal{T}}$ and $\mathsf{N_{cF}}^{C_0,\mathcal{T}}$, we denote the sets of concept names, role names, and concrete features that occur in the concept C_0 and the TBox $\mathcal{T}$. We use $|C|$ to denote the length of a concept C and $|\mathcal{T}|$ to denote $\sum_{C \sqsubseteq D \in \mathcal{T}} (|C| + |D|)$.

Lemma 7.4.1 (Termination). *The tableau algorithm terminates on every input.*

the rule only *adds* C to the already existing label.

Proof. Let $S_0, S_1, \ldots$ be the sequence of completion systems generated during the run of the tableau algorithm started on input C_0, $\mathcal{T}$, and let $S_i = (T_i, \mathcal{N}_i)$. Set $n := |C_0| + |\mathcal{T}|$. Obviously, we have $|\mathsf{sub}(C_0, \mathcal{T})| \leq n$. We first show the following:

(a) For all $i \geq 0$, the out-degree of T_i is bounded by n.

(b) For $i \geq 0$, the depth of T_i is bounded by $\ell = 2^{2n} \cdot |\mathsf{Rel}|^{n^2} + 2$.

First for (a). Nodes from $\mathsf{V_c}$ do not have successors. Let $a \in \mathsf{V_a}$. Successors of a are created only by applications of the rules $R\exists$ and $R\exists_c$. The rule $R\exists$ generates at most one abstract successor (i.e., element of $\mathsf{V_a}$) of a for each $\exists R.C \in \mathsf{sub}(C_0, \mathcal{T})$, and $R\exists_c$ generates at most two abstract successors of a for every $\exists U_1, U_2.(r_1 \vee \cdots \vee r_k) \in \mathsf{sub}(C_0, \mathcal{T})$. Moreover, $R\exists_c$ generates at most one concrete successor for every element of $\mathsf{N}_{\mathsf{cF}}^{C_0,\mathcal{T}}$. It is not difficult to verify that this implies that the number of (abstract and concrete) successors of a is bounded by n.

Now for (b). Assume, to the contrary of what is to be shown, that there is an $i \geq 0$ such that the depth of T_i exceeds $\ell = 2^{2n} \cdot |\mathsf{Rel}|^{n^2} + 2$. Moreover, let i be smallest with this property. This means that S_i has been obtained from S_{i-1} by applying one of the rules $R\exists$ and $R\exists_c$ to a node on level ℓ, or by applying $R\exists_c$ to a node on level $\ell - 1$.

Let $T_{i-1} = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$. Since T_i is obtained from T_{i-1} by application of $R\exists$ or $R\exists_c$ and since $R\mathsf{net}$ is applied with highest precedence, $R\mathsf{net}$ is not applicable to T_{i-1}. This means that, for every $a, b \in \mathsf{V_a}$ such that b is potentially blocked by a, $\mathcal{N}_{i-1}(a)$ and $\mathcal{N}_{i-1}(b)$ are complete. Let us define a binary relation $\approx$ on $\mathsf{V_a}$ as follows:

$$a \approx b \quad \text{iff} \quad \mathcal{L}(a) = \mathcal{L}(b),\ \mathsf{feat}(a) = \mathsf{feat}(b),\ \text{and } \mathcal{N}_{i-1}(a) \sim \mathcal{N}_{i-1}(b).$$

Obviously, $\approx$ is an equivalence relation on $\mathsf{V_a}$. The definition of blocking implies that if a is an ancestor of b and $a \approx b$, then b is blocked by a in S_{i-1}. Let $\mathsf{V_a}/_{\approx}$ denote the set of $\approx$-equivalence classes and set $m := |\mathsf{N}_{\mathsf{cF}}^{C_0,\mathcal{T}}|$. Since $\mathcal{L}(a) \subseteq \mathsf{sub}(C_0, \mathcal{T})$, and $\mathcal{N}_{i-1}(a)$ is a complete Rel-network with $|V_{\mathcal{N}_{i-1}(a)}| \leq m$ for all $a \in \mathsf{V_a}$, it is not difficult to verify that

$$|\mathsf{V_a}/_{\approx}| \leq 2^{|\mathsf{sub}(C_0,\mathcal{T})|} \sum_{i=0}^{m} \binom{m}{i} |\mathsf{Rel}|^{i^2}$$

Since $m \leq n$, we obtain $|\mathsf{V_a}/_{\approx}| \leq 2^n \cdot 2^n \cdot |\mathsf{Rel}|^{n^2} = 2^{2n} \cdot |\mathsf{Rel}|^{n^2}$. Let $a \in \mathsf{V_a}$ be the node to which a rule is applied in T_{i-1} to obtain T_i. As already noted, the level k of a in T_{i-1} is at least $\ell - 1 \geq |\mathsf{V_a}/_{\approx}| + 1$. Let $a_0, \ldots, a_k$ be the path in T_{i-1} leading from the root to a. Since $k > |\mathsf{V_a}/_{\approx}|$, we have $a_i \approx a_j$ for some i, j with $0 \leq i < j \leq k$. This means that a is blocked and contradicts the assumption that a completion rule was applied to a. Thus, the proof of (b) is finished.

The tableau algorithm terminates due to the following reasons:

1. It constructs a finitely labelled completion tree T of bounded out-degree and depth (by (a) and (b)) in a monotonic way, i.e., no nodes are removed from T and no concepts are removed from node labels. Also, no constraints are removed from the constraint system $\mathcal{N}$;

2. every rule application adds new nodes or node labels to T, or new constraints to $\mathcal{N}$;

3. the cardinality of node labels is bounded by $|\mathsf{sub}(C_0, \mathcal{T})|$ and the number of constraints in $\mathcal{N}$ is bounded by $|\mathsf{Rel}| \cdot k^2$, with k the (bounded) number of concrete nodes.

❑

Lemma 7.4.2 (Soundness). *If the tableau algorithm returns* satisfiable*, then the input concept C_0 is satisfiable w.r.t. the input TBox $\mathcal{T}$.*

Proof. If the tableau algorithm returns satisfiable, then there exists a complete and clash-free completion system $S = (T, \mathcal{N})$ for C_0 and $\mathcal{T}$. Our aim is to use S for defining a model $\mathcal{I}$ for C_0 and $\mathcal{T}$. We start with a brief outline of the proof.

To obtain the desired model $\mathcal{I}$, the completion tree T is unravelled to another (possibly infinite) tree by replacing directly blocked nodes with nodes that block them. The second condition of "potentially blocked" ensures that by doing this, we do not violate any existential or universal conditions in the predecessor of a directly blocked node. This yields only the abstract part of $\mathcal{I}$. Defining the concrete part is less straightforward. To start with, the described unravelling process can be seen as follows. We start with the tree T where all indirectly blocked nodes are dropped, and then repeatedly patch subtrees of T to the existing tree. More precisely, such a patched subtree is rooted by a node that blocks the node onto which the root of the subtree is patched. The third condition of "directly blocked" ensures that the networks $\mathcal{N}(a)$ and $\mathcal{N}(b)$ (which comprise only the concrete successors a and b) are complete and identical if a is blocked by b. This means that we can obtain a (possibly infinite) constraint network $\mathbf{N}$ that corresponds to the unravelled tree by patching together fragments of $\mathcal{N}$ which coincide on overlapping parts. Since $\mathcal{N}$ is satisfiable, patchwork and compactness property ensure that the network $\mathbf{N}$ is satisfiable as well and thus we can use a model of $\mathbf{N}$ to define the concrete part of the model $\mathcal{I}$.

Formally, we proceed in several steps. Let $S = (T, \mathcal{N})$ be as above, $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$, and let $\mathsf{root} \in \mathsf{V_a}$ denote the root of T. Let blocks be a function that for every directly blocked $b \in \mathsf{V_a}$, returns an unblocked $a \in \mathsf{V_a}$ such that b is blocked by a in S. It is easily seen that, by definition of blocking, such node a always exists. A *path* in S is a (possibly empty) sequence of pairs of nodes $\frac{a_1}{b_1}, \ldots, \frac{a_n}{b_n}$, with $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ nodes from $\mathsf{V_a}$, such that, for $1 \leq i < n$, one of the following holds:

1. a_{i+1} is a successor of a_i in T, a_{i+1} is unblocked, and $b_{i+1} = a_{i+1}$;
2. b_{i+1} is a successor of a_i in T and $a_{i+1} = \mathsf{blocks}(b_{i+1})$.

Intuitively, a path $\frac{a_1}{b_1}, \ldots, \frac{a_n}{b_n}$ represents the sequence of nodes $a_1, \ldots, a_n$, and the b_i provide justification for the existence of the path in case of blocking situations. Observe that b_{i+1} is always a successor of a_i. We use Paths to denote the set of all paths in S including the empty path. For $p \in \mathsf{Paths}$ nonempty, $\mathsf{tail}(p)$ denotes the last pair of p. We now define the "abstract part" of the model $\mathcal{I}$ we are constructing:

$$
\begin{aligned}
\Delta^{\mathcal{I}} &:= \{p \in \mathsf{Paths} \mid p \text{ non-empty and first pair is } \frac{\mathsf{root}}{\mathsf{root}}\} \\
A^{\mathcal{I}} &:= \{p \in \Delta^{\mathcal{I}} \mid \mathsf{tail}(p) = \frac{a}{b} \text{ and } A \in \mathcal{L}(a)\},\ A \in \mathsf{N_C}^{C_0,\mathcal{T}} \\
R^{\mathcal{I}} &:= \{(p, p \cdot \frac{a}{b}) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \mathsf{tail}(p) = \frac{a'}{b'} \text{ and } b \text{ is} \\
&\qquad\qquad R\text{-successor of } a' \text{ in } T\ \},\ R \in \mathsf{N_R}^{C_0,\mathcal{T}}
\end{aligned}
$$

Observe that

(i) $\Delta^{\mathcal{I}}$ is non-empty, since $\frac{\mathsf{root}}{\mathsf{root}} \in \Delta^{\mathcal{I}}$.

(ii) $f^{\mathcal{I}}$ is functional for every $f \in \mathsf{N_{aF}}$: this is ensured by the "$\oplus$" operation which generates at most one f-successor per abstract node, and by the definition of Paths in which we choose only a single blocking node to be put into a path.

