

ply deep), terminating in an endgame database position (possibly worth 100s of plies of search).

5. Number of nodes in a back-end search (25 seconds of clock time for each CHINOOK search and 100 seconds for a proof number search): averaging 13,000,000 positions (slow because of extensive I/O).
6. Average number of processors used: seven (with 1.5 to 4 GB of RAM).
7. Total number of nodes searched in the proof: roughly $841,810 \times 13,000,000 > 1 \times 10^{13}$ positions. The number of unique positions is considerably smaller.

So, how does the computer proof stack up against human analysis? We used Fortman's classic *Basic Checkers* as our guide and compared its analysis to that of the proof [Fortman, 1977]. Of the 428 positions given in the book for this opening, there were no disagreements in the game-theoretic result of a position. This is a bit misleading since some of the positions were not part of the proof, and for others the proof only had a bound (less than or equal to a draw) whereas the book had a result (draw). Nevertheless, this result provides further evidence that the computer proof is correct. More impressive, however, is the quality of the human analysis. The analysis of this opening evolved over 75 years, had many human analysts contributing, requires analysis of lines that are over 50 ply long, and requires extensive insight into the game to assess positions and identify lines to explore.

The checker sacrifice in the White Doctor is correct according to our proof. This line is sufficient to prove that the opening is a draw. We have not invested computer cycles to find out if alternate non-sacrifice defences also lead to a draw.

We have made substantial progress on two more checkers openings. Of interest is that one of these openings appears to be roughly 10 times more difficult than the White Doctor, whereas the other one seems to be half as difficult.

6 Solving Checkers

When will checkers be weakly solved? The White Doctor is solved, and two other proofs are well under way (both appear to be draws). There are 174 three-move checkers openings and, ideally, we would like to solve all of them. However, to solve checkers for the initial starting position, with no moves made, roughly 50 openings need to be computed ($\alpha\beta$ cutoffs eliminate most of the openings). Solving subsequent openings will not take as long as the first one did. The computations will speed up for several reasons:

1. Algorithm efficiency. We will switch from experimental mode (trying different performance parameters) to production mode (fixed parameters).
2. More endgame databases. A new algorithm will allow us to compute parts of the 11-piece database, getting 50% of the benefits for 1% of the computing effort.
3. Data reuse. Openings transpose into each other. Each additional opening increases the chances for reusing the results of one opening for another.

The only obstacle remaining is access to computing resources. Computer cycles are easy to obtain; the bottleneck

is large local data storage and I/O speeds. With enough resources, the game could be weakly solved within a year.

References

- [Allis *et al.*, 1991] V. Allis, J. van den Herik, and I. Herschberg. Which games will survive. In *Heuristic Programming in Artificial Intelligence 2*, pages 232–243. Ellis Horwood Limited, 1991.
- [Allis, 1988] V. Allis. A knowledge-based approach to Connect-Four. The game is solved: White wins. Master's thesis, Vrije Universiteit, Amsterdam, 1988.
- [Allis, 1994] V. Allis. *Searching for Solutions in Games and Artificial Intelligence*. PhD thesis, Department of Computer Science, University of Limburg, 1994.
- [Chinook, 1995] Chinook, 1995. <http://www.cs.ualberta.ca/~chinook/databases/total.php>.
- [Fortman, 1977] R. Fortman. Basic checkers, 1977. Privately published, but online at <http://home.clara.net/davey/basicche.html>.
- [Gasser, 1996] R. Gasser. Solving Nine Men's Morris. *Computational Intelligence*, 12:24–41, 1996.
- [Gilbert, 2005] E. Gilbert, 2005. <http://pages.prodigy.net/eyg/Checkers>.
- [Kishimoto and Müller, 2003] A. Kishimoto and M. Müller. Df-pn in Go: Application to the one-eye problem. In van den Herik *et al.* [2003], pages 125–141.
- [Kishimoto and Müller, 2004] A. Kishimoto and M. Müller. A general solution to the graph history interaction problem. In *AAAI*, pages 644–649, 2004.
- [McAllester, 1988] D. McAllester. Conspiracy numbers for min-max search. *Artificial Intelligence*, 35:287–310, 1988.
- [Nagai, 2002] A. Nagai. *Df-pn Algorithm for Searching AND/OR Trees and Its Applications*. PhD thesis, Department of Information Science, University of Tokyo, 2002.
- [Romein and Bal, 2003] J. Romein and H. Bal. Solving the game of awari using parallel retrograde analysis. *IEEE Computer*, 36(10):26–33, 2003.
- [Schaeffer *et al.*, 2003] J. Schaeffer, Y. Björnsson, N. Burch, R. Lake, P. Lu, and S. Sutphen. Building the checkers 10-piece endgame databases. In van den Herik *et al.* [2003], pages 193–210.
- [Schaeffer, 1997] J. Schaeffer. *One Jump Ahead*. Springer-Verlag, 1997.
- [Thompson, 1986] K. Thompson. Retrograde analysis of certain endgames. *Journal of the International Computer Chess Association*, 9(3):131–139, 1986.
- [Trice and Dodgen, 2003] E. Trice and G. Dodgen. The 7-piece perfect play lookup database for the game of checkers. In van den Herik *et al.* [2003], pages 211–230.
- [van den Herik *et al.*, 2003] J. van den Herik, H. Iida, and E. Heinz, editors. *Advances in Computer Games 10*. Kluwer Academic Publishers, 2003.

Structural Symmetry Breaking

Meinolf Sellmann and Pascal Van Hentenryck

Brown University, Department of Computer Science

P.O. Box 1910, Providence, RI 02912, U.S.A.

sello,pvh@cs.brown.edu

Abstract

Symmetry breaking has been shown to be an important method to speed up the search in constraint satisfaction problems that contain symmetry. When breaking symmetry by dominance detection, a computationally efficient symmetry breaking scheme can be achieved if we can solve the dominance detection problem in polynomial time. We study the complexity of dominance detection when value and variable symmetry appear simultaneously in constraint satisfaction problems (CSPs) with single-valued variables and set-CSPs. We devise an efficient dominance detection algorithm for CSPs with single-valued variables that yields symmetry-free search trees and that is based on the abstraction to the actual, intuitive structure of a symmetric CSP.

1 Introduction

During the past five years, symmetry breaking has been a topic of increasing interest in the constraint programming community. It was shown that symmetry breaking can play a decisive role in the solution of numerous constraint problems, and sophisticated symmetry breaking methods have been developed, such as the addition of symmetry breaking constraints (see for example [Crawford *et al.*, 1996]), symmetry breaking by adapting the search strategy [Brown *et al.*, 1988], symmetry breaking during search (SBDS) [Gent and Smith, 2000], or symmetry breaking by dominance detection (SBDD) [Fahle *et al.*, 2001; Focacci and Milano, 2001]. Especially the latter has attracted a lot of interest: improvements were suggested in [Barnier and Brisset, 2002; Puget, 2002] for example. SBDD works by checking whether the current choice point under investigation represents a symmetric variant of a part of the search space that has been investigated completely before.

The core of an SBDD symmetry breaking code is dominance detection which was automated in [Gent *et al.*, 2003] by using the generic computational group theory tool, yielding a method named GAP-SBDD. While using a generic tool is appealing from the perspective of the ease-of-use for the constraint programmer, the generality comes with no guarantee of efficiency. Therefore, in [Van Hentenryck *et al.*, 2003], the special case of dominance detection in the presence of

some classes of pure value symmetry was investigated and proven to be computationally tractable. The approach taken was generalized in [Roney-Dougal *et al.*, 2004], where the notion of symmetry-free search trees, so-called GE-trees, was introduced. In combination with GAP, GE-trees were shown to break arbitrary value symmetries in polynomial time.

In this paper, we continue this successful line of research by studying combinations of value and variable symmetry in constraint satisfaction problems (CSPs) and CSPs with set variables (set-CSPs). We offer three main contributions: The first regards a combination of variable and value symmetry for CSPs with single-valued variables. For the first time, in Section 3, we show that the corresponding dominance detection problem is computationally tractable. In Section 4, we build up on this result and show an efficient way of using the dominance detection algorithm for filtering rather than just pruning. The third main contribution regards set variables: in Section 5 we show that the general dominance detection problem becomes NP-hard for set-CSPs that contain symmetric set variables and value symmetry. Finally, as a minor side-note, in Section 6 we show that higher forms of symmetry where sets of set variables are symmetric to other sets of set variables in combination with value symmetry can be reduced to the partial set variable symmetry case.

2 Preliminaries

Let us start out by introducing some notation that we will use throughout the paper. In the reminder of this paper, let us assume that $n, m, p \in \mathbb{N}$.

Definition 1

- A Constraint Satisfaction Problem (CSP) is a tuple (Z, V, D, C) where $Z = \{X_1, \dots, X_n\}$ is a finite set of variables, $V = \{v_1, \dots, v_m\}$ is a set of values, $D = \{D_1, \dots, D_n\}$ is a set of finite domains where each $D_i \in D$ is the set of possible instantiations to variable X_i , and $C = \{c_1, \dots, c_p\}$ is a finite set of constraints where each $c_i \in C$ is defined on a subset of the variables in Z and specifying their valid combinations. We say that the CSP has single-valued variables iff for all $D_i \in D$ it holds that $D_i \subseteq V$. If for all $D_i \in D$ it holds that $D_i \subseteq 2^V$ (where 2^V denotes the set of all subsets of V), we say that the CSP is a set-CSP.

- Given a CSP with single-valued variables, an assignment A is a set of pairs $(X, v) \in Z \times V$ such that $(X, v), (X, w) \in A$ implies $v = w$. Given a set-CSP, a set-assignment A is a set of pairs $(X, S) \in Z \times 2^V$ such that $(X, S_1), (X, S_2) \in A$ implies $S_1 = S_2$.

Definition 2

- Given a set S and a set of sets $P = \{P_1, \dots, P_r\}$ such that $\bigcup_i P_i = S$ and the P_i are pairwise non-overlapping, we say that P is a partition of S , and we write $S = \sum_i P_i$.
- Given a set S and a partition $S = \sum_i P_i$, a bijection $\pi : S \mapsto S$ such that $\pi(P_i) = P_i$ (where $\pi(P_i) = \{\pi(s) \mid s \in P_i\}$) is called a partial permutation over $S = \sum_i P_i$.

Definition 3

- Given a CSP (Z, V, D, C) , and partitions $Z = \sum_{k \leq r} P_k$, $V = \sum_{l \leq s} Q_l$, we say that the CSP has partial variable and value symmetry iff all variables within each P_k and all values within each Q_l are considered as symmetric.
- Given two assignments A and B on a partially symmetric CSP with single-valued variables, we say that A dominates B iff there exist partial permutations π over $Z = \sum_{k \leq r} P_k$ and α over $V = \sum_{l \leq s} Q_l$ such that for all $(X, v) \in A$ it holds that $(\pi(X), \alpha(v)) \in B$.
- Given a partially symmetric set-CSP and set-assignments A and B , we say that A dominates B iff there exist partial permutations π over $Z = \sum_{k \leq r} P_k$ and α over $V = \sum_{l \leq s} Q_l$ such that $A(\pi, \alpha) \subseteq B$, where $A(\pi, \alpha) := \{(\pi(X), \alpha(S)) \mid (X, S) \in A\}$.
- Given two arbitrary (set-)assignments A and B for a partially symmetric (set-)CSP, we call the problem of determining if A dominates B the Dominance Detection Problem.

3 Symmetric Single-Valued Variables over Symmetric Values

The first general symmetry model that we study is powerful enough to manage symmetric single-valued variables over symmetric values. Throughout this section, we consider the partially symmetric CSP (Z, V, D, C) with single-valued variables. We assume also that we are given a partition of the variables $\sum_{k \leq r} P_k = Z$ and a partition of the values $\sum_{l \leq s} Q_l = V$ such that all variables within each P_k and all values within each Q_l are considered symmetric. Note that such partitions could be derived by a static analysis of a constraint program. The main objective in this section will be to show that there exists an effective symmetry breaking algorithm that runs in polynomial time for this scenario.

The key idea consists in the introduction of *structural abstractions*: to model a CSP, we need to uniquely label each value and each variable with a name — which is, of course, not natural when certain variables and certain values are actually indistinguishable. We can rectify this by viewing each

variable and each value as a member of a symmetry class. In the beginning, these classes correspond directly to the sets P_k and Q_l . When assignments are committed, though, some of those initial symmetries are broken. Then, in order to check which CSP objects are still indistinguishable, we need to introduce subclasses of the original symmetry classes. We will see that we can detect the remaining symmetries by labeling each of those subclasses with a certain *signature* that is defined by the set of initial symmetries and the given assignments. We will see also that it is really these signatures that capture our intuitive wish to abstract from the CSP model at hand to the actual structure of the problem.

3.1 Signatures

Consider the following example: We have variables $X_1, \dots, X_8$ over domains $D(X_1) = \dots = D(X_8) = \{v_1, \dots, v_6\}$. Now assume that the first four and the last four variables are indistinguishable, i.e. $P_1 = \{X_1, \dots, X_4\}$ and $P_2 = \{X_5, \dots, X_8\}$. Furthermore, assume that $Q_1 = \{v_1, \dots, v_3\}$, $Q_2 = \{v_4, \dots, v_6\}$, and that we are given the following two assignments: $A_1 = \{(X_1, v_1), (X_2, v_1), (X_3, v_2), (X_6, v_5), (X_7, v_1), (X_8, v_2)\}$ and $A_2 = \{(X_1, v_6), (X_2, v_1), (X_3, v_2), (X_4, v_2), (X_5, v_1), (X_6, v_6), (X_7, v_2), (X_8, v_2)\}$. See Figure 1(a) for an illustration. When looking at the first assignment, we see that: 1. There is one value (v_1) in Q_1 that is taken by two variables in P_1 and one variable in P_2 . 2. There is one value (v_2) in Q_1 that is taken by one variable in P_1 and one variable in P_2 . 3. There is one value (v_5) in Q_2 that is taken by one variable in P_2 . On the other hand, in the second assignment: I. There is one value (v_2) in Q_1 that is taken by two variables in P_1 and two variables in P_2 . II. There is one value (v_1) in Q_1 that is taken by one variable in P_1 and one variable in P_2 . III. There is one value (v_6) in Q_2 that is taken by one variable in P_1 and one variable in P_2 . Lining up 1-I ($v_1 \mapsto v_2, \{X_1, X_2\} \mapsto \{X_3, X_4\}, \{X_7\} \mapsto \{X_7, X_8\}$), 2-II ($v_2 \mapsto v_1, \{X_3\} \mapsto \{X_2\}, \{X_8\} \mapsto \{X_5\}$), and 3-III ($v_5 \mapsto v_6, \{X_6\} \mapsto \{X_6\}$), we see that A_2 is structurally an assignment extended from A_1 , or, in other words, that A_1 dominates A_2 (see also Figure 1(b)).