Intuitively, the abstract part of $\mathcal{I}$ as defined above is obtained by "patching together" parts of the completion tree T. For defining the concrete part of $\mathcal{I}$, we make this patching explicit: For $p \in \Delta^{\mathcal{I}}$, p is called a *hook* if $p = \frac{\mathsf{root}}{\mathsf{root}}$ or $\mathsf{tail}(p) = \frac{a}{b}$ with $a \neq b$ (and thus b is blocked by a). We use Hooks to denote the set of all hooks. Intuitively, the hooks, which are induced by blocking situations in T, are the points where we patch together parts of T. The part of T patched at a hook p with $\mathsf{tail}(p) = \frac{a}{b}$ is comprised of (copies of) all the nodes c in T that are reachable from a, except indirectly blocked ones. Formally, for $p \in \Delta^{\mathcal{I}}$ and $q \in \mathsf{Hooks}$, we call p a *q-companion* if there exists $q' \in \mathsf{Paths}$ such that $p = qq'$ and all nodes $\frac{a}{b}$ in q' satisfy $a = b$, with the possible exception of $\mathsf{tail}(q')$. Then, the part of $\mathcal{I}$ patched at p is defined as

$$P(p) := \{q \in \Delta^{\mathcal{I}} \mid q \text{ is a } p\text{-companion}\}.$$

For $p, q \in \mathsf{Hooks}$, q is called a *successor* of p if q is a p-companion and $p \neq q$. Observe that, for each hook p, $P(p)$ includes p and all successor hooks of p. Intuitively, this means that the parts patched together to obtain the abstract part of $\mathcal{I}$ are overlapping at the hooks.

To define the concrete part of $\mathcal{I}$, we need to establish some additional notions. Since S is clash-free, $\mathcal{N}$ is satisfiable. It is an easy exercise to show that then there exists a completion of $\mathcal{N}$. We fix such a completion $\mathcal{N}^c$ with the nodes renamed as follows: each concrete node x that is a g-successor of an abstract node a is renamed to the pair (a, g). This naming scheme is well-defined since the "$\oplus$" operation ensures that every abstract node a has at most one g-successor, for every $g \in \mathsf{N_{cF}}$. We now define a network $\mathbf{N}$ which, intuitively, describes the constraints put on the concrete part of the model. If $q \in \mathsf{Hooks}$, $p \in P(q)$, and $\mathsf{tail}(p) = \frac{a}{b}$, we set

$$\mathsf{rep}_q(p) := \begin{cases} b & \text{if } p \neq q \text{ and } a \neq b \\ a & \text{otherwise} \end{cases}$$

Intuitively, this notion is needed for the following reason: let $p, q \in \mathsf{Hooks}$ with q a successor of p. Then $\mathsf{tail}(q) = \frac{a}{b}$ with b blocked by a, $q \in P(p)$, and $q \in P(q)$. As part of $P(p)$, q represents the blocked node b. As part of $P(q)$, q represents the blocking node a. This overlapping of patched parts at hooks is made explicit via the notion $\mathsf{rep}_q(p)$. Now define $\mathbf{N}$ as follows:

$$\begin{aligned} \mathbf{N} \;:=\; \{((p,g)\, r\, (p',g')) \mid & \text{ there is a } q \in \mathsf{Hooks} \text{ such that } p, p' \in P(q) \\ & \text{and } ((\mathsf{rep}_q(p), g)\, r\, (\mathsf{rep}_q(p'), g')) \in \mathcal{N}^c\} \end{aligned}$$

Our next aim is to show that $\mathbf{N}$ is satisfiable. To this end, we first show that $\mathbf{N}$ is patched together from smaller networks: every hook p gives rise to a part of $\mathbf{N}$ as follows:

$$N(p) := \mathbf{N}|_{\{(q,g) \in V_{\mathbf{N}} \mid q \in P(p)\}},$$

i.e, $N(p)$ is the restriction of $\mathbf{N}$ to those variables (q, g) such that q is a p-companion.

The following claim shows that $\mathbf{N}$ is patched together from the networks $N(p)$, $p \in \mathsf{Hooks}$.

Claim 1. The following holds:

(a) $\mathbf{N} = \bigcup_{p \in \mathsf{Hooks}} N(p)$.

(b) if $p, q \in \mathsf{Hooks}$, $p \neq q$, q is not a successor of p, and p is not a successor of q, then $V_{N(p)} \cap V_{N(q)} = \emptyset$;

(c) if $p, q \in \mathsf{Hooks}$ and q is a successor of p, then $N(p)|_{V_{N(p)} \cap V_{N(q)}} = N(q)|_{V_{N(p)} \cap V_{N(q)}}$;

Proof. (a) As $\mathbf{N} \supseteq \bigcup_{p \in \mathsf{Hooks}} N(p)$ is immediate by definition of $N(p)$, it remains to show $\mathbf{N} \subseteq \bigcup_{p \in \mathsf{Hooks}} N(p)$. Thus, let $((p, g)\, r\, (p', g')) \in \mathbf{N}$. Then there is a $q \in \mathsf{Hooks}$ such that $p, p' \in P(q)$. By definition of $N(q)$, this implies $((p, g)\, r\, (p', g')) \in N(q)$.

(b) We show the contrapositive. Let $(q^*, g) \in V_{N(p)} \cap V_{N(q)}$. It follows that $q^* \in P(p) \cap P(q)$, i.e., there are $q', q'' \in \mathsf{Paths}$ such that (i) $q^* = pq'$, $q^* = qq''$, and (ii) all nodes $\frac{a}{b}$ in q', q'' satisfy $a = b$, with the possible exception of the last one. Due to (i), $p = q$, p is a prefix of q, or vice versa. In the first case, we are done. In the second case, since $q \in \mathsf{Hooks}$ we have that $\mathsf{tail}(q) = \frac{a}{b}$ for some a, b with $a \neq b$. Together with $q^* = pq'$, (ii), and since p is a prefix of q is a prefix of q^*, this implies that $q = q^*$. Thus $q = pq'$. Again by (ii), we have that q is a successor of p. The third case is analogous to the second.

(c) By definition of $N(p)$ and $N(q)$, we have $N(p)|_{V_{N(p)} \cap V_{N(q)}} = \mathbf{N}|_{V_{N(p)} \cap V_{N(q)}} = N(q)|_{V_{N(p)} \cap V_{N(q)}}$ for all $p, q \in \mathsf{Hooks}$.

Claim 1 shows that $\mathbf{N}$ is patched together from smaller networks. Our aim is to apply the patchwork and compactness property to derive satisfiability of $\mathbf{N}$. For being able to do this, we additionally need to know that the smaller networks are complete and satisfiable, and that they agree on overlapping parts. Before we prove this, we establish some crucial properties.

(P1) If $q, q' \in \mathsf{Hooks}$ with q' successor of q, then $V_{P(q)} \cap V_{P(q')} = \{q'\}$.

(P2) If $((q, g)\, r\, (q', g')) \in N(p)$ then $((\mathsf{rep}_p(q), g)\, r\, (\mathsf{rep}_p(q'), g')) \in \mathcal{N}^c$.

(P1) is obvious by definition of hooks and q-companions. For (P2), let $((q, g)\, r\, (q', g')) \in N(p)$. Then $q, q' \in P(p)$. Since $N(p) \subseteq \mathbf{N}$, there is a $p' \in \mathsf{Hooks}$ such that $q, q' \in P(p')$ and

$(*)$ $((\mathsf{rep}_{p'}(q), g)\, r\, (\mathsf{rep}_{p'}(q'), g')) \in \mathcal{N}^c$.

If $p = p'$, we are done. Thus, let $p \neq p'$. By Claim 1(b) and (P1), $q, q' \in P(p) \cap P(p')$ implies that $q = q' = p$ and p is a successor-hook of p', or $q = q' = p'$ and p' is a successor-hook of p. W.l.o.g., assume that the former is the case. Let $\mathsf{tail}(q) = \frac{a}{b}$. Since $q = p$ and p is a hook, we have $a \neq b$, and thus b is blocked by a in T. By definition of rep, we have $\mathsf{rep}_{p'}(q) = b$ and $\mathsf{rep}_p(q) = a$. Thus, $(*)$ yields $((b, g)\, r\, (b, g')) \in \mathcal{N}^c$. Since b is blocked by a, the blocking condition yields $((a, g)\, r\, (a, g')) \in \mathcal{N}^c$ and we are done. This finishes the proof of Claim 1.

Claim 2. For every $p \in \mathsf{Hooks}$, $N(p)$ is finite, complete, and satisfiable.

Proof. Let $p \in \mathsf{Hooks}$. Since the completion tree T is finite, so are $P(p)$ and $N(p)$. Next, we show that $N(p)$ is complete. This involves two subtasks: showing that (i) for all $(q, g), (q', g') \in V_{N(p)}$, there is at least one relation r with $((q, g)\, r\, (q', g')) \in N(p)$; and (ii) there is at most one such relation.

For (i), let $(q, g), (q', g') \in V_{N(p)}$. By (P2), we obtain that $(\mathsf{rep}_p(q), g), (\mathsf{rep}_p(q'), g') \in V_{\mathcal{N}^c}$. Since $\mathcal{N}^c$ is complete, there is an r such that $((\mathsf{rep}_p(q), g)\, r\, (\mathsf{rep}_p(q'), g')) \in \mathcal{N}^c$. By definition of $\mathbf{N}$ and $N(p)$, we have $((q, g)\, r\, (q', g')) \in N(p)$. For (ii), assume that $((q, g)\, r\, (q', g')) \in$

$N(p)$, for each $r \in \{r_1, r_2\}$. Then, (P2) implies $((\mathsf{rep}_p(q), g)\, r_i\, (\mathsf{rep}_p(q'), g')) \in \mathcal{N}^c$ for each $r \in \{r_1, r_2\}$. Thus, completeness of $\mathcal{N}^c$ implies that $r_1 = r_2$ as required.

Finally, we show satisfiability of $N(p)$. By (P2), $((q, g)\, r\, (q', g')) \in N(p)$ implies $((\mathsf{rep}_p(q), g)\, r\, (\mathsf{rep}_p(q'), g')) \in \mathcal{N}^c$. Thus, satisfiability of $\mathcal{N}^c$, yields satisfiability of $N(p)$.

We are now ready to apply the patchwork and compactness properties.

Claim 3. $\mathbf{N}$ is satisfiable.

Proof. First assume that there are no blocked nodes in S. Then, $\mathsf{Hooks} = \{\frac{\mathsf{root}}{\mathsf{root}}\}$. By Claim 1(a), we have that $\mathbf{N} = N(\frac{\mathsf{root}}{\mathsf{root}})$, and by Claim 2 we obtain that $\mathbf{N}$ is satisfiable. Now assume that there are blocked nodes in S. Since $\mathsf{V_a}$ is finite (c.f. Lemma 7.4.1), Hooks is a countably infinite set. Moreover, the "successor" relation on Hooks is easily seen to arrange Hooks in an infinite tree whose out-degree is bounded by the cardinality of $\mathsf{V_a}$. Therefore, we can fix an enumeration $\{p_0, p_1, ...\}$ of Hooks such that:

- $p_0 = \frac{\mathsf{root}}{\mathsf{root}}$,
- if p_i is a successor of p_j, then $i > j$.

By Claim 1(a), we have that $\mathbf{N} = \bigcup_{i \geq 0} N(p_i)$. We first show by induction that, for all $k \geq 0$, the network $\mathbf{N}_k := \bigcup_{0 \leq i \leq k} N(p_i)$ is satisfiable.

- $k = 0$: $\mathbf{N}_0 = N(p_0)$ is satisfiable by Claim 2.
- $k > 0$. We have that $\mathbf{N}_k = \mathbf{N}_{k-1} \cup N(p_k)$. By induction, $\mathbf{N}_{k-1}$ is satisfiable. Let $\mathbf{N}^c_{k-1}$ be a completion of $\mathbf{N}_{k-1}$ and let $\mathbf{N}'_k = \mathbf{N}^c_{k-1} \cup N(p_k)$. There exists a unique $p_n \in \mathsf{Hooks}$, $n < k$, such that p_k is a successor of p_n. By definition of $\mathbf{N}_{k-1}$ and Claim 1(b), and since $V_{\mathbf{N}^c_{k-1}} = V_{\mathbf{N}_{k-1}}$, we have that

$$V_{\mathbf{N}^c_{k-1}} \cap V_{N(p_k)} = V_{N(p_n)} \cap V_{N(p_k)}.$$

Moreover, by Claim 2, $N(p_n)$ is complete, and thus

$$\mathbf{N}^c_{k-1}|_{V_{N(p_n)} \cap V_{N(p_k)}} = N(p_n)|_{V_{N(p_n)} \cap V_{N(p_k)}}.$$

Finally, Claim 1(c) yields

$$N(p_n)|_{V_{N(p_n)} \cap V_{N(p_k)}} = N(p_k)|_{V_{N(p_n)} \cap V_{N(p_k)}}.$$

Summing up, we obtain that the intersection parts of $\mathbf{N}^c_{k-1}$ and $N(p_k)$ are identical:

$$\begin{aligned} \mathbf{N}^c_{k-1}|_{V_{\mathbf{N}^c_{k-1}} \cap V_{N(p_k)}} &= \mathbf{N}^c_{k-1}|_{V_{N(p_n)} \cap V_{N(p_k)}} \\ &= N(p_n)|_{V_{N(p_n)} \cap V_{N(p_k)}} \\ &= N(p_k)|_{V_{N(p_n)} \cap V_{N(p_k)}} \\ &= N(p_k)|_{V_{\mathbf{N}^c_{k-1}} \cap V_{N(p_k)}} \end{aligned}$$

By Claim 2, we have that $N(p_k)$ is finite, complete and satisfiable. The same holds for $\mathbf{N}^c_{k-1}$. Thus, the patchwork property of $\mathcal{C}$ yields that $\mathbf{N}'_k$ is satisfiable. Since $\mathbf{N}_k \subseteq \mathbf{N}'_k$, $\mathbf{N}_k$ is satisfiable.

Now, satisfiability of the networks $\mathbf{N}_k$, $k \geq 0$, and the compactness property of $\mathcal{C}$ imply satisfiability of $\mathbf{N}$. This finishes the proof of Claim 3.

We are now ready to define the concrete part of the model $\mathcal{I}$. Since $\mathbf{N}$ is satisfiable, there is an $M_\mathcal{I} \in \mathfrak{M}$ and a mapping $\tau : V_\mathbf{N} \to V_{M_\mathcal{I}}$ such that $(x\ r\ y) \in \mathbf{N}$ implies $(\tau(x)\ r\ \tau(y)) \in V_{M_\mathcal{I}}$. Define $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I}, M_\mathcal{I})$ with $\Delta^\mathcal{I}$ and $\cdot^\mathcal{I}$ defined as above, and, additionally:

$$g^\mathcal{I} := \{(p, \tau(p,g)) \in \Delta^\mathcal{I} \times V_{M_\mathcal{I}} \mid \mathsf{tail}(p) = \frac{a}{b} \text{ and } g \in \mathsf{feat}(a)\},\ g \in \mathsf{N}_{\mathsf{cF}}^{C_0,\mathcal{T}}$$

Note that, by definition, $g^\mathcal{I}$ is functional for every $g \in \mathsf{N}_{\mathsf{cF}}$. In order to show that $\mathcal{I}$ is a model of C_0 and $\mathcal{T}$, we require one more claim:

Claim 4. For all $s \in \Delta^\mathcal{I}$ and $C \in \mathsf{sub}(C_0, \mathcal{T})$, if $\mathsf{tail}(s) = \frac{a}{b}$ and $C \in \mathcal{L}(a)$, then $s \in C^\mathcal{I}$.

Proof. We prove the claim by structural induction on C. Let $s \in \Delta^\mathcal{I}$, $\mathsf{tail}(s) = \frac{a}{b}$, and $C \in \mathcal{L}(a)$. In the following, we will implicitly use the fact that, by construction of Paths, a is not blocked in S. We make a case distinction according to the topmost operator in C:

1. C is a concept name. By construction of $\mathcal{I}$, we have $s \in C^\mathcal{I}$.

2. $C = \neg D$. Since C is in NNF, D is a concept name. Clash-freeness of S implies $D \notin \mathcal{L}(a)$. The construction of $\mathcal{I}$ implies $s \notin D^\mathcal{I}$ which yields $s \in (\neg D)^\mathcal{I}$.

3. $C = D \sqcap E$. The completeness of S implies $\{D, E\} \subseteq \mathcal{L}(a)$. The induction hypothesis yields $s \in D^\mathcal{I}$ and $s \in E^\mathcal{I}$, therefore $s \in (D \sqcap E)^\mathcal{I}$.

4. $C = D \sqcup E$. The completeness of S implies $\{D, E\} \cap \mathcal{L}(a) \neq \emptyset$. By induction hypothesis it holds that $s \in D^\mathcal{I}$ or $s \in E^\mathcal{I}$, and therefore $s \in (D \sqcup E)^\mathcal{I}$.

5. $C = \exists R.D$. Since the $R\exists$ rule is not applicable, a has an R-successor c such that $D \in \mathcal{L}(c)$. By definition of $\mathcal{I}$, there is a $t = s \cdot \frac{d}{c} \in \Delta^\mathcal{I}$ such that either $c = d$ or c is blocked by d in S. Since $\mathcal{L}(c) \subseteq \mathcal{L}(d)$ in both cases, we have that $D \in \mathcal{L}(d)$. By induction, it holds that $t \in D^\mathcal{I}$. By definition of $\mathcal{I}$, we have $(s,t) \in R^\mathcal{I}$ and this implies $s \in C^\mathcal{I}$.

6. $C = \forall R.D$. Let $(s,t) \in R^\mathcal{I}$. By construction of $\mathcal{I}$, $t = s \cdot \frac{d}{c}$ such that c is an R-successor of a. Since $R\forall$ is not applicable, we have that $D \in \mathcal{L}(c)$. Since $\mathcal{L}(c) \subseteq \mathcal{L}(d)$ (as in the previous case), we have $C \in \mathcal{L}(d)$, and by induction $t \in C^\mathcal{I}$. Since this holds independently of the choice of t, we obtain $s \in C^\mathcal{I}$.

7. $C = \exists U_1, U_2.(r_1 \vee \cdots \vee r_k)$. Since C is in PNF, U_i is either a concrete feature or of the form Rg, for each $i \in \{1,2\}$. We consider only the case $U_1 = R_1 g_1$, $U_2 = R_2 g_2$, as the remaining cases are similar but easier. Since the $R\exists_c$ rule is not applicable, there exists an R_j-successor c_j of a and a g_j-successor y_j of c_j for $j = 1, 2$ such that $(y_1\ r_i\ y_2) \in \mathcal{N}$ for some $1 \leq i \leq k$. Then $((c_1, g_1)\ r_i\ (c_2, g_2)) \in \mathcal{N}^c$. Moreover, there is a $t_j = s \cdot \frac{d_j}{c_j} \in \Delta^\mathcal{I}$ such that $c_j = d_j$ or c_j is blocked by d_j, $j = 1, 2$. By definition of $R_j^\mathcal{I}$, we have that $(s, t_j) \in R_j^\mathcal{I}$, $j = 1, 2$. Moreover, since a is not blocked and c_1 and c_2 are its successors, there is a $p \in \mathsf{Hooks}$ such that t_1 and t_2 are p-companions and $\mathsf{rep}_p(t_1) = c_1$, $\mathsf{rep}_p(t_2) = c_2$. Thus, by definition of $\mathbf{N}$ we obtain $((t_1, g_1)\ r_i\ (t_2, g_2)) \in \mathbf{N}$, implying $(\tau(t_1, g_1)\ r_i\ \tau(t_2, g_2)) \in M_\mathcal{I}$. Since $g_1^\mathcal{I}(t_1) = \tau(t_1, g_1)$ and $g_2^\mathcal{I}(t_2) = \tau(t_2, g_2)$, we obtain that $s \in C^\mathcal{I}$.