What we have done in this small example is to abstract from the given model and the (arbitrary) names of variables and values to the actual *structure* of the problem. That is, instead of talking about specific variables and values, we considered members of classes. Specifically, for each assignment we implicitly assigned each value a *signature* that captures by how many members of each variable-symmetry class it was taken. For instance, in A_1 v_1 has the signature $(2 \times P_1, 1 \times P_2)$, or, in shorter writing, the signature of v_1 is $sig_{A_1}(v_1) = (2, 1)$. In A_2 , on the other hand, the signature of v_2 is $sig_{A_2}(v_2) = (2, 2)$. Consequently, v_2 in A_2 can be viewed as more specialized than v_1 in A_1 , or one may also say that v_1 in A_1 dominates v_2 in A_2 . In this terminology, v_1 in A_2 has signature $sig_{A_2}(v_1) = (1, 1)$ and therefore dominates v_1 in A_1 . Note that $sig_{A_2}(v_6)$ is also $(1, 1)$, but that v_6 in A_2 does not dominate v_1 in A_1 since $v_6 \in Q_2$ whereas $v_1 \in Q_1$. In general:

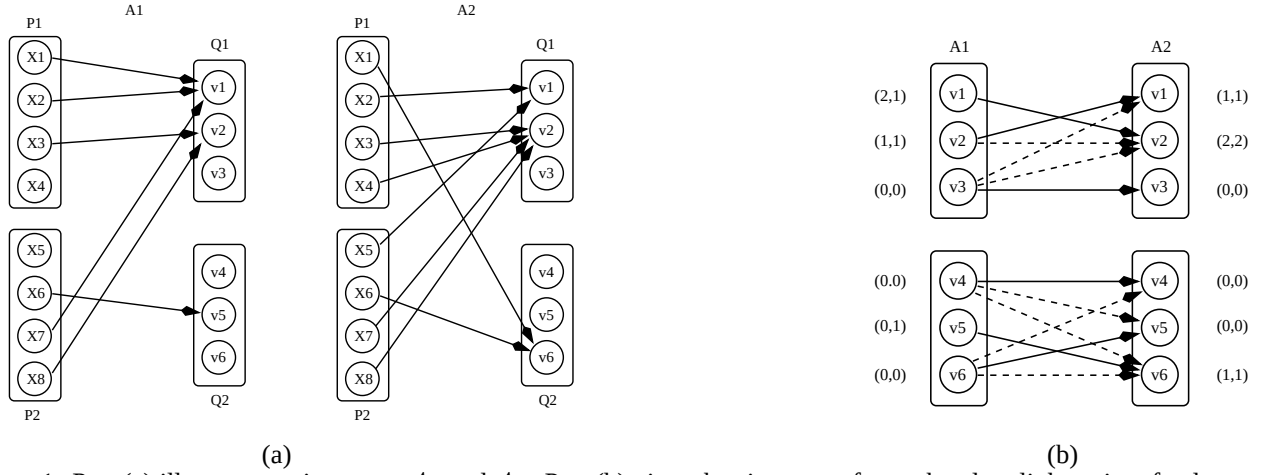


Figure 1: Part (a) illustrates assignments A_1 and A_2 . Part (b) gives the signatures for each value, links pairs of values where the one in assignment A_1 dominates to the one in A_2 , and a perfect matching that proves that A_1 dominates A_2 is designated by solid lines.

Definition 4

- We say that a value v in an assignment A dominates a value w in assignment B iff v and w belong to the same value-symmetry class and $\text{sig}_A(v) \leq \text{sig}_B(w)$.¹
- We say that a value v in an assignment A is structurally equivalent to a value w in assignment B iff v and w belong to the same value-symmetry class and $\text{sig}_A(v) = \text{sig}_B(w)$.

3.2 Dominance Detection Using Signatures

The following lemma shows how signature-abstractions can help to detect dominance relations among assignments:

Lemma 1

An assignment A dominates another assignment B in a partially symmetric CSP iff there exists a partial permutation α over $\sum_{l \leq s} Q_l$ such that v in A dominates $\alpha(v)$ in B for all $v \in V$.

Proof: First, let us assume that A dominates B . Then, there exist partial permutations π over $\sum_{k \leq r} P_k$ and α over $\sum_{l \leq s} Q_l$ such that for all $(X, v) \in A$ it holds that $(\pi(X), \alpha(v)) \in B$. Since both X and $\pi(X)$ belong to the same symmetry class, we have that for all values $v \in V$ it is $\text{sig}_A(v) \leq \text{sig}_B(\alpha(v))$, which is the same as to say that v in A dominates $\alpha(v)$ in B . Now assume there exists a partial permutation α over $\sum_{l \leq s} Q_l$ such that $\text{sig}_A(v) \leq \text{sig}_B(\alpha(v))$ for all $v \in V$. Then, since each variable is assigned to at most one value, there exists a partial permutation π over $\sum_{k \leq r} P_k$ such that for all $(X, v) \in A$ it holds that $(\pi(X), \alpha(v)) \in B$. Thus, we have that A dominates B . ■

Consequently, we have that A dominates B iff there exists a perfect matching in a bipartite graph where the edges are defined by the signature-relation of values (see Figure 1(b)):

¹Where the $\leq$ -relation on vectors is defined as the usual component-wise comparison that yields to the so-called dominance ordering - which is different from a lexicographic ordering.

Definition 5

Given two assignments A and B , let us denote with V' a set of duplicates of the values in V by attaching a prime sign after the values' names (i.e. $V' := \{v' \mid v \in V\}$). Then, we define the dominance detection graph $DDG(A, B) := (V \cup V', E)$ where $E := \{(v, w') \mid v \text{ in } A \text{ dominates } w \text{ in } B\}$ denotes the set of arcs.

Theorem 1

Given two assignments A and B over a CSP with partially symmetric single-valued variables and partially symmetric values, the dominance detection problem between A and B has complexity $O(M + m^2 + mn)$, where $M = O(m^{2.5})$ is the time needed to determine whether there exists a perfect matching in $DDG(A, B)$, m is the number of values, and n is the number of variables.

In the interest of space here and at many other occasions we must omit a formal proof. However, we would like to note that, with Lemma 1, it is clear that the dominance detection problem can be solved basically by determining whether there exists a perfect bipartite matching in $DDG(A, B)$. The additional complexity denoted in the theorem is due to the necessity to construct $DDG(A, B)$ first. It is obvious that this can be achieved in time $O(nm^2)$, which already proves that symmetry breaking in this scenario is tractable. However, the runtime can be improved to the complexity that is claimed here by using sparse representations of signatures. Then, in the algorithm analysis we can exploit that the total number of non-zero signature components is bounded by $O(|A| + |B|) = O(n)$.

Interestingly, it can also be shown that every bipartite graph can also be viewed as a dominance detection graph of a CSP and assignments A and B that can be determined in time linear in the size of the given graph. And therefore, a perfect bipartite matching exists iff A dominates B , which makes the dominance detection problem at least as hard as bipartite matching. In other words, we can show that dominance detection takes time T where $T \in \Omega(M) \cap O(M + m^2 + mn)$.

4 Symmetry-Based Filtering

With Theorem 1, we can break all symmetries when given a CSP with partially symmetric variables and values in polynomial time when using a symmetry breaking by dominance detection (SBDD) approach [Fahle *et al.*, 2001; Focacci and Milano, 2001]. What is annoying in this setting is that we still have to check every choice point to see if it is not dominated by one that was previously expanded, i.e. we still have to touch the garbage in order to see that it is garbage. We will now develop an algorithm that does not suffer from this disadvantage.

We achieve this goal by using dominance detection also for *filtering* rather than just pruning². Obviously, a brute-force approach could simply try assignments out and use the dominance detection algorithm above to perform filtering as well. This procedure would lead to a very poor runtime, though. In the following, we will show that filtering based on symmetry can be performed much more efficiently.

Within SBDD, there exists a natural distinction between two types of filtering that apply: The first consists in making sure that none of the newly created children are symmetric to a node that was fully expanded before the node that is currently branching off. When applying unary branching constraints (which we assume are used here), this can be achieved by shrinking domain variables accordingly. The other, fundamentally different type of “filtering” consists in the creation of children that are also not symmetrical to each other. Both types need to be addressed to achieve a symmetry-free search tree (which corresponds to the GE-trees in [Roney-Dougal *et al.*, 2004]). We distinguish the two types of filtering by naming them differently: *symmetric-ancestor based filtering* and *symmetric-sibling based filtering*.

Symmetric-Ancestor based Filtering

The goal of symmetric-ancestor based filtering is to shrink the domains such that instantiating a variable with one of its domain values will not result in the creation of a search node that is symmetric to one that was previously expanded.

Definition 6

Given a depth-first-search tree T ,³ we say that a choice point B (associated with its homonymous assignment B that captures previously committed unary branching decisions) is ancestor-symmetry resistant iff for all previously fully expanded nodes $A \in T$ (A is called an ancestor of B) and for all variables X and values $v \in D(X)$ it holds that A does not dominate $B \cup \{(X, v)\}$.

Assume that we are currently investigating choice point B and that A is some ancestor node that does not dominate B . Observe that instantiating one more variable $X \in P_k$ for some k by setting $X \mapsto w \in Q_l$ for some l will change only the signature of w from $\text{sig}_B(w)$ to $\text{sig}_B(w) + e_k$, where e_k denotes the unit vector with a 1 in the k th component. We set $B' := B \cup \{(X, w)\}$. Then, $D_1 := \text{DDG}(A, B)$

²With ‘filtering’, we refer to the idea of domain reduction in CP; ‘pruning’ refers to the detection of a sufficient reason to backtrack.

³In the interest of space, we must omit a formal definition of a search tree here. The meaning should become clear from the context.

and $D_2 := \text{DDG}(A, B')$ only differ in that the latter bipartite graph may contain some additional edges that must all be incident to w' in the right partition. Obviously, if D_2 contains an m -matching, this matching must contain exactly one of those additional edges. Consequently, if A dominates B' , then D_1 must contain an $m - 1$ -matching. Only if this is the case, work needs to be done to make B ancestor-symmetry resistant with respect to A .

So let us assume that D_1 contains an $m - 1$ -matching. Provided with that matching, using some straight-forward matching theory we can identify efficiently those and only those additional edges that would allow us to transform the existing matching into a perfect one. Furthermore, it can also be shown that those critical edges are independent of the particular $m - 1$ -matching that we computed (For an introduction to matching theory we refer to [Ahuja *et al.*, 1993]). Among those critical edges that, if added, would allow us to construct an m -matching, the only ones that we need to consider are those which run between nodes v and w' with $v, w \in Q_l$ for some $1 \leq l \leq s$ and for which there exists $1 \leq k \leq r$ such that $\text{sig}_A(v) \leq \text{sig}_B(w) + e_k$. If and only if we find such a pair of nodes, a single extra assignment added to B will result in a successful dominance detection. Precisely, every assignment of w to a previously unassigned variable $X \in P_k$ will result in a dominated choice point. Thus, if we remove w from the domain of X for all unassigned $X \in P_k$, we keep the unique parts of the search space and we never produce choice points that are symmetric to one that was expanded previously to B .

With Theorem 1, the runtime needed for the initial value-matching algorithm is bounded by $O(m^{2.5} + mn)$. We can also prove that a filtering algorithm can be formulated that, once the matching is computed, runs in time $O(m^2 + mn)$.⁴ Therefore, since within SBDD at most $n(m - 1)$ ancestor nodes need to be considered, we can prove the following:

Theorem 2

Given a CSP with partially symmetric single-valued variables and partially symmetric values, we can achieve ancestor-symmetry resistance for a given search node in time $O(nm^{3.5} + n^2m^2)$.

Symmetric-Sibling based Filtering

To achieve full symmetry prevention, we also need to guarantee that newly created siblings are not symmetric to each other. Therefore, after we choose the next variable to be assigned, but before we branch on it, we need to perform one more “filtering” step (it is actually more of an implicit pruning step) where we choose a single representative value out of each equivalence class of values which, when assigned to the chosen variable, would result in the creation of symmetric choice points. Due to the fact that, whenever a sibling dominates another, they both must already be structurally equivalent (see Definition 4), we can avoid producing symmetric siblings simply by choosing exactly one representative value

⁴Again, the main technical idea consists in the use of sparse signatures. The proof of the complexity is not particularly insightful but lengthy, which is why we leave it out here.

among those that are structurally equivalent. The complexity of this filtering step is dominated by that of symmetric-ancestor based filtering.

Putting ancestor and sibling-based filtering together, we have completed our development of an effective symmetry breaking algorithm for CSPs with partial single-valued variable and value symmetry that runs in polynomial time. Note that the practical performance of the algorithms sketched can be enhanced in practice: for example, it is fully sufficient to check against previously expanded nodes for which an $m - 1 - h$ -maximum matching was found only after variable instantiations to h different values have been committed. And as usual, by considering incremental updates of matchings, memory can be traded for cpu-time.

5 Limits of Efficient Dominance Detection

After having developed a polynomial symmetry breaking algorithm for CSPs with partially symmetric single-valued variables in the presence of partial value symmetry, we now show that dominance detection for partially symmetric set variables in the presence of partial value symmetry is NP-hard. More precisely, we reduce the corresponding dominance detection problem to subgraph-isomorphism.

In order to achieve the desired reduction we construct a set-assignment from a graph in the following way:

Definition 7

Given an undirected graph $G = (V, E)$ with $c := |V|$, we define a set of symmetric values $N := \{n_1, \dots, n_c\}$, and a set of symmetric variables $P := \{p_{ij} \mid \{i, j\} \in E\}$. Then, the set-assignment $A(G)$ is defined as $A(G) := \{(p_{ij}, \{n_i, n_j\}) \mid \{i, j\} \in E\}$.

Theorem 3

Given two undirected graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, G_1 is sub-isomorphic to G_2 iff $A(G_1)$ dominates $A(G_2)$ when all variables and values are considered to be symmetric.

Proof: We start by showing that $A(G_1)$ dominates $A(G_2)$ if G_1 is sub-isomorphic to G_2 . Let $\sigma : V \mapsto V$ bijective such that $\{i, j\} \in E_1$ implies $\{\sigma(i), \sigma(j)\} \in E_2$. Then, for all $(p_{ij}, \{n_i, n_j\}) \in A(G_1)$ it holds that $(p_{\sigma(i), \sigma(j)}, \{n_{\sigma(i)}, n_{\sigma(j)}\}) \in A(G_2)$. Therefore, $A(G_1)$ dominates $A(G_2)$.

Now let us assume that $A(G_1)$ dominates $A(G_2)$. Then, there exist functions $\pi : E_1 \mapsto E_2$ and $\alpha : V \mapsto V$ such that $(p_{ij}, \{n_i, n_j\}) \in A(G_1)$ implies $(p_{\pi(\{i, j\})}, \{n_{\alpha(i)}, n_{\alpha(j)}\}) \in A(G_2)$. By construction of $A(G_2)$, this is equivalent to $\{n_{\alpha(i)}, n_{\alpha(j)}\} \in E$ for all $\{i, j\} \in E$. Thus, α is a sub-isomorphism between G_1 and G_2 . ■

With Theorem 3, it is easy to prove the following

Corollary 1

The dominance detection problem over partially symmetric set variables and partially symmetric values is NP-hard.