8. $C = \forall U_1, U_2.(r_1 \vee \cdots \vee r_k)$. As in the previous case, we will assume that U_1 and U_2 are of the form $U_1 = Rg_1$, $U_2 = R_2g_2$. Let t_1, t_2 be such that $(s, t_j) \in R_j^{\mathcal{I}}$ and $g_j^{\mathcal{I}}(t_j)$ is defined, $j = 1, 2$. By definition of $\mathcal{I}$, we have that $t_j = s \cdot \frac{d_j}{c_j} \in \Delta^{\mathcal{I}}$ such that c_j is an R_j-successor of a, $j = 1, 2$. Moreover, there is a g_j-successor y_j of c_j for $j = 1, 2$. Since $R\forall_c$ is inapplicable, $\forall U_1, U_2.(r_1 \vee \cdots \vee r_k) \in \mathcal{L}(a)$ implies that $(y_1\ r_i\ y_2) \in \mathcal{N}$ for some $1 \leq i \leq k$. Thus, $((c_1, g_1)\ r\ (c_2, g_2)) \in \mathcal{N}^c$. Moreover, since a is unblocked there is a $p \in \mathsf{Hooks}$ such that t_1 and t_2 are p-companions and $\mathsf{rep}_p(t_1) = c_1$, $\mathsf{rep}_p(t_2) = c_2$. Thus, by definition of $\mathbf{N}$, we have that $((t_1, g_1)\ r_i\ (t_2, g_2)) \in \mathbf{N}$, which implies $(\tau(t_1, g_1)\ r_i\ \tau(t_2, g_2)) \in M_{\mathcal{I}}$. Thus, $s \in C^{\mathcal{I}}$.

This finishes the proof of Claim 4.

Since $C_0 \in \mathcal{L}(\mathsf{root})$ and $\frac{\mathsf{root}}{\mathsf{root}} \in \Delta^{\mathcal{I}}$, Claim 4 implies that $\mathcal{I}$ is a model of C_0. Finally, let us show that $\mathcal{I}$ is a model of the input TBox $\mathcal{T} = \{\top \sqsubseteq C_{\mathcal{T}}\}$. Choose an $s \in \Delta^{\mathcal{I}}$. Let $\mathsf{tail}(s) = \frac{a}{b}$. Since S is complete, $R\mathsf{tbox}$ is not applicable, and thus $C_{\mathcal{T}} \in \mathcal{L}(a)$. By Claim 4 we have that $s \in C_{\mathcal{T}}^{\mathcal{I}}$. Since this holds independently of the choice of s, we have $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ as required. ❑

Lemma 7.4.3 (Completeness). *If the input concept C_0 is satisfiable w.r.t. the input TBox $\mathcal{T}$, then the algorithm returns* satisfiable.

Proof. Let C_0 be satisfiable w.r.t. $\mathcal{T}$, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, M_{\mathcal{I}})$ a common model of C_0 and $\mathcal{T}$, and $a_0 \in \Delta^{\mathcal{I}}$ such that $a_0 \in C_0^{\mathcal{I}}$. We use $\mathcal{I}$ to guide (the non-deterministic parts of) the algorithm such that it constructs a complete and clash-free completion system. A completion system $S = (T, \mathcal{N})$ with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$ is called *$\mathcal{I}$-compatible* if there exist mappings $\pi : \mathsf{V_a} \to \Delta^{\mathcal{I}}$ and $\tau : \mathsf{V_c} \to V_{M_{\mathcal{I}}}$ (i.e., to the variables used in $M_{\mathcal{I}}$) such that

(Ca) $C \in \mathcal{L}(a) \Rightarrow \pi(a) \in C^{\mathcal{I}}$

(Cb) b is an R-successor of $a \Rightarrow (\pi(a), \pi(b)) \in R^{\mathcal{I}}$

(Cc) x is a g-successor of $a \ \Rightarrow g^{\mathcal{I}}(\pi(a)) = \tau(x)$

(Cd) $(x\, r\, y) \in \mathcal{N} \Rightarrow (\tau(x)\, r\, \tau(y)) \in M_{\mathcal{I}}$

We first show the following.

Claim 1: If a completion system S is $\mathcal{I}$-compatible and a rule R is applicable to S, then R can be applied such that an $\mathcal{I}$-compatible completion system S' is obtained.

Proof. Let $S = (T, \mathcal{N})$ be an $\mathcal{I}$-compatible completion system with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$, let π and τ be functions satisfying (Ca) to (Cd), and let R be a completion rule applicable to S. We make a case distinction according to the type of R.

$R\sqcap$ The rule is applied to a concept $C_1 \sqcap C_2 \in \mathcal{L}(a)$. By (Ca), $C_1 \sqcap C_2 \in \mathcal{L}(a)$ implies $\pi(a) \in (C_1 \sqcap C_2)^{\mathcal{I}}$ and hence $\pi(a) \in C_1^{\mathcal{I}}$ and $\pi(a) \in C_2^{\mathcal{I}}$. Since the rule adds C_1 and C_2 to $\mathcal{L}(a)$, it yields a completion system that is $\mathcal{I}$-compatible via π and τ.

$R\sqcup$ The rule is applied to $C_1 \sqcup C_2 \in \mathcal{L}(a)$. $C_1 \sqcup C_2 \in \mathcal{L}(a)$ implies $\pi(a) \in C_1^{\mathcal{I}}$ or $\pi(a) \in C_2^{\mathcal{I}}$. Since the rule adds either C_1 or C_2 to $\mathcal{L}(a)$, it can be applied such that it yields a completion system that is $\mathcal{I}$-compatible via π and τ.

$R\exists$ The rule is applied to $\exists R.C \in \mathcal{L}(a)$. By (Ca), $\pi(a) \in (\exists R.C)^{\mathcal{I}}$ and hence there exists a $d \in \Delta^{\mathcal{I}}$ such that $(\pi(a), d) \in R^{\mathcal{I}}$ and $d \in C^{\mathcal{I}}$. By definition of $R\exists$ and the "$\oplus$" operation, rule application either (i) adds a new R-successor b of a and sets $\mathcal{L}(b) = \{C\}$; or (ii) re-uses an existing R-successor, renames it to b in T and sets $\mathcal{L}(b) = \mathcal{L}(b) \cup \{C\}$. Extend π by setting $\pi(b) = d$. The resulting completion system is $\mathcal{I}$-compatible via the extended π and the original τ.

$R\forall$ The rule is applied to $\forall R.C \in \mathcal{L}(a)$ and it adds C to the label $\mathcal{L}(b)$ of an existing R-successor of a. By (Ca), $\pi(a) \in (\forall R.C)^{\mathcal{I}}$ and by (Cb), $(\pi(a), \pi(b)) \in R^{\mathcal{I}}$. Therefore, $\pi(b) \in C^{\mathcal{I}}$ and the resulting completion system is $\mathcal{I}$-compatible via π and τ.

$R\exists_c$ The rule is applied to a concept $\exists U_1, U_2.(r_1 \vee \cdots \vee r_k) \in \mathcal{L}(a)$. We assume that $U_1 = R_1 g_1$ and $U_2 = R_2 g_2$. The case where one or both of U_1, U_2 are only concrete features is similar, but easier. The rule application generates new abstract nodes b_1 and b_2 and concrete nodes x_1 and x_2 (or re-uses existing ones and renames them) such that

- b_j is an R_j-successor of a and
- x_j is a g_j-successor of b_j for $j = 1, 2$.

By (Ca), we have $\pi(a) \in (\exists U_1, U_2.(r_1 \vee \cdots \vee r_k))^{\mathcal{I}}$. Thus, there exist $d_1, d_2 \in \Delta^{\mathcal{I}}$, $v_1, v_2 \in V_{M_{\mathcal{I}}}$ and an i with $1 \leq i \leq k$ such that

- $(\pi(a), d_j) \in R_j^{\mathcal{I}}$,
- $g_j^{\mathcal{I}}(d_j) = v_j$ for $j = 1, 2$, and
- $(v_1\, r_i\, v_2) \in M_{\mathcal{I}}$.

Thus, the rule can be guided such that it adds $(x_1\, r_i\, x_2)$ to $\mathcal{N}$. Extend π by setting $\pi(b_j) := d_j$, and extend τ by setting $\tau(x_j) := v_j$ for $j = 1, 2$. It is easily seen that the resulting completion system is $\mathcal{I}$-compatible via the extended π and τ.

$R\forall_c$ The rule is applied to an abstract node a with $\forall U_1, U_2.\ (r_1 \vee \cdots \vee r_k) \in \mathcal{L}(a)$ such that there are $x_1, x_2 \in \mathsf{V_c}$ with x_i a U_i-successor of a, for $i = 1, 2$. By (Ca), $\pi(a) \in (\forall U_1, U_2.(r_1 \vee \cdots \vee r_k))^{\mathcal{I}}$. By (Cb) and (Cc), we have $(\pi(a), \tau(x_1)) \in U_1^{\mathcal{I}}$ and $(\pi(a), \tau(x_2)) \in U_2^{\mathcal{I}}$. By the semantics, it follows that there is an i with $1 \leq i \leq k$ such that $(\tau(x_1)\, r_i\, \tau(x_2)) \in M_{\mathcal{I}}$. The application rule can be guided such that it adds $(x_1\ r_i\ x_2)$ to $\mathcal{N}$. Thus, the resulting completion system is $\mathcal{I}$-compatible via π and τ.

Rnet The rule is applied to an abstract node a such that a is potentially blocked by an abstract node b and $\mathcal{N}(a)$ is not complete (the symmetric case is analogous). The rule application guesses a completion $\mathcal{N}'$ of $\mathcal{N}(a)$, and sets $\mathcal{N} := \mathcal{N} \cup \mathcal{N}'$. Define

$$\begin{aligned} \mathcal{N}' \;:=\; \{(x\, r\, y) \mid\ & x \text{ is a } g\text{-successor of } a, \\ & y \text{ is a } g'\text{-successor of } a, \text{ and } (\tau(x)\, r\, \tau(y)) \in M_{\mathcal{I}}\}. \end{aligned}$$

By definition of $\mathcal{N}(a)$, we have $V_{\mathcal{N}(a)} = V_{\mathcal{N}'}$. By (Cd), we have $\mathcal{N}(a) \subseteq \mathcal{N}'$. Since $M_{\mathcal{I}}$ is complete, $\mathcal{N}'$ is complete. Finally, τ witnesses that $M_{\mathcal{I}}$ is a model of $\mathcal{N}'$, and thus $\mathcal{N}'$ is satisfiable. It follows that $\mathcal{N}'$ is a completion of $\mathcal{N}(a)$. Apply Rnet such that $\mathcal{N}'$ is guessed. Then, the resulting completion system is $\mathcal{I}$-compatible via π and τ.