Proof: We reduce the problem to subgraph-isomorphism. In order to apply Theorem 3, we need to ensure that both graphs operate over the same set of nodes. In case that the set of nodes of the given graphs differ, it is easy to see that G_1 cannot be sub-isomorphic to G_2 if G_1 contains more nodes than G_2 . In case that G_1 actually contains fewer nodes than G_2 , it is easy to see that we can add isolated nodes to G_1 without affecting subgraph-isomorphism. Then, we have that both graphs contain the same number of nodes, and by relabelling the nodes in both graphs, we may assume that both graphs operate on the same set of nodes. ■

Note that, despite this negative result, in some important special cases the symmetry detection problem for CSPs with partially symmetric set variables and values is still tractable. For instance, when the set variables cannot overlap, the algorithm developed in Section 3 can be adapted easily (by simply exchanging the roles of values and variables) to break all symmetries efficiently.

Corollary 2

The dominance detection problem over partially symmetric non-overlapping set variables and partially symmetric values is tractable.

Note that the dominance detection problem as we consider it here regards arbitrary assignments. This implies that, when the detection problem is tractable, we can break symmetries efficiently. However, the situation changes when we achieve an intractability result like the previous one: Within methods like SBDD the assignments that need to be compared can only differ in a rather specific fashion. We can also show that these more specific dominance detection problems are NP-hard as well, therefore proving that SBDD in its general form is incapable of breaking symmetries in partially symmetric set-CSPs efficiently. However, we would like to stress that this result does not imply that symmetry breaking is NP-hard in general since we do not consider other methods here like remodeling or the adaption of the branching scheme.

6 Higher Forms of Symmetry

Note that our intractability proof above shows that dominance detection over symmetric set variables and symmetric values is already NP-hard when there exists only one set of symmetric set variables and one set of symmetric values. Clearly, the problem stays NP-hard when we allow partial symmetry. Partial symmetry is very helpful when even more complicated forms of symmetry need to be handled. Consider for example the Social Golfer Problem (SGP):

32 golfers want to play in 8 groups of 4 each week, such that any two golfers play in the same group at most once. How many weeks can they do this for?⁵

The problem can be generalized by parameterizing it to g groups of s players each, playing for w weeks (instances are usually written in $g-s-w$ -format). A common model for the problem introduces a set variable for each group, so that intra-group symmetries are broken automatically. Groups within

⁵Problem 10 in CSPLib - <http://www.csplib.org/>.

	Group 1		Group 2		Group 3	
Week 1	1	2	3	4	5	6
Week 2	1	3	2	5	*	*

Table 1: A partial instantiation of SGP 3-2-2.

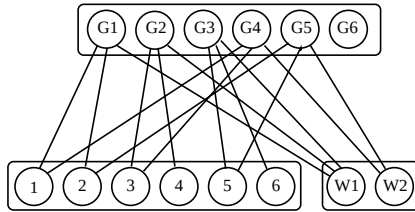


Figure 2: Symmetry-breaking model for SGP 3-2-2 from Table 1. We view the problem as containing six groups $G_1, \dots, G_6$ (3 per week). Only upon instantiation of the associated set variable, a group is assigned to a week.

the same week are symmetric, and furthermore, whole weeks are also symmetric, i.e. whole sets of set variables are symmetric to other sets of set variables. We can model this by introducing new week-values, one of which each group must be assigned to. Consider the assignment to an SGP instance 3-2-2 given in Table 1: By introducing two week-values W_1 and W_2 to one of which each group-variable is assigned upon instantiation, we can model the SGP as a dominance detection problem over symmetric set variables and partially symmetric values (see Figure 2). By means of this little trick, higher forms of symmetry can be reduced to symmetric set variables over partially symmetric values, which makes the corresponding dominance detection problem a most relevant research subject. We believe that developing fast algorithms that (approximately) solve this NP-hard problem should be a focus of symmetry breaking research at this point.

7 Conclusion

When breaking symmetry by dominance detection, a computationally efficient symmetry breaking scheme can be achieved if we can solve the dominance detection problem in polynomial time. Therefore, we studied the complexity of dominance detection between arbitrary assignments for combined variable and value symmetry. For CSPs with partially symmetric single-valued variables and partially symmetric values we developed an efficient dominance detection algorithm that shaves the search-tree to a point where no two search-nodes can be symmetrical to each other. In contrast to these positive tractability results, for set-CSPs with symmetric variables and symmetric values, we found that the dominance detection problem is NP-hard.

Our algorithms are based on the structural abstraction of a given CSP, a method that we refer to as *structural symmetry breaking (SSB)*. When only the structure of variables and values that partition into whole equivalence classes of pairwise symmetric elements is considered, the work presented here effectively closes the outstanding complexity questions regarding SBDD. Further research will have to regard CSPs

whose symmetry structures are more refined. Consider for example the case where the variables form a ring structure and all rotations on the ring yield to an equivalent CSP. Under what conditions can we break value and variable symmetry when the permutations on the CSP elements form structures that are different from partitions?

In our view, it is perceivable that structural symmetries could be derived automatically by a static analysis of a given constraint program if the modeling language used provided semantics regarding symmetries. For instance, an all-different constraint could provide the information that all variables and values are treated as symmetric by this constraint. Then, built-in efficient dominance detection algorithms like the one that we developed can be used efficiently to break the symmetries that were derived automatically, thus making symmetry breaking an effective and efficient component of constraint programming that is totally seamless for the user.

Acknowledgments

Many thanks to Pierre Flener and Justin Pearson for helping tremendously in polishing up the presentation of this work!

References

- [Ahuja *et al.*, 1993] R. Ahuja, T. Magnati, J. Orlin. *Network Flows*. Prentice Hall, 1993.
- [Barnier and Brisset, 2002] N. Barnier and P. Brisset. Solving the Kirkman’s schoolgirl problem in a few seconds. *Proc. CP’02*, 477–491, 2002.
- [Brown *et al.*, 1988] C. Brown, L. Finkelstein, P. Purdom Jr. Back-track searching in the presence of symmetry. *Proc. AAECC-6*, 99–110, 1988.
- [Crawford *et al.*, 1996] J. Crawford, M. Ginsberg, E. Luks, A. Roy. Symmetry-breaking predicates for search problems. *Proc. KR’96*, 149–159, 1996.
- [Fahle *et al.*, 2001] T. Fahle, S. Schamberger, M. Sellmann. Symmetry Breaking. *Proc. CP’01*, 93–107, 2001.
- [Flener *et al.*, 2002] P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, T. Walsh. Breaking row and column symmetries in matrix models. *Proc. CP’02*, 462–476, 2002.
- [Focacci and Milano, 2001] F. Focacci and M. Milano. Global cut framework for removing symmetries. *Proc. CP’01*, 77–92, 2001.
- [Gent *et al.*, 2003] I. Gent, W. Harvey, T. Kelsey, S. Linton. Generic SBDD using computational group theory. *Proc. CP’03*, 333–347, 2003.
- [Gent and Smith, 2000] I. Gent, B. Smith. Symmetry breaking in constraint programming. *Proc. ECAI’00*, 599–603, 2000.
- [Puget, 2002] J.-F. Puget. Symmetry breaking revisited. *Proc. CP’02*, 446–461, 2002.
- [Roney-Dougal *et al.*, 2004] C. Roney-Dougal, I. Gent, T. Kelsey, S. Linton. Tractable symmetry breaking using restricted search trees. *Proceedings of ECAI’04*, 2004.
- [Van Hentenryck *et al.*, 2003] P. Van Hentenryck, P. Flener, J. Pearson, M. Agren. Tractable symmetry breaking for CSPs with interchangeable values. *Proc. IJCAI’03*, 2003.

Decentralized Search in Networks Using Homophily and Degree Disparity

Özgür Şimşek and David Jensen

Computer Science Department

University of Massachusetts

Amherst, MA 01003-9264

{ozgur, jensen}@cs.umass.edu

Abstract

We propose a new algorithm for finding a target node in a network whose topology is known only locally. We formulate this task as a problem of decision making under uncertainty and use the statistical properties of the graph to guide this decision. This formulation uses the homophily and degree structure of the network simultaneously, differentiating our algorithm from those previously proposed in the literature. Because homophily and degree disparity are characteristics frequently observed in real-world networks, the algorithm we propose is applicable to a wide variety of networks, including two families that have received much recent attention: small-world and scale-free networks.

1 Introduction

In a well known study, Travers & Milgram [1969] asked individuals in Boston, Massachusetts and Omaha, Nebraska to deliver a letter to a target person in Boston, using an unconventional method: the letters were to reach the target person through a chain of acquaintances. The person starting the chain and all subsequent recipients of the letter were given some basic information about the target—including name, address, and occupation—and were asked to forward the letter to someone they knew on a first name basis, in an effort to deliver the letter to the target person with as few intermediaries as possible. Of the 296 letters that were distributed, 22% reached the target, with a median chain length of six.

These findings revealed two surprising properties of the social network—that short paths exist between seemingly unconnected individuals and that people are able to find them—and raised a number of questions: How do people perform this type of search? What properties of the social network make it searchable? And in their presence, how can we search a network efficiently? In this paper we address this last question and present an algorithm for efficient decentralized search in a class of networks that exhibit the properties of the social network of acquaintances.

At the time, the task faced by the participants in the Travers & Milgram study—searching for a target node in a large network whose topology is known only locally—was highly artificial, designed only to explore the structure of the social

network of acquaintances. Today, it appears naturally in various contexts. For example, a similar task is performed when people and focused crawlers [Diligenti *et al.*, 2000; Chakrabarti *et al.*, 1999] search for information in the World Wide Web by following links. The same is true of search protocols that form the backbone of decentralized peer-to-peer file sharing systems such as Gnutella and Freenet [Clarke *et al.*, 2000] that lack a central server to answer queries.

If decentralized search is to succeed, it is essential that the underlying network possess some form of structure that can guide the search. The acquaintance network has at least two characteristics that create such structure. The first is homophily, the *tendency of like to associate with like*, in other words, the tendency of attributes of connected nodes to be correlated—people tend to be acquainted with other people who live in the same geographical area or who have the same occupation. The second characteristic is degree disparity—some people have more acquaintances than others and may act as hubs that connect different social circles. Consideration of homophily gives rise to a message-passing algorithm that favors the neighbor that is the most similar to the target node (e.g., an acquaintance who lives in Boston, if the target person lives in Boston) [Kleinberg, 2000a; 2000b; 2001; Watts *et al.*, 2002], while consideration of degree structure gives rise to an algorithm that favors the neighbor with the highest degree [Adamic *et al.*, 2001].

Building on the insights gained by this recent body of research, we propose a new message-passing algorithm—*expected-value navigation* (EVN)—for decentralized search in networks. Our formulation of the problem is fundamentally different from prior approaches in that it considers the *entirety* of the factors that may influence the effectiveness of the search. We cast the problem faced by each node in the message chain as a decision making task under uncertainty, in which the objective is to minimize the expected length of the search path. This decision is guided by the statistical properties of the graph, in which both homophily and degree play a role. All prior algorithms have essentially used only part of the available information.

Because homophily and degree disparity are characteristics frequently observed in real-world networks, EVN is applicable to a wide variety of networks. We emphasize two families that have received much recent attention: small-world and scale-free networks.

Small-world networks are loosely defined as a family of graphs with a combination of three properties that distinguish them from other graph families (e.g., random graphs, fully connected graphs, and regular graphs): weak connectivity, strong clustering, and small diameter. Many real world networks show the small-world structure, including the World Wide Web, the electrical power grid of the western United States, the collaboration graph of Hollywood actors, and the neural network of the nematode worm *C. elegans* [Watts and Strogatz, 1998]. These networks, by definition, connect most node pairs by short paths, and EVN may be particularly well suited for finding them as the small world structure may arise from homophily [Kleinberg, 2000b].

Scale-free networks are those networks with a power-law degree distribution, which means that the probability of a given degree k is proportional to $k^{-\beta}$, where β is a parameter known as the degree exponent. Most nodes in such networks have only a few edges, but a few nodes have much higher degree. Many small-world networks are scale-free [Barabasi and Albert, 1999].

The remainder of this paper is organized as follows: We first provide a review of the relevant literature. We then present our formulation of the search problem, describe the algorithm we propose, and evaluate it on a collection of synthetic and real-world networks. We conclude with a discussion of our experimental results and directions for future research.

2 Previous Work

A number of message-passing algorithms have been proposed for conducting decentralized search in networks, in which each node receiving the message forwards it to one of its neighbors until the target is found. Based on the decision criteria they use in selecting a forwarding node, these algorithms can be categorized as follows:

- *Degree-Based*—The decision is based on the degree structure of neighboring nodes.
- *Similarity-Based*—The decision is based on how similar the neighboring nodes are to the target node in terms of attribute values.

2.1 Degree-Based Navigation

Adamic *et al.* [2001] proposed an algorithm that forwards the message to the highest-degree neighbor that has not seen the message. On a scale-free network with degree exponent 2.1, in which nodes knew their immediate neighbors and *their* neighbors, this algorithm performed fairly well. Most nodes were easy to find—about 50% of the target nodes were found within 12 hops in a 10,000 node network—but a small proportion of nodes required a much larger hop count. Similar results were obtained on a 700-node subgraph of the Gnutella peer-to-peer file sharing system, which showed a power-law degree distribution with exponent 2.07. The algorithm, however, was not effective on networks with Poisson degree distribution.

2.2 Similarity-Based Navigation

In similarity-based navigation, nodes forward the message to the neighbor that is the most similar to the target node, given a number of attributes on nodes and a similarity metric. This type of search relies on network homophily. Under some conditions, similarity among attribute values of neighboring nodes provides an approximation to a universal gradient that allows short paths to be identified from only local information.

The first algorithmic analysis of similarity-based navigation was performed by Kleinberg [2000b; 2000a] on a simple network motivated by the geographical distribution of acquaintances. This network had nodes on a two-dimensional lattice; each node was connected to all other nodes within a given lattice distance and also to a number of additional nodes across the grid. The probability of a connection of the latter type was proportional to the lattice distance between the nodes raised to the power $-\alpha$, where α is a model parameter called the clustering exponent. In this network, similarity between nodes is defined by their lattice position, and the clustering exponent controls the homophily in the graph: When $\alpha = 0$, long-range contacts are uniformly distributed on the grid; as α increases, long-range contacts become more and more clustered in the node's vicinity.

Kleinberg showed that when $\alpha = 2$, similarity-based navigation achieves an expected path length bounded by a polylogarithmic function (i.e., a polynomial function of the logarithm) of the number of nodes, and that $\alpha = 2$ is the only clustering exponent at which a polylogarithmic bound on path length is possible. These results generalize to d -dimensional lattices for $d \geq 1$, with the critical value of $\alpha = d$.

Kleinberg [2001] later proved similar results for two other network models that defined node similarity differently, but as in his first model, the link probabilities were a function of the similarity between the nodes and a parameter that controls the degree of homophily in the graph. In both of these models, there was a critical value of the homophily parameter that allowed similarity-based navigation to achieve a search time polylogarithmic in the number of nodes, and for all other values of this parameter, a polylogarithmic upper bound was not possible.

Similarity-based navigation was also explored by Watts *et al.*, [2002], who proposed a hierarchical model of society and a homophily structure that measured similarity with distance in this hierarchy. The authors explored the influence of the number of hierarchies and the homophily parameter on the searchability of the network and found that similarity-based navigation was effective for a large region of the parameter space.

3 Proposed Algorithm: Expected-Value Navigation

We propose a message passing algorithm that builds on the strengths of the algorithms discussed above. We derive this algorithm by formulating the problem as a decision making task under uncertainty, in which the goal is to minimize the expected path length to the target.

The expected value of the path length l_{st} from neighbor s to target t is a weighted sum of all possible path lengths:

$$E(l_{st}) = \sum_{\forall i} i \cdot P(l_{st} = i) \quad (1)$$

We assume that in computing this expected value the following information is available: a list of nodes that have already seen the message, the properties (i.e., degree and attribute values) of neighboring nodes and of the target node, and the known (or estimated) homophily structure of the graph—in other words, a statistical relationship between node similarity and probability of a link. This last piece of information allows us to compute the probability that a given neighbor links to the target node.

We approximate the entire series in Equation 1 using only the first two terms, which are easy to compute given the information available. This estimate captures much of the necessary information because there is no need to know the exact value of the expectation, only whether it is lower than the expectation computed for another neighbor.

If one of the neighbors is the target, this neighbor has $E(l_{st}) = 0$, the lowest possible value. Otherwise, the node for which the second term in the series is the highest minimizes our estimate of $E(l_{st})$ —the larger the probability of a path length of one, the smaller the probability of larger path lengths, and in general, the smaller the expected path length. Note here that for a neighbor that has already seen the message, the second term in the series is zero—we know with certainty that it does not link to the target, otherwise it would have forwarded the message to the target and completed the search.

This gives rise to the following algorithm: If one of the neighbors is the target node, forward the message to this node. Otherwise, forward the message to the unvisited neighbor with the highest probability of having a direct link to the target. If all neighbors have been visited, forward the message to a randomly selected neighbor.

We call this algorithm expected-value navigation (EVN), based on its method of node selection. If the network shows no homophily (i.e., if links are formed independently of node similarity) or if attributes are not available, EVN reduces to the degree-based navigation of Adamic *et al.* [2001]. On the other hand, if degree information is unavailable or if all nodes have equal degree, EVN reduces to similarity-based navigation that avoids visited nodes when possible.

In order to apply EVN in a given network, one needs to compute or estimate the probability that a link exists from one node to another, given the attribute value and degree of both nodes. We estimate this probability assuming that each link is placed independently of the others. For a link from node s to node t , the desired probability p_{st} can then be computed by subtracting from 1.0 the probability that none of the links that originate at s ends at t :

$$p_{st} = 1 - (1 - q_{st})^k \quad (2)$$

where q_{st} is the probability that the first link from s ends at t , and k is the out-degree of node s . This is one of the simplest estimators that uses information on both homophily and

degree—the underlying assumption of independent links is violated in the networks we consider—but our results show that it performs remarkably well.

4 Experimental Evaluation

We evaluated EVN on a collection of synthetic and real-world networks, comparing its performance to three other message passing algorithms: similarity-based, degree-based, and random navigation. These algorithms treat visited neighbors similarly—ignoring them in the presence of unvisited neighbors, and selecting randomly among them otherwise—but differ in how they select among unvisited neighbors. Similarity-based navigation selects the one most similar to the target node in attribute value, degree-based navigation selects the one with the highest degree, and random navigation selects randomly. If more than one neighbor satisfies the criteria, all algorithms select randomly among them.

It is possible to construct other variations of similarity-based, degree-based, and random navigation that differ in how they treat visited neighbors. For instance, the algorithm may ignore prior visitations, or only avoid the last visited node. The versions described above consistently outperformed these variations in our simulations; we therefore do not discuss them any further.

In addition to these four algorithms, we also present the performance of an optimal global algorithm, which returns the shortest path length from source to target if it is less than the number of hops allowed. The performance of the optimal algorithm is a ceiling for the other algorithms—if there is no short path, no algorithm can find it.

We present four performance measures: proportion of successful searches (*prop*), mean path length when successful (*path*), median path length when successful (*median-path*), and mean optimal path length when successful (*opt-path*). The last measure indicates the difficulty of the search tasks an algorithm succeeds at, and is useful when comparing mean path lengths of different algorithms—if one algorithm is able to succeed at more difficult search tasks than another one, the mean path lengths are not directly comparable.

4.1 Synthetic Networks

We considered directed networks with two types of out-degree distribution: power-law and Poisson. We defined a single attribute on each node, which was distributed uniformly in the interval $[0, 1]$. Each network had 1000 nodes; nodes with out-degree higher than 100 were not allowed. Search was terminated after 100 hops if the target was not reached.

The number of outgoing links from each node was determined based on the out-degree distribution of the graph. For a link originating at node s , the probability of linking to node t was proportional to f_{st} , the preference between the two nodes, which we defined as follows:

$$f_{st} = (\max\{|a_s - a_t|, 0.01\})^{-r} \quad (3)$$

where a_s, a_t are attribute values on nodes s and t , and r is a homophily parameter. The *max* term puts a bound on the preference values—in its absence, the preference between two

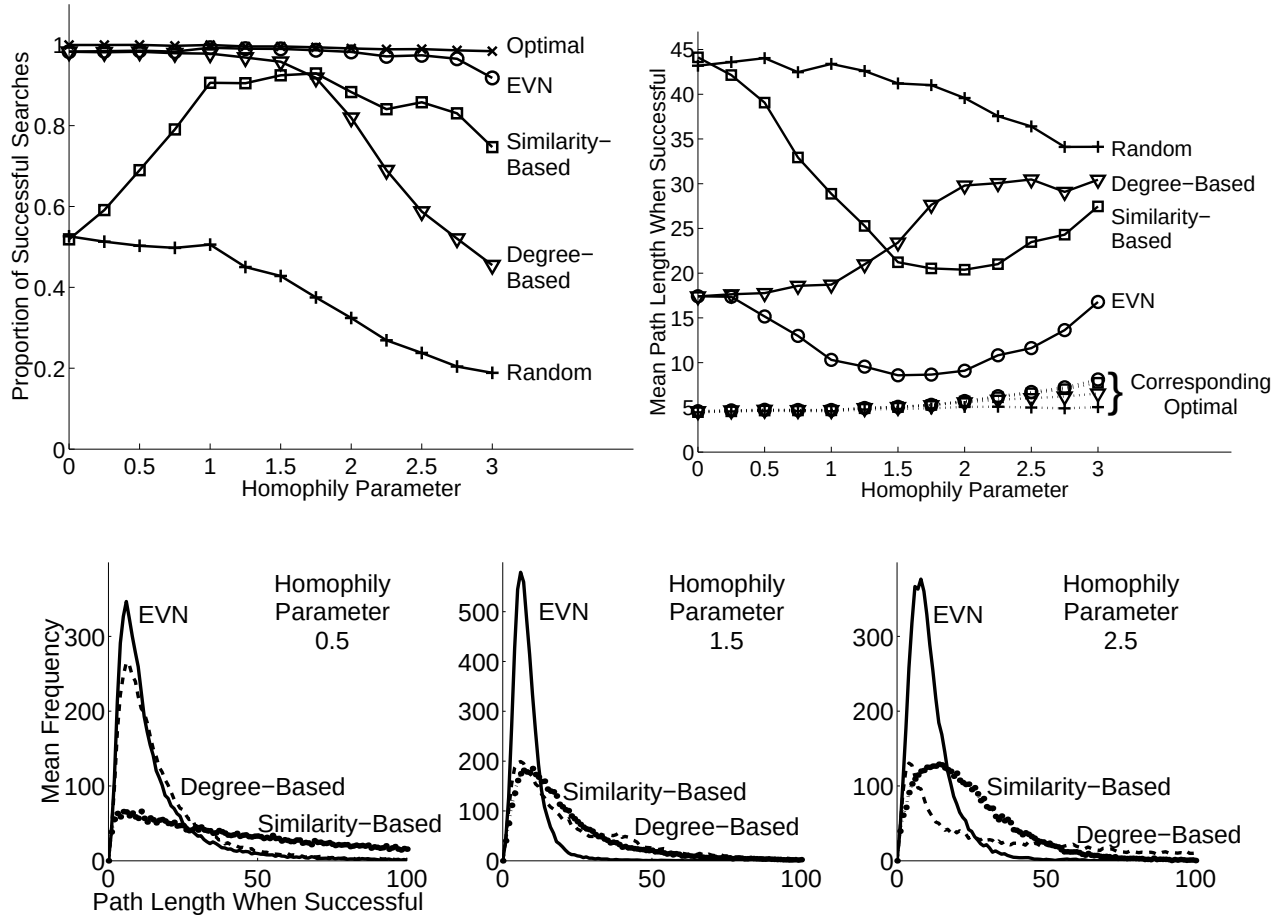


Figure 1: Performance on scale-free networks with degree parameter 1.5.

nodes may be arbitrarily large, as two attribute values may be arbitrarily close.

When r is zero, the graph shows no homophily—a link originating from a given node is equally likely to end at any other node. As r grows, links become more likely to connect nodes with similar attribute values. The findings in Kleinberg [2000b] suggest that small values of r give rise to a homophily structure too weak to guide the search effectively, while large values of r give rise to a graph structure that does not contain short paths.

In applying EVN in these networks, we used $p_{st} = 1 - (1 - f_{st}/(\sum_j f_{sj}))^k$, which was obtained from Equation 2 by substituting q_{st} with $f_{st}/\sum_j f_{sj}$, the ratio of the preference between nodes s and t to the sum of preferences from s to all nodes in the network. In applying similarity-based navigation, we considered all neighbors within 0.01 of the target to be equally close, to account for the presence of the max term in Equation 3.

Networks with Power-Law Degree Distribution

We considered power-law distributions with degree parameters ranging from 1 to 3. This range includes the distributions most frequently observed in real-world networks. The ho-

mophily parameter ranged from 0 to 3. Each possible combination of degree parameter and homophily parameter was evaluated on 10 randomly generated networks (unless noted otherwise), with 5000 randomly selected search tasks in each network.

Figure 1 shows performance on scale-free networks with degree parameter 1.5. In addition to *prop*, *path*, and *opt-path*, this figure also presents the frequency of path lengths when the homophily parameter was 0.5, 1.5, and 2.5. While similarity-based navigation was effective for large values of the homophily parameter and degree-based navigation was effective for lower values, EVN was effective with over 95% success rate for all values of the homophily parameter and returned shorter path lengths than the other algorithms.

Figure 2 shows performance on scale-free networks with varying degree parameters. The data points in this figure show mean values in 30 randomly generated networks. EVN was effective under a wide range of parameter settings and consistently outperformed both degree-based and similarity-based navigation. EVN succeeded at more difficult search tasks than degree-based navigation, as measured by *opt-path*, while returning considerably shorter path lengths. EVN and similarity-based navigation performed similarly in *opt-path*,

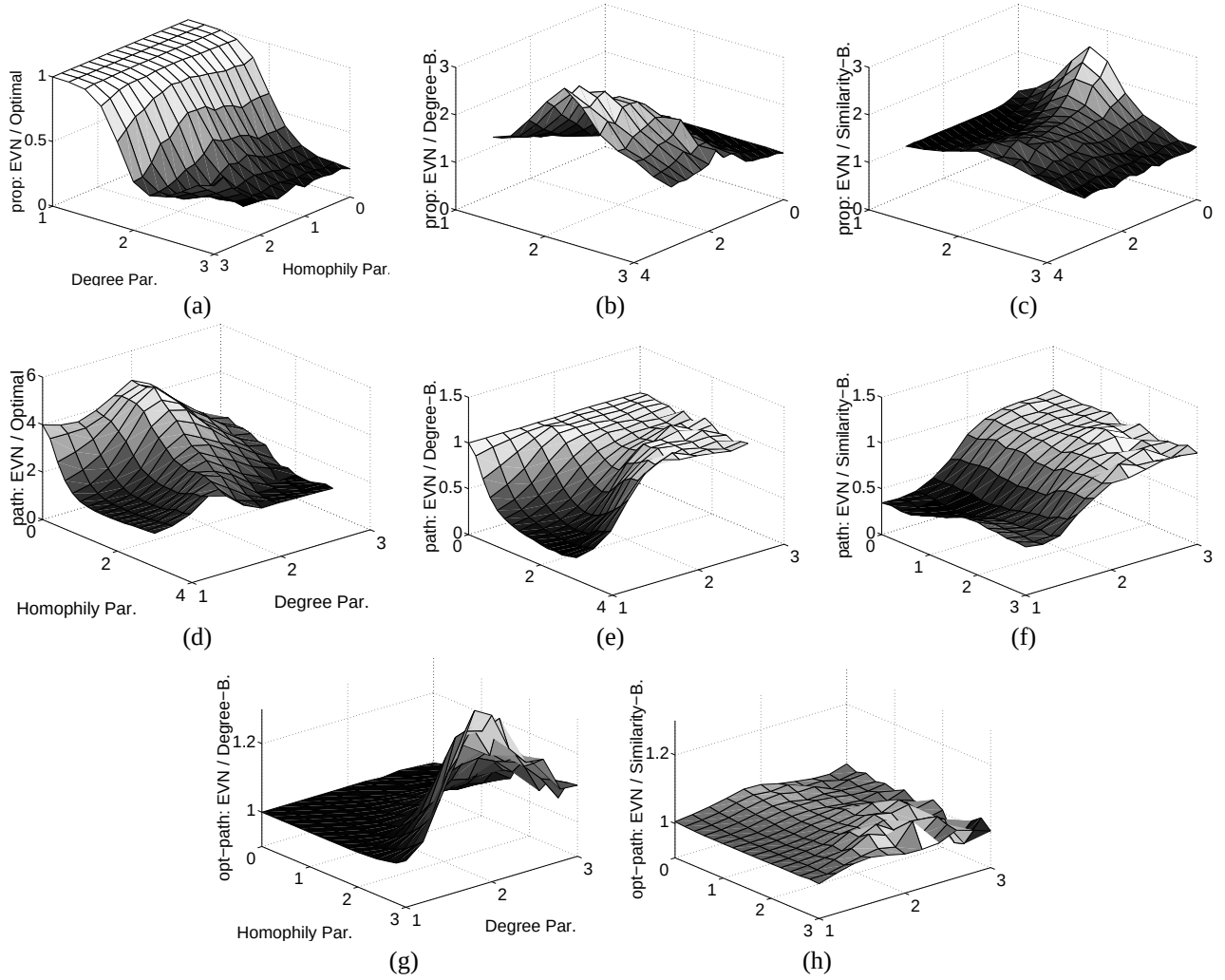


Figure 2: Performance of EVN on scale-free networks. The first row shows the proportion of successful searches by EVN divided by that of (a) optimal, (b) degree-based, and (c) similarity-based navigation; the second row shows mean path length (when successful) for EVN divided by that of (d) optimal, (e) degree-based, and (f) similarity-based navigation; the third row shows mean opt-path of EVN divided by that of (g) degree-based and (h) similarity-based navigation.

but EVN returned shorter path lengths. The proportion of successful searches was higher for EVN than for both degree-based and similarity-based navigation.

Networks with Poisson Degree Distribution

Similar experiments were conducted on networks with a Poisson degree distribution with a mean out-degree of 3.5, which approximately equals the mean degree in the scale-free networks that were tested.

Figure 3 shows performance results. The figure reveals that similarity-based and degree-based navigation were not effective in these networks, succeeding in less than half of the search tasks for all values of the homophily parameter. EVN returned a higher proportion of successful searches than both degree-based and similarity-based navigation; the difference was substantial for a large range of values of the homophily parameter. EVN was most effective with homophily param-

eter values close to 2.