*R*tbox The rule application adds $C_\mathcal{T}$ to $\mathcal{L}(a)$, for some $a \in \mathsf{V_a}$. Since $\mathcal{I}$ is a model of $\mathcal{T}$, we have $\pi(a) \in C_\mathcal{T}^\mathcal{I}$. Thus, the resulting completion system is $\mathcal{I}$-compatible via π and τ.

We now show that $\mathcal{I}$-compatibility implies clash-freeness.

Claim 2: Every $\mathcal{I}$-compatible completion system is clash-free.

Proof. Let $S = (T, \mathcal{N})$ be an $\mathcal{I}$-compatible completion system with $T = (\mathsf{V_a}, \mathsf{V_c}, E, \mathcal{L})$. Consider the two kinds of a clash:

- Due to (Ca), a clash of the form $\{A, \neg A\} \in \mathcal{L}(a)$ contradicts the semantics.
- Property (Cd) implies that $M_\mathcal{I}$ is a model of $\mathcal{N}$. Thus, $\mathcal{N}$ is satisfiable.

We can now describe the "guidance" of the tableau algorithm by the model $\mathcal{I}$: we ensure that, at all times, the considered completion systems are $\mathcal{I}$-compatible. This obviously holds for the initial completion system. By Claim 1, we can guide the rule applications such that only $\mathcal{I}$-compatible completion systems are obtained. By Lemma 7.4.1, the algorithm always terminates, hence also when guided in this way. Since, by Claim 2, we will not find a clash, the algorithm returns satisfiable. ❑

As an immediate consequence of Lemmas 7.4.1, 7.4.2 and 7.4.3, we get the following theorem:

Theorem 7.4.4. *If $\mathcal{C}$ is an ω-admissible constraint system, the tableau algorithm decides satisfiability of $\mathcal{ALC}(\mathcal{C})$ concepts w.r.t. general TBoxes.*

A close inspection of our algorithm shows that it runs in 2-NExpTime if $\mathcal{C}$-satisfiability is in NP. We conjecture that, by mixing the techniques from the previous section with those from [Lut04a, Lut02a], it is possible to prove ExpTime-completeness of satisfiability in $\mathcal{ALC}(\mathcal{C})$ provided that satisfiability in $\mathcal{C}$ can be decided in ExpTime.

7.5 Practicability

With Theorem 7.4.4, we have achieved the main aim of this chapter: providing a general decidability result for description logics with both general TBoxes and concrete domains. Our second aim is to identify an algorithm that is more practicable than the existing approaches based on automata [Lut04a, Lut02a], i.e., that can be implemented such that an acceptable runtime behaviour is observed on realistic inputs. Since we have not yet implemented our algorithm,[4] an empirical evaluation is out of reach. In the following, we discuss the practicability on a general level.

Regarding an efficient implementation, the main difficulties of our algorithm compared with successfully implemented tableau algorithms such as the ones in [SSS91, HS99] are the following:

- Our algorithm requires satisfiability checks of the network $\mathcal{N}$ constructed as part of the completion system. The problem is that this check involves the *whole* network $\mathcal{N}$ rather than only small parts of it. In practice, the constructed completion systems (and associated networks) are often too large to be considered as a whole.

[4] This is a non-trivial task since a large number of sophisticated optimization techniques is required, c.f. [HPS99].

- The rules $R\exists_c$, $R\forall_c$, and $R\mathsf{net}$ introduce additional non-determinism. In implementations, this non-determinism induces backtracking.

It is possible that these difficulties can be overcome by developing appropriate heuristics and optimization techniques. However, there is also an easy way around them. In the following, we argue that there is a fragment of our language that still provides interesting expressive power and in which the implementation difficulties discussed above are non-existent.

The fragment of $\mathcal{ALC}(\mathcal{C})$ that we consider is obtained by making the following assumptions:

- There is only a single concrete feature g. Note that this is acceptable with constraint systems such as $\mathsf{RCC8}_{\mathbb{R}^2}$ and $\mathsf{Allen}_{\mathbb{R}}$, where g could be $\mathsf{has\text{-}extension}$ and $\mathsf{has\text{-}lifetime}$, respectively.

- There are no paths of length greater than 2, i.e., Clause 3 is eliminated from Definition 7.2.1. This is necessary since we need to introduce additional concrete features to establish path normal form if Clause 3 is present. We believe that paths of length three or more are only needed in exceptional cases, anyway.

- There exists a unique equality predicate eq in $\mathcal{C}$, i.e., for all models $N \in \mathfrak{M}$ and all $v \in V_N$, we have $(v\ \mathsf{eq}\ v) \in N$.

Going to this fragment of $\mathcal{ALC}(\mathcal{C})$ allows the following simplification of our tableau algorithm.

1. The non-deterministic $R\mathsf{net}$ rule can simply be dropped because, for each abstract node a, the network $\mathcal{N}(a)$ is either empty or consists of a single node that is related to itself via eq. Thus, every potential blocking situation is an actual blocking situation.

2. We can localize the satisfiability check of the network $\mathcal{N}$ as follows. For $a \in \mathsf{V_a}$, let $\widehat{\mathcal{N}}(a)$ denote the restriction of $\mathcal{N}$ to the g-successor of a and the g-successors of all abstract successors of a. Instead of checking the whole network $\mathcal{N}$ for satisfiability, we separately check, for each $a \in \mathsf{V_a}$, satisfiability of $\widehat{\mathcal{N}}(a)$. It can be seen as follows that this is equivalent to a global check: first, $\mathcal{C}$ has the patchwork property. Second, due to the fact that there is only a single concrete feature g, the networks $\widehat{\mathcal{N}}(a)$ overlap at single nodes only. Due to the presence of the equality predicate eq, the overlapping part of two such networks is thus complete. Finally, it is easy to see that the patchwork property implies a more general version of itself where only the overlapping part of the two involved networks is complete, but the networks themselves are not.

Hence, the only difficulty that remains is the non-determinism of the rules $R\exists_c$ and $R\forall_c$. However, we believe that this non-determinism is not too difficult to deal with. To see this, observe that the non-deterministic choices made by these rules have only a very local impact: they only influence the outcome of the satisfiability check of the relevant local network $\widehat{\mathcal{N}}(a)$. Therefore, it does not seem necessary to implement a complex backtracking/backjumping machinery. If the concrete domain reasoner used for deciding $\mathcal{C}$-satisfiability supports disjunctions, it is even possible to push the non-determinism out of the tableau algorithm into the reasoner for $\mathcal{C}$-satisfiability. Roughly, one would need to allow disjunctions in the constraint network $\mathcal{N}$ and pass these on to the reasoner for $\mathcal{C}$.

Chapter 8

Conclusion

8.1 Description Logic Actions

8.1.1 Summary

In this thesis, we have presented foundational work on integrating description logics with action formalisms.

We have introduced action formalisms based on description logics and have investigated the standard reasoning problems executability and projection, as well as the plan existence problem in this context. In these action formalisms, ABox assertions are used to describe states of the world as well as pre- and post-conditions of actions. TBoxes have a different role in the two action formalisms $\mathfrak{A}_1$ and $\mathfrak{A}_2$ that we have developed. While in the action formalism $\mathfrak{A}_1$, acyclic TBoxes are used to define abbreviations for complex concepts, in the action formalism $\mathfrak{A}_2$ general TBoxes serve as state constraints. By forbidding complex or defined concepts in post-conditions of $\mathfrak{A}_1$-actions we ensure that changes induced by these actions may occur only at an atomic level. We have shown that instantiations of $\mathfrak{A}_1$ with expressive DLs can be viewed as fragments of the standard Situation Calculus, and thus inherit its solution to the frame problem. Possible ramifications of executions of $\mathfrak{A}_1$-actions are specified by the accompanying acyclic TBox: when interpretations of atomic concepts and roles change, interpretations of concepts defined in the TBox need to change accordingly. The action formalism $\mathfrak{A}_2$, besides supporting general TBoxes, allows for complex concepts in action post-conditions; thus raising the frame problem and general TBox ramification problem. Since the attempt to automatically solve these problems turns out to lead to semantic and computational problems, we have adopted an alternative approach: action designers are supposed to fine-tune ramifications of $\mathfrak{A}_2$-actions by means of complex occlusion patterns. Occlusions are used to describe in which part of the domain interpretations of concept and role names may change and in which they may not. Besides projection and executability, consistency of $\mathfrak{A}_2$-actions has emerged as an important (non-trivial) reasoning task.

One of the main technical results of this thesis is that standard problems in reasoning about action (projection, executability) become decidable if one restricts the logic for describing pre- and post-conditions as well as the state of the world to certain decidable description logics $\mathcal{L}$. Hence, we give a positive answer to the question of whether there exist decidable action formalisms with expressivity beyond propositional logic. For a propositionally closed $\mathcal{L}$, the complexity of projection and executability is determined by the complexity of standard DL reasoning in $\mathcal{L}$ extended with nominals. This means that the complexity of these infer-

Logic	Projection/Executability
$\mathcal{EL}$ and $\mathcal{EL}^{(\neg)}$ without TBoxes	co-NP-complete
between $\mathcal{EL}$ and $\mathcal{ALCQO}$	PSPACE-complete
$\mathcal{ALCI}$, $\mathcal{ALCIO}$	EXPTIME-complete
$\mathcal{ALCQI}$, $\mathcal{ALCQIO}$	co-NEXPTIME-complete

Figure 8.1: Complexity of projection and executability in $\mathfrak{A}_1$

Logic	Projection/Executability	Weak Consistency
$\mathcal{ALC}$, $\mathcal{ALCO}$, $\mathcal{ALCI}$, $\mathcal{ALCIO}$ $\mathcal{ALCQ}$, $\mathcal{ALCQO}$	EXPTIME-complete	EXPTIME-complete
$\mathcal{ALCQI}$, $\mathcal{ALCQIO}$	co-NEXPTIME-complete	NEXPTIME-complete

Figure 8.2: Complexity of projection, executability, and weak consistency in $\mathfrak{A}_2$

ences in $\mathcal{ALC}$ and its extensions is ranging from PSPACE-complete to co-NEXPTIME-complete for $\mathfrak{A}_1$-actions in the presence of acyclic TBoxes. It is EXPTIME-complete or co-NEXPTIME-complete for $\mathfrak{A}_2$-actions in the presence of general TBoxes. Our results have shown that, even in the lightweight description logic $\mathcal{EL}$, projection is not tractable. Even if we disallow TBoxes, projection in $\mathcal{EL}$ is co-NP-complete, while in presence of acyclic TBoxes it is PSPACE-hard and thus not easier than in $\mathcal{ALC}$. One negative result we have obtained is the undecidability of strong consistency of $\mathfrak{A}_2$-actions. However, weak consistency of $\mathfrak{A}_2$-actions, still very useful for detecting errors in action design, is polynomially reducible to projection and thus decidable. Complexity results for $\mathfrak{A}_1$ and $\mathfrak{A}_2$ are summarized in Figures 8.1 and 8.2.