The *path* results shown in Figure 3 may seem counter-intuitive, with random navigation returning the lowest values for most values of the homophily parameter. Recall, however, that *path* refers to the mean path length in *successful* searches, so the mean path lengths are not directly comparable—random navigation returned the lowest *path* values, but it succeeded in only the easiest search tasks as measured by *opt-path*. The *opt-path* results show that EVN succeeded at more difficult search tasks than both degree-based and similarity-based navigation. Furthermore, EVN returned considerably shorter path lengths than degree-based navigation, despite succeeding in more difficult search tasks.

Robustness of EVN

In estimating link probabilities, we used the sum of preferences from a given node to all other nodes in the network.

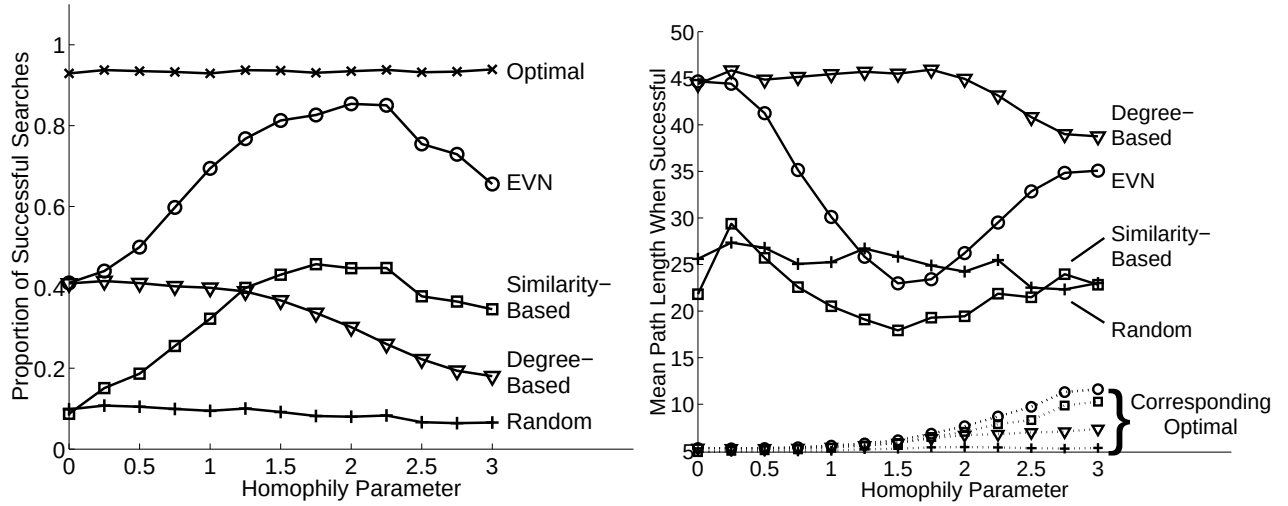


Figure 3: Performance on networks with Poisson degree distribution.

This sum is a normalizing constant that may be thought of as an indicator of network size—it is the sum of similar contributions from all nodes. In experiments we presented so far, we used the exact value of this global constant. Dependence on such global information is clearly not desirable for a decentralized algorithm.

A number of methods for decentralized estimation of global network parameters exist, but all produce local variance in the estimates. Such variance poses little problem for decentralized search algorithms such as EVN, however, because each decision is made locally, and thus estimates of the constant on different nodes need not be consistent. However, serious problems could arise if bias in estimates of the constant degraded the searchability of the network.

We repeated the experiments reported on scale-free networks with degree parameter 1.5 by distorting the normalizing constant with a multiplier of 0.001, 0.01, 0.1, 10, 100, and 1000 for all nodes in the network. Because each node makes its decisions independently of the other nodes, that the constant was distorted in the same way for all nodes is irrelevant. No performance decrements were observed in either performance measure, except when the multiplier was 0.001 or 0.01. Performance decrements for such low values of the multiplier are not unexpected—in these cases, the nodes operated in a 1000 node network, but made decisions as if the network had only a single node or ten nodes. We expect that such poor estimates of the normalizing constant would be avoided easily in practice.

4.2 Scientific Citation Network

We next present results on a real-world network: a citation graph of scientific papers. The nodes in our network were papers from the theoretical high-energy physics (hep-th) area of [arXiv.org](http://arxiv.org), an on-line archive of research papers. We included in our network papers that were published in 1995–2000 and had more than 50 non-self citations. The network included 833 nodes and 13,267 links.

Decentralized search in this citation graph is an artificial task—though it does resemble searching for a particular piece of information before the advent of search engines—but the results are useful in evaluating the applicability of EVN to a network that evolves naturally over time, with no known patterns of link formation.

We treated the citation graph as an undirected network, defining node similarity using paper titles and abstracts. The title and abstract of each paper were represented as weighted-term vectors using TFIDF (term frequency $\times$ inverse document frequency) weighting. Paper similarity was computed using a standard cosine correlation measure. We discretized this continuous similarity measure and for each discrete value it took, estimated q_{st} in Equation 2 in a straightforward manner from the network.

We conducted 10,000 randomly selected search tasks, terminating them after 100 hops if the target was not found. Table 1 shows our performance measures; Figure 4 shows the distribution of path lengths returned by each algorithm. Similarity-based navigation was not effective in this task, while degree-based navigation was competitive. EVN performed better than both in all performance measures. Further comparison of EVN and degree-based navigation revealed that EVN succeeded at all search tasks for which degree-based navigation failed and that there were no search tasks in which degree-based navigation succeeded but EVN did not. In those search tasks for which both algorithms succeeded, 47% of the time EVN returned a shorter path than degree-

Algorithm	<i>prop</i>	<i>path</i>	<i>median-path</i>	<i>opt-path</i>
Random	0.89	25.12	20	2.50
Similarity-Based	0.90	23.84	18	2.50
Degree-Based	0.93	9.86	5	2.51
EVN	0.99	6.07	4	2.55
Optimal	1.0	2.55	3	2.55

Table 1: Performance on hep-th citation graph.

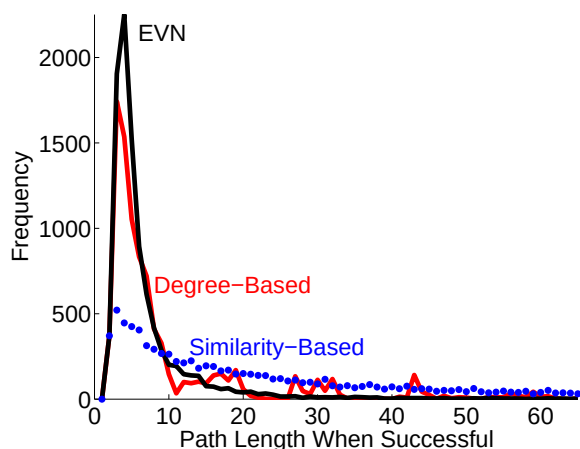


Figure 4: Path length distributions in hep-th citation graph.

based search (mean difference in path length = 11.02); 22% of the time degree-based search yielded a shorter path than EVN (mean difference in path length = 3.75); 31% of the time EVN and degree-based search returned paths of same length.

We next analyzed the sensitivity of the algorithm to the q_{st} values, which were estimated using the entire graph. We distorted q_{st} values (for each discrete value of node similarity) by multiplying them by 0.01, 0.1, 10, and 100. The performance measures were essentially unaffected—they were all within 0.2% of their values reported above.

5 Discussion

We presented a simple and principled algorithm for decentralized search in networks that show homophily and degree disparity. Our formulation of the problem allows one to consider all factors that may influence search performance. In that, it differs fundamentally from previous work in this area, while providing a unifying framework for existing algorithms, which are special cases of the algorithm we present here. Experimental results on a collection of synthetic and real-world networks indicate that our algorithm performs remarkably well, though it may be possible to achieve even better performance by using more sophisticated estimates of the statistical quantities involved.

The utility of our approach depends on the availability of statistical information regarding the relationship between node similarity and link formation. This information is typically available when a network is *designed* by an analyst who establishes the rules on how the network evolves over time (e.g., Zhang et al. [2002]). But we expect that this type of information would also be easy to obtain in other types of networks.

Acknowledgments

We would like to thank Jennifer Neville for providing the data on the hep-th scientific citation graph and to Cynthia

Loiselle for providing comments on earlier drafts of this paper. This research is supported by NSF and Lawrence Livermore National Laboratory (LLNL) under contract numbers HR0011-04-1-0013 and W-7405-ENG-48. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of NSF, LLNL, or the U.S. Government.

References

- [Adamic et al., 2001] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physical Review E*, 64, 2001.
- [Barabasi and Albert, 1999] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
- [Chakrabarti et al., 1999] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific Web resource discovery. In *Proceedings of the Eighth International World Wide Web Conference*, 1999.
- [Clarke et al., 2000] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, Lecture Notes in Computer Science 2009*, Berlin, 2000. Springer.
- [Diligenti et al., 2000] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*, pages 527–534, Cairo, Egypt, 10–14 September 2000.
- [Kleinberg, 2000a] J. Kleinberg. Navigation in a small world. *Nature*, 406:845, 2000.
- [Kleinberg, 2000b] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.
- [Kleinberg, 2001] J. Kleinberg. Small-world phenomena and the dynamics of information. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, 2001.
- [Travers and Milgram, 1969] J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.
- [Watts and Strogatz, 1998] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.
- [Watts et al., 2002] D. J. Watts, P. S. Dodds, and M. E. J. Newman. Identity and search in social networks. *Science*, 296, 2002.
- [Zhang et al., 2002] H. Zhang, A. Goel, and R. Govindan. Using the small-world model to improve Freenet performance. In *Proceedings of IEEE Infocom*, 2002.

Value Ordering for Finding All Solutions

Barbara M. Smith

Cork Constraint Computation Centre, Ireland.
b.smith@4c.ucc.ie

Paula Sturdy

University of Huddersfield, U.K.
p.sturdy@hud.ac.uk

Abstract

In finding all solutions to a constraint satisfaction problem, or proving that there are none, with a search algorithm that backtracks chronologically and forms k -way branches, the order in which the values are assigned is immaterial. However, we show that if the values of a variable are assigned instead via a sequence of binary choice points, and the removal of the value just tried from the domain of the variable is propagated before another value is selected, the value ordering can affect the search effort. We show that this depends on the problem constraints; for some types of constraints, we show that the savings in search effort can be significant, given a good value ordering.

1 Introduction

It is well-known that the order in which variables are instantiated can make an enormous difference to the search effort in solving a constraint satisfaction problem, whether just one solution is required, or all solutions. However, value ordering is relatively neglected, partly because no cheap general-purpose value ordering heuristics are known, and partly because it has been accepted that value ordering is not important if all solutions are required, or there is no solution.

Frost and Dechter [1995] showed that, with backjumping, the value ordering can have an effect on the search for all solutions, but say: “With backtracking the order in which values are chosen makes no difference on problems which have no solution, or when searching for all solutions.” Their argument is that when a node corresponding to a variable is created in the search tree, the children correspond to the values of that variable, and that the subtrees rooted at the children nodes are explored independently. To find all solutions, or to prove that there are none, every subtree must be explored, and the order makes no difference to the overall search.

This argument assumes that the search tree is formed by k -way branching [Mitchell, 2003], i.e. when a variable with k values in its domain is selected for instantiation, k branches are formed. Constraint solvers such as ILOG Solver and ECLⁱPS^e by default use a search strategy similar to the MAC algorithm [Sabin and Freuder, 1997], and in particular use 2-way, or binary, branching. When a variable x is

selected for instantiation, its values are assigned via a sequence of binary choices. If the values are assigned in the order $\{v_1, v_2, \dots, v_k\}$, the first choice point creates two alternatives, $x = v_1$ and $x \neq v_1$. The first (the left branch) is explored; if that branch fails, or if all solutions are required, the search backtracks to the choice point, and the right branch is followed instead. Crucially, the constraint $x \neq v_1$ is propagated, before a second choice point is created between $x = v_2$ and $x \neq v_2$, and so on. (The MAC algorithm also allows the possibility that on backtracking after trying $x = v_1$, a different variable could be assigned next, not just a different value: that possibility is not considered here.)

With binary branching, the subtrees resulting from successive assignments to a variable are not explored independently: propagating the removal of a value from the current variable’s domain on the right branch can lead to further domain reductions. This propagation can affect the search when future values of the variable are considered: indeed, sometimes a future value can be removed from the domain. Hence, the order in which the values are assigned can affect the search.

[Smith, 2000] gives an example of the value ordering affecting the search to find all solutions. Here, we present a more extensive investigation and explain how value ordering can affect search effort. We show that, depending on the problem constraints, the saving in search effort over k -way branching can be around 50%, given a good value ordering.

2 Search Trees for a Golomb Ruler Problem

First, we examine in detail the effect of the value ordering when finding all solutions, using a variant of the Golomb ruler problem (prob006 in CSPLib). A Golomb ruler with m marks may be defined as a set of m integers $0 = x_0 < x_1 < \dots < x_{m-1}$, such that the $m(m-1)/2$ differences $x_j - x_i$, $0 \leq i < j \leq m-1$, are distinct. The length of the ruler is x_{m-1} , and the objective is to find a minimum length ruler. Modeling the problem is discussed in [Smith *et al.*, 2000].

To create a problem with no solution, we set $x_{m-1} = \text{minl} - 1$, where minl is the minimum possible length. The model is chosen so that the search trees (using ILOG Solver’s default binary branching) for one instance of the problem are small enough to display, while the value ordering makes a significant difference to the number of backtracks.

The chosen model is not the quickest way to solve the problem, since it does expensive constraint propagation in

order to reduce search. Auxiliary variables represent the $m(m-1)/2$ differences between the marks on the ruler, defined by $d_{i,j} = x_j - x_i$, where $i < j$; generalized arc consistency (GAC) is enforced on these ternary constraints using ILOG Solver's table constraints. We also enforce GAC on the allDifferent constraint on the difference variables. Since x_i is the sum of i difference variables, its minimum value is set to be the sum of the first i integers. Similarly, the length x_{m-1-i} to x_{m-1} is at least the sum of the first i integers, and this reduces the maximum value of x_{m-1-i} . We add the constraint $d_{0,1} < d_{m-2,m-1}$ to break the reflection symmetry.

The search variables are $x_0, x_1, \dots, x_{m-1}$, assigned in that order. We compare two value ordering heuristics; choosing either the smallest value in the domain, or the largest. Choosing the largest value would not be a sensible strategy for finding a minimum length Golomb ruler, since the length is not known in advance, but here it shows the effect on search of changing the value ordering.

Figure 1 shows the search trees resulting from proving that there is no 6-mark Golomb ruler with length 16, with the two selected value orderings. The difference is striking, given the conventional view that the value ordering only reorders the search, and does not affect the overall search effort.