Most of the decidability (and complexity) results for projection and executability in this thesis have been obtained by polynomial reductions of these inferences to ABox consequence. This shows that reasoning about actions in expressive description logics can be delegated to standard DL reasoners.

We have also investigated the complexity of the plan existence problem for planning with $\mathfrak{A}_1$-actions in the presence of acyclic TBoxes. We have shown that the plan existence problem (PLANEX) is decidable and of the same computational complexity as projection in the logics between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ if operators have only unconditional post-conditions, thus ranging from PSPACE-complete to co-NEXPTIME-complete. If operators have conditional post-conditions, PLANEX is shown to be in 2-EXPSPACE. Since the only known lower complexity bound is inherited from propositional logic (EXPSPACE-hard), we leave the exact computational complexity of conditional PLANEX as an open problem. Finally, we have shown that the complexity results for propositional planning carry over to planning in the lightweight description logic $\mathcal{EL}$.

8.1.2 Future Work

The foundational results we have presented provide promising starting points for possible implementations and extensions. The first natural step would be to implement practicable variants of the algorithms for projection and executability defined in this thesis, in particular the reduction of the projection problem to the ABox consequence problem from Section 3.2.1. Considering implementations of DL planners, it would be reasonable to start with lightweight DLs, such as $\mathcal{EL}$ or $\mathcal{EL}^{(\neg)}$, for which projection is in co-NP, and to try to adapt some of the known techniques for propositional conformant planning.

Regarding possible extensions, there are several ways to continue.

- For different applications, it might be useful to have *concrete datatypes* in DLs underlying an action formalism. We are convinced that concrete datatypes can be treated without major problems within our action formalisms.

- Instead of focusing only on sequences of atomic actions, it would be interesting to investigate more *complex operators*, such as PDL- or GOLOG-like programs over atomic actions. Such programs would be very useful in the context of services in the Semantic Web. Moreover, for the same application it would be of great interest to formalize and investigate *sensing actions* – they naturally correspond to "information-gathering" services such as Google.

- The action formalisms introduced in this thesis are general-purpose formalisms, developed without taking into account features of *specific application domains*. We hope that a careful investigation of TBoxes in use, e.g., in the medical domain, could give us more insight into the nature of action ramifications that can be expected. We also suspect that various additional constructs in action definitions may be desired in medical and other applications. One such example are *universally quantified post-conditions* of the form $\forall x.\varphi(x)/\psi(x)$.

- We have shown that conditional post-conditions make planning harder due to the uncertainty about which post-conditions are triggered. For applications, it would make sense to adopt a more *pragmatic semantics* of conditional post-conditions which would enable a compact representation of states of the search space by means of the initial ABox and updates. Intuitively, post-conditions φ/ψ would be interpreted as follows: if it can be proved that φ holds in the current state then ψ holds in the successor state. This semantics would be along the lines with operational ADL semantics.

- Finally, instead of using an approach similar to regression in order to decide the projection problem, one could also try to apply *progression*, i.e., to calculate an updated ABox that has as its models all the successors of the models of the original ABox. This approach has already been followed in [LLMW06c] and showed that updated ABoxes may be exponentially large in the size of the original ABox and the update. For applications it would be interesting to investigate approximated or projective ABox updates of polynomial size.

8.2 Description Logics with Concrete Domains

The second part of this thesis was dedicated to concrete domains. We have identified a general property of concrete domains, called ω-admissibility, that is sufficient for proving decidability of DLs with both concrete domains and general TBoxes. For defining ω-admissibility, we have concentrated on a particular kind of concrete domains: constraint systems – concrete domains that only have binary predicates interpreted as JEPD relations. We have exhibited some useful constraint systems with this property, most notably a spatial one based on the RCC8 relations and a temporal one based on Allen relations.

We have also exhibited the first tableau algorithm for DLs with concrete domains and general TBoxes in which the concrete domain constructors are not limited to concrete features.

Although we have focused on $\mathcal{ALC}(\mathcal{C})$, a combination of $\mathcal{ALC}$ with a constraint system $\mathcal{C}$, various language extensions such as transitive roles and number restrictions should also be possible in a straightforward way. The algorithm has some aspects likely to have a negative impact on practicability unless addressed by dedicated optimization techniques. However, we have identified a useful fragment of $\mathcal{ALC}(\mathcal{C})$ in which these impairing aspects of the algorithm can be avoided, thus ensuring that it can be easily integrated into existing DL reasoners such as FaCT++ and RacerPro.

While we have proved that requiring a concrete domain $\mathcal{C}$ to be an ω-admissible constraint system is a sufficient condition for decidability of $\mathcal{ALC}$ combined with $\mathcal{C}$, it is not clear whether it is also a necessary one. We conjecture that ω-admissibility is not a necessary condition. One obvious relaxation would be to admit certain unary predicates in $\mathcal{C}$, as in [Lut02a].

Bibliography

[ABM99] C. Areces, P. Blackburn, and M. Marx. A Road-map on Complexity for Hybrid Logics. In J. Flum and M. Rodríguez-Artalejo, eds., *Computer Science Logic*, number 1683 in Lecture Notes in Computer Science, pp. 307–321. Springer-Verlag, 1999.

[AF98] A. Artale and E. Franconi. A Temporal Description Logic for Reasoning about Actions and Plans. *Journal of Artificial Intelligence Research (JAIR)*, 9:463–506, 1998.

[All83] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[Baa03] F. Baader. The instance problem and the most specific concept in the description logic $\mathcal{EL}$ w.r.t. terminological cycles with descriptive semantics. In *In Proceedings of 26th Annual German Conference on Artificial Intelligence (KI2003)*, volume 2821 of *Lecture Notes in Artificial Intelligence*, pp. 64–78. Springer, 2003.

[BBL05] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ Envelope. In *In Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-2005)*, pp. 364–369. Morgan-Kaufmann Publishers, Edinburgh, UK, 2005.

[BC02] P. Balbiani and J.-F. Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *Frontiers of Combining Systems (FroCoS 2002)*, number 2309 in LNAI, pp. 162–176. Springer, 2002.

[BC07] C. J. Baker and K.-H. Cheung, eds. *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences*. Springer, 2007.

[BCM+03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Ben97] B. Bennett. Modal Logics for Qualitative Spatial Reasoning. *Journal of the Interest Group in Pure and Applied Logic*, 4(1), 1997.

[Ber66] R. Berger. The Undecidability of the Dominoe Problem. *Memoirs of the American Mathematical Society*, 66, 1966.

[BF97] A. L. Blum and M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):281–300, 1997.

[BH91] F. Baader and P. Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence, IJCAI-91*, pp. 452–457. Sydney (Australia), 1991.

[BHS03] F. Baader, I. Horrocks, and U. Sattler. Description Logics as Ontology Languages for the Semantic Web. In D. Hutter and W. Stephan, eds., *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2003.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5), 2001.

[BLM+05a] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. A Description Logic Based Approach to Reasoning about Web Services. In *Proceedings of the WWW 2005 Workshop on Web Service Semantics (WSS2005)*. Chiba City, Japan, 2005.

[BLM+05b] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating Description Logics and Action Formalisms: First Results. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pp. 572–577. Pittsburgh, USA, 2005.

[BLM+05c] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating Description Logics and Action Formalisms: First Results. In *Proceedings of the 2005 International Workshop on Description Logics (DL2005)*, number 147 in CEUR-WS. 2005.

[BLM+05d] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating Description Logics and Action Formalisms for Reasoning about Web Services. LTCS-Report 05-02, TU Dresden, Germany, 2005. See http://lat.inf.tu-dresden.de/research/reports.html.

[BLS06] F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—A Polynomial-time Reasoner for Life Science Ontologies. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pp. 287–291. Springer-Verlag, 2006.

[Bor96] A. Borgida. On the relative Expressiveness of Description Logics and Predicate Logics. *Artificial Intelligence*, 82(1 - 2):353–367, 1996.

[Bra04] S. Brandt. Polynomial Time Reasoning in a Description Logic with Existential Restrictions, GCI Axioms, and—What Else? In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-2004)*, pp. 298–302. IOS Press, 2004.

[BS85] R. J. Brachman and J. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9(2):171–216, 1985.

[BSL06] S. Bechhofer, R. D. Stevens, and P. W. Lord. GOHSE: Ontology driven linking of biology resources. *Journal of Web Semantics*, 4(3):155–163, 2006.

[BvHH+04] S. Bechhofer, F. van Hamerlen, J. Hendler, I. Horrocks, D. L. McGuinness, P. Patel-Schneider, and L. Stein. OWL Web Ontology Language reference. W3C Recommendation, 2004. Http://www.w3.org/TR/owl-ref/.

[Byl94] T. Bylander. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

[CELN07] J. Claßen, P. Eyerich, G. Lakemeyer, and B. Nebel. Towards an Integration of Golog and Planning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*. AAAI Press, 2007.

[CGL+05] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable Description Logics for Ontologies. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pp. 602–607. 2005.

[CGLR07] D. Calvanese, G. D. Giacomo, M. Lenzerini, and R. Rosati. Actions and Programs over Description Logic Ontologies. In *Proceedings of The Twentieth Internationa Workshop on Description Logics (DL-2007)*. 2007.