The black circles show where a failure is detected and the search backtracks. When values are assigned in increasing order, on backtracking to take the right branch the domain of the variable can sometimes be reduced, or eliminated altogether. We examine a case in detail. Initial constraint propagation reduces the domains of the search variables to $x_0 : 0$; $x_1 : \{1..5\}$; $x_2 : \{3..10\}$; $x_3 : \{6..13\}$; $x_4 : \{10..14\}$; $x_5 : 16$. On backtracking from $x_1 = 1$, the right branch has $x_1 \in \{2..5\}$. Constraint propagation reduces this to $x_1 = 2$: there are 14 pairwise differences between the marks in this instance (excluding $d_{0,5}$, which has already been assigned) and just 14 possible values for these differences, from 1 to 14. If $x_1 \neq 1$, then $x_4 \leq 13$, because of the symmetry constraint. Hence, the value 14 can only be assigned to the difference $x_5 - x_1$, which means that $x_1 = 2$. Enforcing GAC on the allDifferent constraint on the difference variables makes this inference, and the branch $x_1 \in \{3..5\}$ is not considered further. On the other hand, when values are assigned in decreasing order, the domain of x_1 on the right branch always contains the value 1, and the allDifferent constraint is already GAC.

n	Increasing order	Decreasing order	k -way branching
6	8	18	18
7	71	97	113
8	304	456	525
9	1,748	2,809	3,252
10	6,759	11,395	13,592
11	117,251	206,510	227,815

Table 1: Number of backtracks to prove that there is no Golomb ruler with n marks of length $\text{minl} - 1$, where minl is the minimum possible length, using binary branching with increasing or decreasing value ordering, or k -way branching.

Table 1 gives the results for larger instances of the same

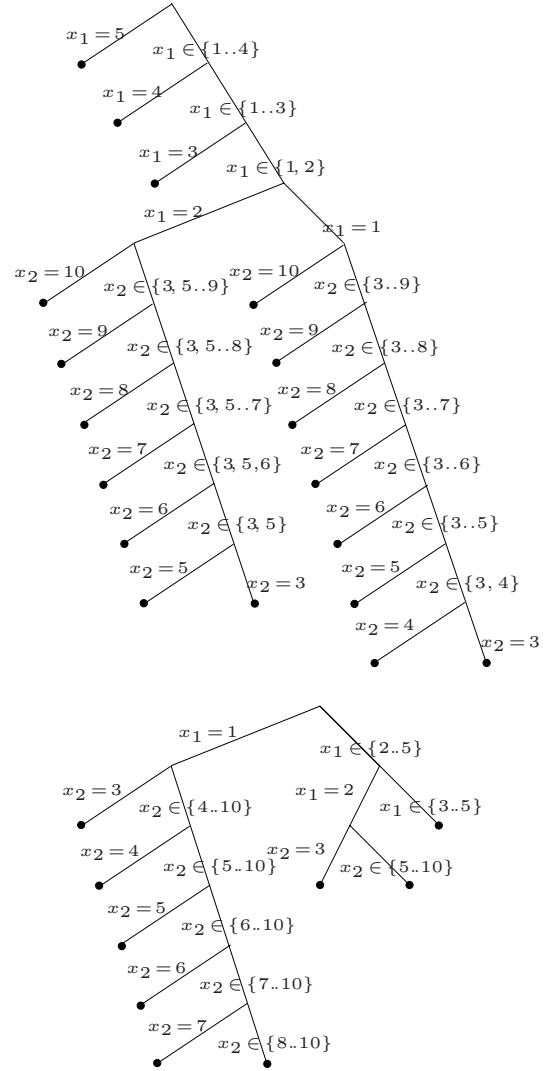


Figure 1: Search trees formed in proving that there is no Golomb ruler with 6 marks of length 16, using (top) decreasing value order and (bottom) increasing value order.

problem, and also shows the effect of using an implementation of k -way branching in ILOG Solver. k -way branching nearly doubles the number of backtracks compared with binary branching using increasing value order. Although decreasing order is much worse than increasing order, it still allows significantly less search than k -way branching for the larger instances.

The runtimes for k -way branching are usually shorter than for binary branching with decreasing value ordering, even when the search tree is larger. For $n = 11$, k -way branching takes 1390 sec. (on a 600MHz Celeron PC), compared to 1410 sec. for binary branching with decreasing value ordering. With increasing value ordering, binary branching is significantly faster (980 sec.).

The explanation given earlier for the immediate failure of the right branch at the choice point between $x_1 = 2$ and $x_1 \in \{3..5\}$ depends on the global allDifferent constraint.

However, even if the allDifferent constraint is treated as a clique of binary $\neq$ constraints and the constraints $d_{i,j} = x_j - x_i$ are not made GAC, choosing the smallest value in the domain requires fewer backtracks to prove insolubility than choosing the largest value. The crucial constraints are $x_0 < x_1 < \dots < x_{m-1}$ and the symmetry constraint $d_{0,1} < d_{m_2,m-1}$, which is equivalent to $x_1 < 16 - x_4$ in the example analysed. With monotonic binary constraints such as these, the removal of the largest or smallest value in the domain of one of the variables can reduce the domain of the other, and removing any other value has no effect [van Hentenryck *et al.*, 1992]. Given a constraint $x < y$ with x assigned before y , trying the values of x in increasing order means that on backtracking, the smallest value in the domain of x is removed on the right branch and this removal in turn reduces the domain of y . Since the variables are assigned in lexicographic order in this case, this explains why assigning the values in increasing order has the largest effect on the search.

3 Graceful Labeling of a Graph

In this section, we consider further the effect of value ordering when there are monotonic constraints, and the interaction of the variable ordering and the value ordering.

The problem is that of determining all graceful labelings of the graph shown in Figure 2. A labeling f of the nodes of a graph with q edges is *graceful* if f assigns each node a unique label from $\{0, 1, \dots, q\}$ and when each edge xy is labeled with $|f(x) - f(y)|$, the edge labels are all different.

A possible CSP model has a variable for each node, $x_1, x_2, \dots, x_n$, each with domain $\{0, 1, \dots, q\}$ and a variable for each edge, $d_1, d_2, \dots, d_q$, each with domain $\{1, 2, \dots, q\}$. The constraints are: if edge k joins nodes i and j then $d_k = |x_i - x_j|$; $x_1, x_2, \dots, x_n$ are all different, as are $d_1, d_2, \dots, d_q$. Since the edge variables must be assigned a permutation of the values 1 to q , it is worthwhile to enforce GAC on the all-Different constraint. The allDifferent constraint on the node variables is looser (9 variables and 17 possible values in the example of Figure 2) and this is expressed by $\neq$ constraints. The search variables are $x_1, x_2, \dots, x_n$.

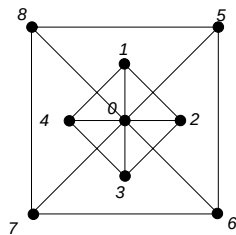


Figure 2: A graph that can be gracefully labeled. The node numbers correspond to the variables of the CSP.

Symmetrically equivalent solutions can be eliminated by adding constraints to the model [Petrie and Smith, 2003].

- $x_0 < 8$ (to eliminate the complement symmetry, i.e. the symmetry that replaces every value v by $q - v$).
- $x_1 < x_2$; $x_1 < x_3$; $x_2 < x_4$ (to eliminate rotations and reflections in the cycle consisting of nodes 1, 2, 3, 4).

- $x_5 < x_6$; $x_5 < x_7$; $x_6 < x_8$ (to eliminate symmetry in the cycle consisting of nodes 5, 6, 7, 8).
- $x_1 < x_5$ (to prevent interchanging the two cycles).

What propagation can follow from the removal of a value from the domain of a variable in this case? The (binary) symmetry constraints are monotonic and so a value removal will propagate as in the Golomb Rulers problem. Given a clique of $\neq$ constraints, as on the node variables, removing a value from the domain of a variable does not affect the other variables. However, if there is a global allDifferent constraint, as on the edge variables, removing a value may affect other domains; for instance, if there are only as many values as variables and a value occurs in the domain of only two variables, removing it from the domain of one means that it must be assigned to the other. However, the immediate effect on the search variables (the node variables) is limited, since GAC is not maintained on the ternary constraints linking the edge and node variables.

We have tried several different variable orders for this problem. In each case, if binary branching is used with a value ordering that chooses a value from the middle of the domain, it takes the same number of backtracks to find all solutions as k -way branching. However, if the value chosen is either the smallest or the largest in the domain, binary branching takes fewer backtracks than k -way branching. This suggests that the reduction in search is mainly due to the symmetry constraints: any propagation due to the other constraints would not be restricted to the largest and smallest value in the domain.

Table 2 shows the results of solving this problem with different variable and value orders. As well as lexicographic and reverse lexicographic variable order, we use the orders that we found to be respectively best and worst, when assigning the values in increasing order [Sturdy, 2003]. (Incidentally, the comparison also demonstrates that the value order cannot compensate for a poor variable order in finding all solutions.)

It is not necessary to use the same order (increasing or decreasing) for all variables. The rightmost columns in Table 2 give the results for a heuristic that, for most variable orders, assigns the values of x_1 and x_5 in increasing order, and those of the other variables in decreasing order. However, if x_5 comes after x_6 , x_7 and x_8 and before x_1 in the variable ordering, it is assigned in decreasing order, and if x_1 comes after x_2 to x_8 in the variable order, it is assigned in decreasing order. (Hence, when the variables are assigned in reverse lexicographic order, decreasing order for all variables is best.) Table 2 shows that the heuristic is at least as good as the better of increasing or decreasing order in all cases, and this was true for the other variable orders that we tried.

The heuristic was derived empirically by trying increasing or decreasing order for each variable, with a range of variable orders. We found that the effect of changing the value order for a single variable does not depend on the value order chosen for the other variables, so that the best order, for a given variable order, can be chosen independently for each variable. The heuristic can be related to the binary symmetry-breaking constraints: unless x_1 and x_5 are late in the variable order, assigning their values in ascending order will lead to reducing

Variable order	k -way branching		Increasing order		Decreasing order		Heuristic	
	bt.	sec.	bt.	sec.	bt.	sec.	bt.	sec.
$x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$	6043	0.21	3126	0.21	3793	0.21	2929	0.21
$x_8, x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0$	11963	0.46	10167	0.55	5304	0.37	5304	0.37
$x_8, x_0, x_4, x_3, x_1, x_5, x_2, x_6, x_7$	3129	0.15	1839	0.14	2374	0.16	1730	0.14
$x_1, x_7, x_5, x_3, x_2, x_8, x_0, x_6, x_4$	39469	1.57	31338	1.68	15871	1.04	15834	1.04

Table 2: Finding all graceful labelings of the graph in Figure 2, with k -way branching or binary branching with different value orderings.

the domains of other variables on backtracking. The variables assigned in decreasing order are those that appear mainly on the right of $>$ constraints; on backtracking, the largest value in the domain will be removed, and again this will reduce the domains of other variables.

4 Langford’s Problem

In the previous examples, assigning values in either increasing or decreasing order reduces the search effort considerably, in comparison with k -way branching, because of the monotonic binary constraints in the problem. In this section, a problem with no monotonic constraints is discussed.

Langford’s problem (prob024 in CSPLib) can be stated as follows: “A sequence of $n \times m$ integers consists of the integers 1 to m each appearing n times. There is one integer between consecutive occurrences of the integer 1, two integers between consecutive occurrences of the integer 2, and so on. Find all possible such sequences.”

Modeling this problem as a CSP is discussed in [Hnich *et al.*, 2004]. A possible model has nm variables, one for each location in the sequence; its value represents the integer at this location. Thus, the value i of d_j , $1 \leq j \leq nm$, is an integer in the interval $[1, m]$, representing the fact that occurrence $(i \div n) + 1$ of the integer $i \bmod n$ occurs at location j . For example, in the (3,9) instance, i.e. $n = 3$, $m = 9$, $d_1 = 1, d_3 = 2, d_5 = 3$ represent the 1st, 2nd and 3rd occurrences of the integer 1 appearing in positions 1, 3 and 5 of the sequence. A dual model of the problem has a variable x_i , $1 \leq i \leq nm$, for each occurrence of each integer: its value is the position in the sequence of this occurrence. The problem can be viewed as a permutation problem: any valid sequence assigns a permutation of the nm possible values to the variables $d_1, d_2, \dots, d_{nm}$. To ensure that any solution is a permutation of the values, both sets of variables are included in the model, with channeling constraints between them, i.e. $d_i = j$ iff $x_j = i$. We also add constraints to break the symmetry: given any solution, another can be found by reversing the sequence. We assign values to $d_1, d_2, \dots, d_{nm}$ in turn.

A value ordering heuristic for this problem can be devised by considering how removing a value from the domain of a variable d_j on backtracking will affect other variables. The removal will also remove a value from the domain of $n - 1$ other variables; for example, in the (3,9) instance, if $d_1 \neq 1$, then $d_3 \neq 2$ and $d_5 \neq 3$. However, the values 2 and 3 are only assigned as a consequence of assigning the value 1, and these deletions will not lead to further propagation. The other possible propagation is from the channeling constraints: as

discussed in [Hnich *et al.*, 2004], if a value appears in the domain of only one d_j variable, it will be assigned to that variable. Hence, if there is a value that appears in the domains of the current variable and only one other search variable, it should be chosen: on backtracking, propagation will assign the value to the other variable.

In [Smith, 2000], *dualsdf* ordering was used, which chooses the value appearing in fewest domains, i.e. corresponding to the dual variable with smallest domain. *dualsdf* was shown to give better results than increasing value order, for finding all solutions. It will always choose a value appearing in the domain of only one other search variable, if there is one; hence, we can now explain its good performance. Its anti-heuristic, *dualldf*, chooses the value corresponding to the dual variable with *largest* domain, and will not choose a value appearing in only two domains unless there is no alternative. Whereas *dualsdf* can be expected to give as much search reduction as possible in finding all solutions, *dualldf* will rarely be able to reduce the domains of other variables on the right branch and so should be nearly as bad as k -way branching for this problem. The results in Table 3 confirm this.

Instance	<i>dualsdf</i>	<i>dualldf</i>	k -way branching
(9,3)	182	208	210
(10,3)	562	619	631
(11,3)	2381	2697	2736

Table 3: Number of backtracks to find all solutions to three instances of Langford’s problem, using binary branching with different value orders, or k -way branching.

Hence, *dualsdf* reduces the search effort to find all solutions compared with k -way branching, and its good performance can be explained in relation to the problem constraints. However, the reduction in search is much less than in the previous examples, and does not make binary branching cost-effective in terms of run-time.

5 Symmetry Breaking During Search

We have shown that the effect of the value order on the search for all solutions depends on propagating the removal of the value just tried, on the right branch at binary choice points. In Symmetry Breaking During Search [Gent and Smith, 2000], constraints to avoid assignments symmetric to those already considered are added dynamically during search, on the right branch. These additional constraints can reinforce the effect of the value order on search. We demonstrate this in the n -

queens problem, whose symmetries are described in [Gent and Smith, 2000].

The basic program (from the ILOG Solver User Manual) has a variable $x_i, i = 1, 2, \dots, n$ for each row of the board, representing the queen on that row, with a value for each column. There are three allDifferent constraints, representing that the queens are on different columns and on different diagonals in each of the two possible directions. SBDS requires a specification of the effect of each symmetry on the assignment of a value to a variable. For instance, in the n -queens problem, the reflection in the top-left to bottom-right diagonal (symmetry $d1$) transforms the assignment $x_i = j$ to $x_j = i$. The first assignments made are $x_1 = 1$ and $x_3 = 2$ (the variable ordering heuristic chooses the variable with smallest domain, using the smallest value as a tie-breaker). On backtracking to take the alternative choice $x_3 \neq 2$, $d1$ is the only symmetry remaining, given the assignment $x_1 = 1$. SBDS adds the symmetric equivalent of $x_3 \neq 2$, i.e. $x_2 \neq 3$, to the right branch. Both constraints, i.e. $x_3 \neq 2$ and $x_2 \neq 3$, are propagated before choosing another value.

When using SBDS to eliminate the symmetry in the n -queens problem, we compare four value ordering heuristics. First, we select values in increasing order, as a basis for comparison. Second, we use a heuristic which seems likely to give most scope for the SBDS constraints to prune the domains of future variables. It is not obvious how best to do this, but the value that attacks the most unattacked squares will free these squares on backtracking: it seems plausible that this will increase the likelihood that the SBDS constraints remove one. The opposite heuristic, choosing the value that attacks fewest unattacked squares, is similar to the *promise* heuristic introduced by Geelen [1992]; hence, we call these two heuristics *anti-promise* and *promise* respectively. Finally, we use *dualsdf*, as in Langford’s problem, taking into account that $x_1, \dots, x_n$ must be assigned a permutation of the values 1 to n ; in a permutation problem, a global allDifferent constraint does at least as much domain pruning as channeling constraints would do, when a value is removed from the domain of one of the variables [Hnich *et al.*, 2004].

We first compare the four heuristics if the symmetry is *not* eliminated: the results are given in Table 4. The effect of the value ordering then depends on how the allDifferent constraints are treated. If they are treated as cliques of binary $\neq$ constraints, the value ordering can have no effect on the search effort. Table 4 gives the results for k -way branching only, since the four binary branching heuristics give the same number of backtracks, though a marginally longer runtime.

On the other hand, if the allDifferent constraints are treated as global constraints, *dualsdf* requires less search than k -way branching, and least search of the binary branching heuristics. However, the reduction in search, in comparison with k -way branching, is not great. Overall, if the symmetry is not eliminated, k -way branching and $\neq$ constraints give the fastest runtime. Maintaining GAC on the allDifferent constraints is time-consuming, especially when GAC has to be re-established after each individual value for a variable has been tried, as in binary branching.

Table 5 compares the four heuristics when the symmetry in the problem is eliminated using SBDS. We cannot com-

n	$\neq$ constraints k -way branching		GAC on allDifferent constraints			
	bt.	sec.	<i>dualsdf</i> bt.	sec.	k -way branching bt.	sec.
9	1111	0.04	854	0.03	858	0.03
10	5072	0.11	3831	0.14	3869	0.12
11	22124	0.41	16308	0.63	16478	0.54
12	103956	1.90	74514	3.05	75448	2.62
13	531401	9.79	366488	15.6	371298	13.5
14	2932626	53.9	1964642	84.8	1990925	73.8

Table 4: Number of backtracks (bt.) and runtime (on a 1.7GHz Pentium M PC) to find all solutions to the n -queens problem, if symmetry is not eliminated.

n	Increasing order	<i>anti-</i> <i>promise</i>	<i>promise</i>	<i>dualsdf</i>
9	144	144	158	148
10	649	617	674	647
11	2433	2375	2375	2384
12	11305	10872	11547	11252
13	52140	51045	52825	51452
14	292869	279844	295522	291825
15	1562199	1521670	1541946	1547019

Table 5: Number of backtracks to find all non-isomorphic solutions to the n -queens problem, using different value ordering heuristics.

pare the heuristics with k -way branching in this case, because SBDS depends on binary branching. *Anti-promise*, intended to allow propagation of the SBDS constraints to reduce search, is indeed the best of the four heuristics. *Dualsdf* is often a rather poor second best, suggesting that the SBDS constraints are more significant than the allDifferent constraints in allowing domain reductions on the right branch. The experience with this problem class shows that it is possible to design value ordering heuristics for finding all solutions by considering the propagation of the constraints added on the right branch during search, whether the removal of the value just tried or SBDS constraints. The differences in search effort are relatively small, however, and the runtimes for *promise*, *anti-promise* and *dualsdf* are very similar; increasing order is fastest, being simplest to implement.

The original *promise* heuristic [Geelen, 1992] was designed to find a first solution quickly, and it is worth noting that our version is much better than *anti-promise* in this respect. For $n \leq 100$, *promise* can usually find a first solution in just a few backtracks, whereas *anti-promise* becomes increasingly successful at avoiding solutions. For instance, when $n = 100$, *promise* finds a solution in 2 backtracks, whereas *anti-promise* takes more than 21 million. However, *promise* is slightly worse than *anti-promise* for finding all solutions.

6 Conclusions

Mitchell [2003] showed that binary branching can do much less search than k -way branching, due to the possibility that on backtracking to a choice point, the next assignment tried need not be another value (if there is one) of the same vari-

able. In practice, this is not provided by default in solvers such as ECLⁱPS^e and ILOG Solver, although the user can implement such a search strategy, and Sabin and Freuder [1997] allowed it in their description of the MAC algorithm.

We have shown that even when the search always assigns another value of the current variable on backtracking, binary branching offers another potential advantage. Before another value is assigned, the removal of the value just tried from the domain of the variable is propagated. This can lead to further domain reductions, which can in turn mean that a future value of the current variable is pruned, or that the search when a future value is assigned is reduced. As a result, given the same variable ordering, binary branching does no more search than k -way branching in finding all solutions, and may do considerably less, depending on the order in which values are tried.

The reduction in search effort is much less than Mitchell found, since if all the values of the current variable are tried in turn, the propagation of the removal of the value just tried will be subsumed by the propagation of the assignment of another value. Nevertheless, we have shown that binary branching can result in a 50% reduction in backtracks over k -way branching, with a reduction of around 30% in runtime.

We found the largest search savings in the problems with monotonic binary constraints, when assigning the values in increasing or decreasing order. A reasonable value ordering heuristic in that case seems to be that if a variable is constrained to be less than several other variables, and is assigned before them, its values should be tried in increasing order, and otherwise in decreasing order. In that way, on backtracking, removing the value just tried means reducing the range of the variable, and this propagates to the variables it constrains. Furthermore, propagating range reductions in monotonic binary constraints is cheap, so that the additional overhead in propagating the removal of a value before the assignment of the next is not large.

In other cases, although it is possible to choose good value orderings by considering how the problem constraints would propagate the removal of a value from the domain of a variable, the reduction in search effort is much less, and k -way branching is faster. The experience with these problems suggests that constraint propagation must be cheap and further domain reductions must be very likely in order for binary branching to be worthwhile. For instance, enforcing GAC on an allDifferent constraint fails on both counts: it is time-consuming, and removing a value from the domain of a single variable only occasionally leads to further domain reductions. As shown in the n -queens problem, when finding all solutions without breaking symmetry, it is better to use k -way branching than binary branching, if using global allDifferent constraints, or better still to use k -way branching with $\neq$ constraints.

Binary branching, with *any* value ordering, does no more search than k -way branching in finding all solutions, or proving that there are no solutions. However, even when binary branching does less search, k -way branching can be faster. Propagating the effect of removing the value just tried from the domain of the current variable can be time-consuming, and may not give sufficient reduction in search to be worthwhile. Our investigations have been concerned with finding

all solutions; however, when the extra propagation done by binary branching is not worthwhile for finding all solutions, it is unlikely to be worthwhile for finding one solution either. Although binary branching on successive values of the current variable is the default search strategy in some constraint solvers, k -way branching may often be a better choice.

Acknowledgements

We thank Gene Freuder, Deepak Mehta, Barry O'Sullivan and Nic Wilson for their helpful comments, and Peter van Beek for providing k -way branching code. This work was supported in part by the Science Foundation Ireland under Grant No. 00/PI.1/C075.

References

- [Frost and Dechter, 1995] D. Frost and R. Dechter. Look-ahead value ordering for constraint satisfaction problems. In *Proceedings IJCAI95*, pages 572–578, 1995.
- [Geelen, 1992] P. A. Geelen. Dual Viewpoint Heuristics for Binary Constraint Satisfaction Problems. In B. Neumann, editor, *Proceedings ECAI'92*, pages 31–35, 1992.
- [Gent and Smith, 2000] I. P. Gent and B. M. Smith. Symmetry Breaking During Search in Constraint Programming. In W. Horn, editor, *Proceedings ECAI'2000*, pages 599–603, 2000.
- [Hnich *et al.*, 2004] B. Hnich, B. M. Smith, and T. Walsh. Dual Models of Permutation and Injection Problems. *JAIR*, 21:357–391, 2004.
- [Mitchell, 2003] D. G. Mitchell. Resolution and constraint satisfaction. In F. Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003*, LNCS 2833, pages 555–569. Springer, 2003.
- [Petrie and Smith, 2003] K. E. Petrie and B. M. Smith. Symmetry Breaking in Graceful Graphs. Technical Report APES-56a-2003, APES Research Group, June 2003.
- [Sabin and Freuder, 1997] D. Sabin and E. C. Freuder. Understanding and Improving the MAC Algorithm. In G. Smolka, editor, *Principles and Practice of Constraint Programming - CP97*, LNCS 1330, pages 167–181. Springer, 1997.
- [Smith *et al.*, 2000] B. M. Smith, K. Stergiou, and T. Walsh. Using auxiliary variables and implied constraints to model non-binary problems. In *Proceedings AAAI-2000*, pages 182–187, 2000.
- [Smith, 2000] B. M. Smith. Modelling a Permutation Problem. Research Report 2000.18, School of Computer Studies, University of Leeds, June 2000.
- [Sturdy, 2003] P. Sturdy. Learning Good Variable Orderings. Technical Report APES-64-2003, APES Research Group, June 2003. Available from <http://www.dcs.st-and.ac.uk/~apes/apesreports.html>.
- [van Hentenryck *et al.*, 1992] P. van Hentenryck, Y. Deville, and C.-M. Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57:291–321, 1992.

Leaf-Value Tables for Pruning Non-Zero-Sum Games

Nathan Sturtevant

University of Alberta Department of Computing Science
Edmonton, AB Canada T6G 2E8
nathanst@cs.ualberta.ca

Abstract

Algorithms for pruning game trees generally rely on a game being zero-sum, in the case of alpha-beta pruning, or constant-sum, in the case of multi-player pruning algorithms such as speculative pruning. While existing algorithms can prune non-zero-sum games, pruning is much less effective than in constant-sum games. We introduce the idea of leaf-value tables, which store an enumeration of the possible leaf values in a game tree. Using these tables we are can make perfect decisions about whether or not it is possible to prune a given node in a tree. Leaf-value tables also make it easier to incorporate monotonic heuristics for increased pruning. In the 3-player perfect-information variant of Spades we are able to reduce node expansions by two orders of magnitude over the previous best zero-sum and non-zero-sum pruning techniques.

1 Introduction

In two-player games, a substantial amount of work has gone into algorithms and techniques for increasing search depths. In fact, all but a small fraction of computer programs written to play two-player games at an expert level do so using the minimax algorithm and alpha-beta pruning. But, the pruning gains provided by alpha-beta rely on the fact that the game is zero-sum. Two-player games most commonly become non-zero-sum when opponent modeling is taken into consideration. Carmel and Markovitch [1996] describe one method for pruning a two-player non-constant-sum game.

A multi-player game is one with three or more players or teams of players. Work on effective pruning techniques in multi-player games began with shallow pruning [Korf, 1991], and continued most recently with speculative pruning, [Sturtevant, 2003]. While a game does not need to be constant-sum for pruning to be applied, the amount of pruning possible is greatly reduced if a game is not constant-sum.

Both two-player and multi-player pruning algorithms consist of at least two stages. They first collect bounds on

players' scores, and secondly test to see if, given those bounds, it is provably correct to prune some branch of the game tree. This work focuses on the second part of this process. Alpha-beta and other pruning methods use very simple linear tests as a decision rule to determine whether or not they can prune. For zero-sum games these decisions are optimal. That is, they will make perfect decisions about whether pruning is possible. For non-zero-sum games, however, current techniques will not catch every possibility for pruning.

In order to prune optimally in a non-zero-sum game we must have some knowledge about the space of possible values that can occur within the game tree. Carmel and Markovitch assume a bound on the difference of player scores. We instead assume that we can enumerate the possible outcomes of the game. This is particularly easy in card games, where there are relatively few possible outcomes to each hand. Given that we can enumerate possible game outcomes, we can then make optimal pruning decisions in non-zero-sum games. In addition, these techniques can be enhanced by incorporating information from monotonic heuristics to further increase pruning.

In section 2 we illustrate the mechanics of a few pruning algorithms, before moving to a concrete example from the game of Spades in section 3. In section 4 we examine the computational complexity and the leaf-value table method that is used to implement different decision rules for pruning, followed by experimental results and conclusions.

2 Pruning Algorithms

To prune a node in a game tree, it must be proven that no value at that node can ever become the root value of the tree. We demonstrate how this decision is made in a few algorithms.

2.1 Alpha-beta

We assume that readers are familiar with the alpha-beta prun-

```
maxValue(state,  $\alpha$ ,  $\beta$ )
  IF cutoffTest(state) return eval(state)
  FOR EACH  $s$  in successor(state)
     $\alpha \leftarrow \max(\alpha, \text{minValue}(s, \alpha, \beta))$ 
    IF ( $\beta - \alpha \leq 0$ )† RETURN  $\beta$ 
  RETURN  $\alpha$ 
```

Figure 1: Alpha-beta pseudo-code.

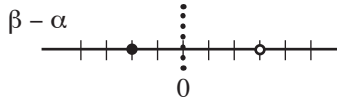


Figure 2: Alpha-beta pruning decision space.

ing algorithm. In Figure 1 we show pseudo-code for the maximizing player, modified slightly from [Russell and Norvig, 1995]. The statement marked † is where alpha-beta determines whether a prune is possible. This is illustrated in Figure 2. The x -axis is the value $\beta - \alpha$, and we plot points as they are tested. If a point, such as the solid one, falls to the left of 0, we can prune, and if it falls to the right, like the hollow point, we can't. It is useful to think of this as a *linear classifier*, because it makes a single comparison with a linear function to determine if a prune is possible.

2.2 Maxⁿ

The maxⁿ algorithm [Luckhardt and Irani, 1986] is a generalization of minimax for any number of players. In a maxⁿ tree with n players, the leaves of the tree are n -tuples, where the i th element in the tuple is the i th player's score or utility for that position. At the interior nodes in the tree, the maxⁿ value of a node where player i is to move is the maxⁿ value of the child of that node for which the i th component is maximum. At the leaves of a game tree an exact or heuristic evaluation function can be applied to calculate the n -tuples that are backed up in the game tree.

We demonstrate this in Figure 3. In this tree there are three players. The player to move is labeled inside each node. At node (a), Player 2 is to move. Player 2 can get a score of 3 by moving to the left, and a score of 1 by moving to the right. So, Player 2 will choose the left branch, and the maxⁿ value of node (a) is (1, 3, 5). Player 2 acts similarly at node (b) selecting the right branch, and at node (c) breaks the tie to the left, selecting the left branch. At node (d), Player 1 chooses the move at node (c), because 6 is greater than the 1 or 3 available at nodes (a) and (b).