[CGT03] C. Castellini, E. Giunchiglia, and A. Tacchella. SAT-based planning in complex domains: Concurrency, constraints and nondeterminism. *Artificial Intelligence*, 147(1-2):85–117, 2003.

[CL06] J. Claßen and G. Lakemeyer. A Semantics for ADL as Progression in the Situation Calculus. In *Proceedings of the 11th Workshop on Nonmonotonic Reasoning*. 2006.

[CLN98] D. Calvanese, M. Lenzerini, and D. Nardi. Description Logics for Conceptual Data Modeling. In J. Chomicki and G. Saake, eds., *Logics for Databases and Information Systems*, pp. 229–263. Kluwer Academic Publisher, 1998.

[CR00] A. Cimatti and M. Roveri. Conformant Planning via Symbolic Model Checking. *Journal of Artificial Intelligence Research*, 13:305–338, 2000.

[DL96] P. T. Devanbu and D. J. Litman. Taxonomic Plan Reasoning. *Artificial Intelligence*, 84(1-2):1–35, 1996.

[DT07] C. Drescher and M. Thielscher. Integrating Action Calculi and Description Logics. In *Proceedings of 30th Annual German Conference on AI (KI 2007)*, pp. 68–83. 2007.

[EF91] M. J. Egenhofer and R. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991.

[EG92] T. Eiter and G. Gottlob. On the Complexity of Propositional Knowledge Base Revision, Updates, and Counterfactuals. *Artificial Intelligence*, 57(2-3):227–270, 1992.

[ENS95] K. Erol, D. S. Nau, and V. S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence*, 76(1-2):75–88, 1995.

[FG00] P. Ferraris and E. Giunchiglia. Planning as Satisfiability in Nondeterministic Domains. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI'2000)*, pp. 748–753. 2000.

[FL03] M. Fox and D. Long. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research (JAIR)*, 20:61–124, 2003.

[FN71] R. E. Fikes and N. J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[Fra96] A. Frank. Qualitative Spatial Reasoning: Cardinal Directions as an Example. *International Journal of Geographical Information Systems*, 10(3):269–290, 1996.

[GINR96] G. D. Giacomo, L. Iocchi, D. Nardi, and R. Rosati. Moving a Robot: the KR&R Approach at Work. In *Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning (KR-96)*, pp. 198–209. 1996.

[GINR97] G. D. Giacomo, L. Iocchi, D. Nardi, and R. Rosati. Planning with Sensing for a Mobile Robot. In *Preprints of the Fourth European Conference on Planning*, pp. 156–168. 1997.

[GKWZ03] D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Number 148 in Studies in Logic and the Foundations of Mathematics. Elsevier, 2003.

[GLPR06] G. D. Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the Update of Description Logic Ontologies at the Instance Level. In *Proceedings of The Twenty-First National Conference on Artificial Intelligence (AAAI'06)*. Boston, Massachusetts, USA, 2006.

[GLPR07] G. D. Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the Approximation of Instance Level Update and Erasure in Description Logics. In *Proceedings of The Twenty-Second National Conference on Artificial Intelligence (AAAI'07)*, pp. 403–408. Vancouver, British Columbia, Canada, 2007.

[GS07] Y. Gu and M. Soutchanski. Decidable Reasoning in a Modified Situation Calculus. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pp. 1891–1897. Hyderabad, India, 2007.

[GZB06] C. Golbreich, S. Zhang, and O. Bodenreider. The foundational model of anatomy in OWL: Experience and perspectives. *Journal of Web Semantics*, 4(3):181–195, 2006.

[HB06] J. Hoffmann and R. I. Brafman. Conformant planning via heuristic forward search: a new approach. *Artificial Intelligence*, 170(6):507–541, 2006.

[Her96] A. Herzig. The PMA revisited. In *Proceedings of the 5th International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*. Morgan Kaufmann, 1996.

[HJ99] P. Haslum and P. Jonsson. Some Results on the Complexity of Planning with Incomplete Information. In *Proceedings of 5th European Conference on Planning ECP'99*, pp. 308–318. 1999.

[HKNP92] J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. RAT: Representation of Actions using Terminological Logics. Technical report, DFKI, 1992.

[HLM98] V. Haarslev, C. Lutz, and R. Möller. Foundations of Spatioterminological Reasoning with Description Logics. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)*, pp. 112–124. Morgan-Kaufmann Publishers, 1998.

[HLM99] V. Haarslev, C. Lutz, and R. Möller. A Description Logic with Concrete Domains and Role-forming Predicates. *Journal of Logic and Computation*, 9(3), 1999.

[HM01a] V. Haarslev and R. Möller. High Performance Reasoning with Very Large Knowledge Bases: A Practical Case Study. In B. Nebel, ed., *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pp. 161–166. Morgan-Kaufmann, 2001.

[HM01b] V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkow, eds., *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR'01)*, number 2083 in Lecture Notes in Artifical Intelligence, pp. 701–705. Springer-Verlag, 2001.

[HMW01] V. Haarslev, R. Möller, and M. Wessel. The Description Logic $\mathcal{ALCNH}_{R^+}$ Extended with Concrete Domains: A Practically Motivated Approach. In *Proceedings of the First International Joint Conference on Automated Reasoning IJCAR'01*, pp. 29–44. 2001.

[HN01] J. Hoffmann and B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research (JAIR)*, 14:253–302, 2001.

[HPS99] I. Horrocks and P. F. Patel-Schneider. Optimizing Description Logic Subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.

[HPSvH03] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[HS90] S. Hölldobler and J. Schneeberger. A new deductive approach to planning. *New Generation Computing*, 8(3):225–244, 1990.

[HS99] I. Horrocks and U. Sattler. A Description Logic with Transitive and Inverse Roles and Role Hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.

[HS01] I. Horrocks and U. Sattler. Ontology Reasoning in the SHOQ(D) Description Logic. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*. 2001.

[HS05] I. Horrocks and U. Sattler. A Tableaux Decision Procedure for SHOIQ. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pp. 448–453. 2005.

[HST99] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Expressive Description Logics. In H. Ganzinger, D. McAllester, and A. Voronkov, eds., *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pp. 161–180. Springer-Verlag, 1999.

[HTR06] M. Horridge, D. Tsarkov, and T. Redmond. Supporting Early Adoption of OWL 1.1 with Protege-OWL and FaCT++. In *Proceedings of the OWL Experiences And Directions Workshop*. Athens, Georgia, USA, 2006.

[KS92] H. A. Kautz and B. Selman. Planning as Satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, pp. 359–363. 1992.

[KS96] H. Kautz and B. Selman. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *Proceedings of In Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-96)*, pp. 1194–1201. Menlo Park, California, 1996.

[KS98] H. Kautz and B. Selman. BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving. In *In Working notes of the AIPS98 Workshop on Planning as Combinatorial Search*. 1998.

[Lif86] V. Lifschitz. On the semantics of STRIPS. In M. P. Georgeff and A. L. Lansky, eds., *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, pp. 1–9. Morgan Kaufmann, 1986.

[Lif90] V. Lifschitz. Frames in the Space of Situations. *Artificial Intelligence Journal*, 46:365–376, 1990.

[Lif94] V. Lifschitz. Circumscription. In D. Gabbay, C. J. Hogger, and J. A. Robinson, eds., *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*, pp. 298–352. Oxford University Press, 1994.

[Lin95] F. Lin. Embracing Causality in Specifying the Indirect Effects of Actions. In C. S. Mellish, ed., *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 1985–1991. Morgan Kaufmann, Montreal, Canada, 1995.

[Lin96] F. Lin. Embracing Causality in Specifying the Indeterminate Effects of Actions. In B. Clancey and D. Weld, eds., *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-96)*, pp. 670–676. MIT Press, Portland, OR, 1996.

[LLM08] H. Liu, C. Lutz, and M. Milicic. The Projection Problem for $\mathcal{EL}$ Actions. In *Proceedings of the 2008 International Workshop on Description Logics (DL2008)*, CEUR-WS. 2008.

[LLMW06a] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Description Logic Actions with general TBoxes: a Pragmatic Approach. In *Proceedings of the 2006 International Workshop on Description Logics (DL2006)*, number 189 in CEUR-WS. 2006.

[LLMW06b] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Reasoning about Actions using Description Logics with general TBoxes. In *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA 2006)*, volume 4160 of *Lecture Notes in Artificial Intelligence*, pp. 266–279. Springer-Verlag, 2006.

[LLMW06c] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating Description Logic ABoxes. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR2006)*, pp. 46–56. AAAI Press, 2006.

[LM05a] C. Lutz and M. Milicic. A Tableau Algorithm for Description Logics with Concrete Domains and GCIs. In *Proceedings of the 14th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods TABLEAUX 2005*, number 3702 in LNAI, pp. 201–216. Springer, Koblenz, Germany, 2005.

[LM05b] C. Lutz and M. Milicic. A Tableau Algorithm for DLs with Concrete Domains and GCIs. In *Proceedings of the 2005 International Workshop on Description Logics (DL2005)*, number 147 in CEUR-WS. 2005.

[LM07] C. Lutz and M. Milicic. A Tableau Algorithm for Description Logics with Concrete Domains and General TBoxes. *Journal of Automated Reasoning. Special Issue on on Automated Reasoning with Analytic Tableaux and Related Methods*, 38(1-3):227–259, 2007.

[LPR88] H. Levesque, F. Pirri, and R. Reiter. Foundations for the Situation Calculus. *Linköping Electronic Articles in Computer and Information Science*, 3(18), 1988.

[LR94] F. Lin and R. Reiter. State constraints revisited. *Journal of Logic and Computation*, 5:655–678, 1994.

[LR97] F. Lin and R. Reiter. How to progress a database. *Artificial Intelligence*, 92(1-2):131–167, 1997.

[LRL+97] H. J. Levesque, R. Reiter, L. Lesperance, F. Lin, and R. B. Scherl. GOLOG: a Logic Programming Language for Dynamic Domains. *Journal of Logic Programming*, 31(1-3):59–83, 1997.

[Lut02a] C. Lutz. Adding Numbers to the $\mathcal{SHIQ}$ Description Logic—First Results. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, pp. 191–202. Morgan Kaufman, 2002.

[Lut02b] C. Lutz. PSPACE Reasoning with the Description Logic $\mathcal{ALCF}(\mathcal{D})$. *Logic Journal of the IGPL*, 10(5):535–568, 2002.

[Lut02c] C. Lutz. Reasoning about Entity Relationship Diagrams with Complex Attribute Dependencies. In *Proceedings of the International Workshop in Description Logics 2002 (DL2002)*, number 53 in CEUR-WS (http://ceur-ws.org/), pp. 185–194. 2002.

[Lut03] C. Lutz. Description Logics with Concrete Domains—A Survey. In P. Balbiani, N.-Y. Suzuki, F. Wolter, and M. Zakharyaschev, eds., *Advances in Modal Logics Volume 4*, pp. 265–296. King's College Publications, 2003.

[Lut04a] C. Lutz. Combining Interval-based Temporal Reasoning with General TBoxes. *Artificial Intelligence*, 152(2):235–274, 2004.

[Lut04b] C. Lutz. NExpTime-complete Description Logics with Concrete Domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004.

[LW06] C. Lutz and F. Wolter. Modal Logics of Topological Relations. *Logical Methods in Computer Science*, 2(2), 2006.

[MBM+07] D. Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D. McGuinness, E. Sirin, and N. Srinivasan. Bringing Semantics to Web Services with OWL-S. *World Wide Web Journal*, 10(3):243–277, 2007. Special Issue: Recent Advances in Web Services.

[McC63] J. McCarthy. Situations, actions and causal laws. Technical report, Stanford University, 1963. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pages 410-417.

[McI00] S. McIlraith. An axiomatic solution to the ramification problem (sometimes). *Artificial Intelligence Journal*, 116(1–2):87–121, 2000.

[MH69] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[Mil07] M. Milicic. Complexity of Planning in Action Formalisms Based on Description Logics. In *Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2007)*, Lecture Notes in Artificial Intelligence, pp. 408–422. Springer-Verlag, 2007.

[Min75] M. Minsky. A framework for representating knowledge. In P. H. Winston, ed., *The Psychology of Computer Vision*, pp. 211–277. McGraw-Hill, New York, USA, 1975.

[MS02] S. McIlraith and T. Son. Adapting Golog for Composition of Semantic Web Services. In *Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002)*, pp. 482–493. Toulouse, France, 2002.

[MSZ01] S. McIlraith, T. Son, and H. Zeng. Semantic Web Services. *IEEE Intelligent Systems. Special Issue on the Semantic Web*, 16(2):46–53, 2001.

[NB95] B. Nebel and H.-J. Bürckert. Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra. *Journal of the ACM*, 42(1):43–66, 1995.

[Neb91] B. Nebel. Terminological Cycles: Semantics and Computational Properties. In J. F. Sowa, ed., *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pp. 331–361. Morgan Kaufmann, 1991.

[Neb00] B. Nebel. On the Compilability and Expressive Power of Propositional Planning Formalisms. *Journal of Artificial Intelligence Research (JAIR)*, 12:271–315, 2000.

[NM02] S. Narayanan and S. McIlraith. Simulation, Verification and Automated Composition of Web Services. In *Proceedings of the Eleventh International World Wide Web Conference (WWW-11)*, pp. 77–88. Honolulu, Hawaii, USA, 2002.

[Ped89] E. P. D. Pednault. ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus. In *Proceedings of the First International Conference on Knowledge Representation and Reasoning (KR'89)*, pp. 324–332. 1989.

[Ped94] E. P. D. Pednault. ADL and the State-Transition Model of Action. *Journal of Logic and Computation*, 4(5):467–512, 1994.

[PG06] H. Palacios and H. Geffner. Compiling Uncertainty Away: Solving Conformant Planning Problems using a Classical Planner (Sometimes). In *Proceedings of The Twenty-First National Conference on Artificial Intelligence (AAAI'06)*. 2006.

[PH05] I. Pratt-Hartmann. Complexity of the Two-Variable Fragment with Counting Quantifiers. *Journal of Logic, Language and Information*, 14(3):369–395, 2005.

[Pra79] V. R. Pratt. Models of Program Logics. In *Proceedings of the Twentieth Annual Symposium on Foundations of Computer Science*. San Juan, Puerto Rico, 1979.

[PST00] L. Pacholski, W. Szwast, and L. Tendera. Complexity Results for First-Order Two-Variable Logic with Counting. *SIAM Journal on Computing*, 29(4):1083–1117, 2000.

[Qui68] M. R. Quillian. Semantic Memory. In M. Minsky, ed., *Semantic Information Processing*, pp. 227–270. MIT Press, Cambridge, MA, USA, 1968.

[RCC92] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pp. 165–176. Morgan Kaufman, 1992.

[Rei91] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, ed., *Artificial Intelligence and Mathematical Theory of Computation*, pp. 359–380. Academic Press, 1991.

[Rei01] R. Reiter. *Knowledge in Action*. MIT Press, 2001.

[RH97] A. Rector and I. Horrocks. Experience Building a Large, Re-usable Medical Ontology using a Description Logic with Transitivity and Concept Inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI-97)*. AAAI Press, 1997.

[Rin04] J. Rintanen. Complexity of Planning with Partial Observability. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pp. 345–354. 2004.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995.

[RN99] J. Renz and B. Nebel. On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus. *Artificial Intelligence*, 108(1–2):69–123, 1999.

[San94] E. Sandewall. *Features and Fluents*. Oxford University Press, 1994.

[Sav70] W. J. Savitch. Relationship between Nondeterministic and Deterministic Tape Complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[SAW+07] R. Stevens, M. E. Aranguren, K. Wolstencroft, U. Sattler, N. Drummond, M. Horridge, and A. Rector. Using OWL to model biological knowledge. *International Journal of Human-Computer Studies*, 65(7):583–594, 2007.

[Sch90] L. Schubert. Monotonic Solution of the Frame Problem in the Situation Calculus: An Efficient Method for Worlds with Fully Specified Actions. In H. E. Kyburg, R. P. Loui, and G. N. Carlson, eds., *Knowledge Representation and Defeasible Reasoning*, volume Volume 5, pp. 23–67. Kluwer Academic Publishers, 1990.

[Sch91] K. Schild. A correspondence theory for terminological logics: preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pp. 466–471. Sidney, Australia, 1991.

[Sch93] A. Schaerf. On the Complexity of the Instance Checking Problem in Concept Languages with Existential Quantification. *Journal of Intelligent Information Systems*, 2:265–278, 1993.

[SdCH+06] N. Sioutos, S. de Coronado, M. Haber, F. Hartel, W. Shaiu, and L. Wright. NCI Thesaurus: a semantic model integrating cancer-related clinical and molecular information. *Journal of Biomedical Informatics*, 40(1):30–43, 2006.

[SM73] L. J. Stockmeyer and A. R. Meyer. Word Problems Requiring Exponential Time. In *ACM Symposium on Theory of Computing (STOC '73)*, pp. 1–9. ACM Press, New York, USA, 1973.

[Spa01] K. Spackman. Normal forms for description logic expressions of clinical concepts in SNOMED RT. *Journal of the American Medical Informatics Association*, (Symposium Supplement), 2001.

[SPG+07] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[SS89] M. Schmidt-Schauß. Subsumption in KL-ONE is Undecidable. In *KR'89: Principles of Knowledge Representation and Reasoning*, pp. 421–431. Morgan-Kaufmann Publishers, 1989.

[SSS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[ST06] S. Schiffel and M. Thielscher. Reconciling Situation Calculus and Fluent Calculus. In *Proceedings of The Twenty-First National Conference on Artificial Intelligence (AAAI'06)*. Boston, Massachusetts, USA, 2006.

[SW98] D. E. Smith and D. S. Weld. Conformant Graphplan. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pp. 889–896. American Association for Artificial Intelligence, 1998.

[TH06] D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pp. 292–297. Springer, 2006.

[The00] The Gene Ontology Consortium. Gene Ontology: Tool for the Unification of Biology. *Nature Genetics*, 25:25–29, 2000.

[Thi97] M. Thielscher. Ramification and Causality. *Artificial Intelligence Journal*, 89(1–2):317–364, 1997.

[Thi99] M. Thielscher. From situation calculus to fluent calculus: state update axioms as a solution to the inferential frame problem. *Artificial Intelligence*, 111(1-2):277–299, 1999.

[Thi00] M. Thielscher. Nondeterministic Actions in the Fluent Calculus: Disjunctive State Update Axioms. In S. Hölldobler, ed., *Intellectics and Computational Logic*, pp. 327–345. Kluwer Academic, 2000.

[Thi05a] M. Thielscher. FLUX: A logic programming method for reasoning agents. *TPLP*, 5(4-5):533–565, 2005.

[Thi05b] M. Thielscher. *Reasoning Robots: The Art and Science of Programming Robotic Agents*. Number 33 in Applied Logic Series. Kluwer, 2005.

[Tho75] S. K. Thomason. The logical consequence relation of propositional tense logic. *Z. Math. Logik Grundl. Math.*, 21:29–40, 1975.

[Tob00] S. Tobies. The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000.

[Tob01] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. Ph.D. thesis, RWTH Aachen, 2001.

[Tur02] H. Turner. Polynomial-Length Planning Spans the Polynomial Hierarchy. In *Proceedings of the European Conference on Logics in Artificial Intelligence (JELIA '02)*, pp. 111–124. Springer-Verlag, 2002.

[VK86] M. B. Vilain and H. A. Kautz. Constraint Propagation Algorithms for Temporal Reasoning. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86)*, pp. 377–382. 1986.

[VKvB90] M. Vilain, H. Kautz, and P. van Beek. Constraint propagation algorithms for temporal reasoning: a revised report. In *Readings in qualitative reasoning about physical systems*, pp. 373–381. Morgan Kaufmann, San Francisco, CA, USA, 1990.

[VL07] S. Vassos and H. J. Levesque. Progression of Situation Calculus Action Theories with Incomplete Information. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pp. 2029–2024. 2007.

[Win88] M. Winslett. Reasoning about Action Using a Possible Models Approach. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88)*, pp. 89–93. Saint Paul, MN, 1988.

[Win90] M. Winslett. *Updating Logical Databases*. Cambridge University Press, Cambridge, England, 1990.