2.3 Maxⁿ Pruning Algorithms

Given no information regarding the bounds on players' scores, generalized pruning is not possible. But, if we assume that each player's score has a lower bound of 0, and that there is an upper bound on the sum of all players scores, $maxsum$, we can prune. These bounds do not guarantee that a game is constant-sum, and existing pruning algorithms may miss pruning opportunities if a game is not constant-sum.

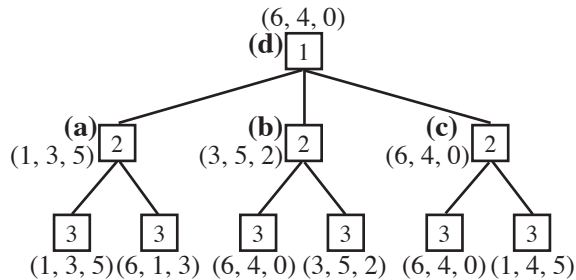


Figure 3: A 3-player maxⁿ game tree

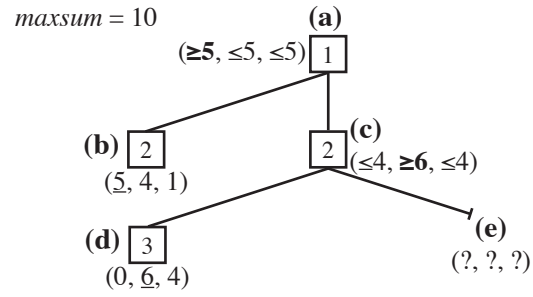


Figure 4: Shallow pruning in a 3-player maxⁿ tree.

2.3.1 Shallow Pruning

Shallow pruning [Korf, 1991] is one of the simplest pruning algorithms for multi-player games. An example of shallow pruning is shown in Figure 4. The sum of players' scores in each maxⁿ value is always 10, so $maxsum$ is 10. In this tree fragment Player 1 is guaranteed at least a score of 5 at node (a) by moving towards (b). Similarly, Player 2 is guaranteed 6 points at (c) by moving towards (d). Regardless what the unseen value is at (e), Player 2 will not select that move unless it gives him more than 6 points. Thus, before exploring (e), we can already guarantee that Player 1 will never get more than $maxsum - 6 = 4$ points at (c). Because Player 1 is guaranteed at least 5 points at (a), the value of (e) is irrelevant and can be pruned.

In this example we used consecutive bounds on Player 1 and Player 2's scores to prune. As stated previously, the general idea is to prove that unseen leaf nodes cannot become the maxⁿ value of the game tree. In Figure 4, for instance, we can consider all possible values which might occur at node (e). If any of these can ever become the maxⁿ value at the root of the tree, we cannot prune.

To formalize this pruning process slightly, we construct a n -tuple similar to a maxⁿ value called the *bound vector*. This is a vector containing the lower bounds on players' scores in a game tree given the current search. While the maxⁿ value is constrained to sum to $maxsum$, the bound vector can sum to as much as $n \cdot maxsum$. In Figure 4, after exploring every node except node (e) our bound vector is (5, 6, 0). This is because Player 1 is guaranteed 5 points at the root, and Player 2 is guaranteed 6 points at node (c). Existing pruning algorithms prune whenever the sum of the components in the bound vector is at least as large as $maxsum$.

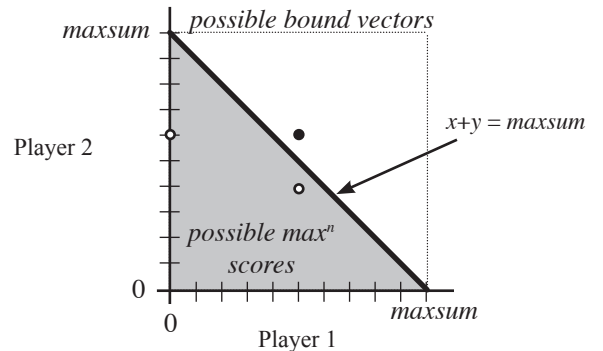


Figure 5: Shallow pruning decision space.

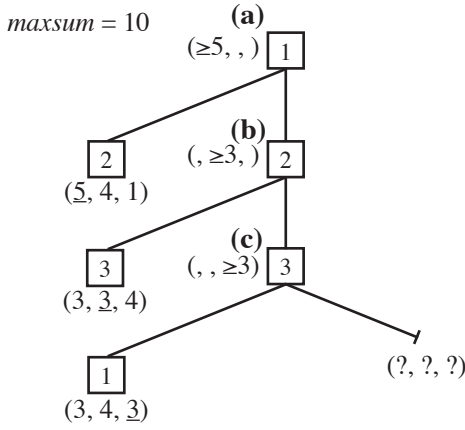


Figure 6: Speculative Pruning

We show a visualization of the shallow pruning space in Figure 5. The x - and y -coordinates are scores or bounds for Player 1 and Player 2 respectively. The shaded area is the space where possible $\max^n$ values for Player 1 and 2 will fall. All $\max^n$ values in the game must be in this space, because their sum is bounded by $\maxsum$. Because shallow pruning ignores Player 3, Player 1 and Player 2's combined scores are not constant-sum, so they can fall anywhere to the left of the diagonal line. Bound vectors, however, can fall anywhere in the larger square. If a bound vector is on or above the diagonal line defined by $x+y = \maxsum$, we can prune, because there cannot be a $\max^n$ value better than those bound vectors. Like alpha-beta, shallow pruning is using a linear classifier to decide when to prune.

Ignoring Player 3's values, we plot the leaf values (5, 4, -) and (0, 6, -) from Figure 4 as open points, and the bound vector (5, 6, -) used to prune in Figure 4 as a solid point. In this instance, there is no gap between the gray region and the diagonal line, so the line defined by $x+y = \maxsum$ is a perfect classifier to determine whether we can prune.

2.3.2 Alpha-Beta Branch-and-Bound Pruning

Although shallow pruning was developed for multi-player games, the basic technique only compares the scores from two players at a time. Alpha-beta branch-and-bound pruning [Sturtevant and Korf, 2000] is similar to shallow pruning, except that it uses a monotonic heuristic to provide bounds for all players in the game. The bound vector in Figure 4 was (5, 6, 0). Player 3's bound was 0, because we had no information about his scores. But, supposing we had a monotonic heuristic that guaranteed Player 3 a score of at least 2 points. Then the bound vector would be (5, 6, 2). This additional bound makes it easier to prune, since we can still prune as soon as the values in the bound vector sum to $\maxsum$.

2.3.3 Speculative Pruning

Speculative pruning [Sturtevant, 2003], like alpha-beta branch-and-bound pruning, takes into account all of the players in the game. It does this by considering multiple levels in the game tree at one time. We demonstrate this in Figure 6. In this figure, the important bounds are Player 1's bound at (a), Player 2's bound at (b) and Player 3's bound at (c). These together form the bound vector (5, 3, 3). If these values sum

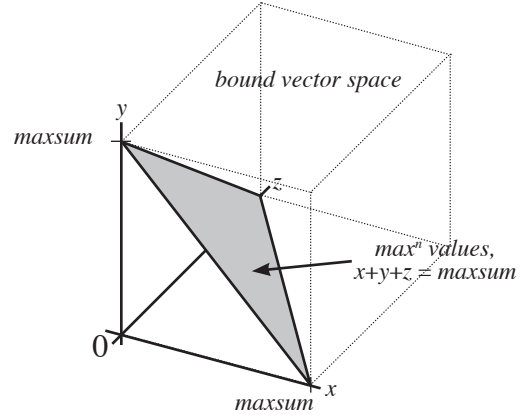


Figure 7: Speculative pruning decision space.

to at least $\maxsum$ we can guarantee that there will never be a value at the right child of (c) which can become the $\max^n$ value of the tree. When pruning over more than 2-ply in multi-player games there is the potential that while a value at (c) cannot become the $\max^n$ value of the tree, it can affect the $\max^n$ value of the tree. Other details of the speculative pruning algorithm prevent that from happening, but we are only concerned with the initial pruning decision here.

We illustrate the pruning decision rule for speculative pruning in Figure 7. In this case, because we are comparing three player's scores, the decision for whether we can prune depends on a 2-d plane in 3-d space, where each axis corresponds to the score bound for each of the 3 players in the game. Thus each $\max^n$ value and bound vector can be represented as a point in 3-d space. For a three-player constant-sum game, all possible $\max^n$ values must fall exactly onto the plane defined by $x+y+z = \maxsum$, which is also perfect classifier for determining when we can prune. As in shallow pruning, bound vectors can fall anywhere in the 3-d cube.

2.4 Generalized Pruning Decisions

In all the pruning algorithms discussed so far, a linear classifier is used to make pruning decisions. This works well when a game is zero-sum or constant-sum, but in non-constant-sum games a linear classifier is inadequate¹.

When a game is non-constant-sum, the boundary of the space of $\max^n$ values is not defined by a straight line. We demonstrate this in the next section using examples from the game of Spades. To be able to prune optimally, we need to know the exact boundary of feasible $\max^n$ values, so that we can always prune when given a bound vector outside this region. We explain methods to do this in Section 4.

3 Sample Domain: Spades

Spades is a card game for 2 or more players. In the 4-player version players play in teams, while in the 3-player version each player is on their own, which is what we focus on. There are many games similar to Spades which have similar proper-

¹Further details on the relationship between constant-sum and non-constant-sum games are found in Appendix A, but they are not necessary for understanding the contributions of this paper.

# Tricks Taken	Utility/Evaluation (bid)			Ranked max ⁿ vals
	P1 (1)	P2 (1)	P3 (2)	
(0, 0, 3)	0	0	19+6	(0, 0, 2)
(0, 1, 2) [†]	0	10+3	20+3	(0, 2, 1)
(0, 2, 1) [†]	0	9+6	0	(0, 4, 0)
(0, 3, 0) [†]	0	8+6	0	(0, 3, 0)
(1, 0, 2)	10+3	0	20+3	(2, 0, 1)
(1, 1, 1) [†]	10+3	10+3	0	(2, 2, 0)
(1, 2, 0) [†]	10+3	9+3	0	(2, 1, 0)
(2, 0, 1)	9+6	0	0	(4, 0, 0)
(2, 1, 0) [†]	9+3	10+3	0	(1, 2, 0)
(3, 0, 0)	8+6	0	0	(3, 0, 0)

Table 1: Outcomes for a 3-player, 3-trick game of Spades.

ties. We will only cover a subset of the rules here. A game of Spades is split into many hands. Within each hand the basic unit of play is a trick. At the beginning of a hand, players must bid how many tricks they think they can take. At the end of each hand they receive a score based on how many tricks they actually took. The goal of the game is to be the first player to reach a pre-determined score, usually 300 points.

If a player makes their bid exactly, they receive a score of $10 \times \text{bid}$. Any tricks taken over your bid are called overtricks. If a player takes any overtricks they count for 1 point each, but each time you accumulate 10 overtricks, you lose 100 points. Finally, if you miss your bid, you lose $10 \times \text{bid}$. So, if a player bids 3 and takes 3, they get 30 points. If they bid 3 and take 5, they get 32 points. If they bid 3 and take 2, they get -30 points. Thus, the goal of the game is to make your bid without taking too many overtricks.

If all players just try to maximize the number of tricks they take, the game is constant-sum, since every trick is taken by exactly one player, and the number of tricks available is constant. But, maximizing the tricks we take in each hand will not necessarily maximize our chances of winning the game, which is what we are interested in. Instead, we should try to avoid overtricks, or employ other strategies depending on the current situation in the game.

In a game with 3 players and t tricks, there are $(t+1)(t+2)/2$ possible ways that the tricks can be taken. We demonstrate this in Table 1, for 3 players and 3 tricks.

Table 1 is an example of a *leaf-value table*. It contains all possible leaf values and their associated utility in the game. In Spades, we build such a table after each player had made their bid, but before game play begins. The first column enumerates all possible ways that the tricks can be taken by each player. The second through fourth columns evaluate the score for each player, that is their utility for each particular outcome. In this example, Player 1 and Player 2 have bid 1 trick each, and Player 3 bid 2 tricks. If a player does not make their bid they have a score of 0, otherwise we use a heuristic estimate for their score, $10 \times \text{bid} - \text{overtricks} + 3 \times (\text{how many opponents miss their bid})$. As has been shown for minimax [Russell and Norvig, 1995], we only care about the relative

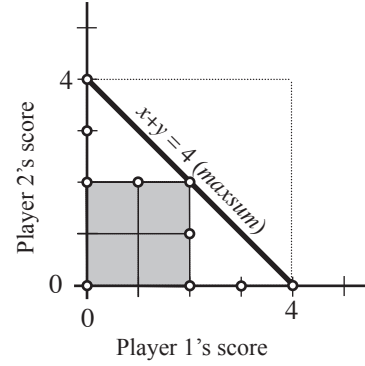


Figure 8: Pruning decision space for Table 1.

value of each state, not the absolute value. So, in the last column we have replaced each player's utility with a rank, and combined all player's ranks into the final maxⁿ value.

As an example, the first possible outcome is that Players 1 and 2 take no tricks, while Player 3 takes 3 tricks. Since Players 1 and 2 both missed their bids, they get 0 points. Player 3 made his bid of 2 tricks, and took 1 overtrick. Since we want to avoid too many overtricks, we evaluate this as $20 - 1 = 19$ points. But, since both Player 1 and 2 miss their bids in this scenario, Player 3 has a bonus of 6 points. This is the best possible outcome for Player 3, so it gets his highest ranking. For Player 1 and 2 it is their worst possible outcome, so they rank it as 0.

We graph the shallow pruning decision space for the first two players of this game in Figure 8. In this game *maxsum* is 4. The 10 possible leaf values for the first two players are all plotted as hollow points in the graph. If we use *maxsum* as a discriminator to decide if we can prune, it will indicate that we can only prune if a bound vector falls on or above the bold diagonal line. But, we can actually prune as long a bound vector is on or above the border of the gray region. So, we can actually prune given any bound vector for player 1 and 2 except (0, 0), (0, 1), (1, 0) and (1, 1).

As a final point, we note that in card games like Spades the number of tricks you have taken can only increase monotonically, which can be used as a heuristic to help pruning. This observation is the key behind the alpha-beta branch-and-bound pruning technique. But, if we are using a utility function like in Table 1, it is difficult to describe how the monotonic heuristic relates to the evaluation function. Leaf-value tables, however, make the task easy.

Assume, for instance, that Player 2 has already taken 1 trick. Then, we can ignore all outcomes in which he does not take one trick. This gives us a reduced set of values, which are marked with a [†] in Table 1. Looking at the associated maxⁿ values, we then see that Player 1 will get no more than 2 and Player 3 will get no more than 1. We show how this is used in the next section.

4 Leaf-Value Tables

The formal definition of a leaf-value table is a table which holds all possible outcomes that could occur at a leaf node in a game. Looking back to the pruning algorithms in Section 2,

	GLOBAL <i>leaf-value table</i> [] { <i>outcome</i> [], <i>rank</i> [] }
1	canLeafValueTablePrune(bounds[], heuristicUB[])
2	IF (inHashTable(bounds, heuristicUB))
3	return hashLookup(bounds, heuristicUB)
4	FOR each <i>entry</i> in <i>leaf-value table</i>
5	FOR <i>i</i> in players 1.. <i>n</i>
6	if <i>entry.outcome</i> [<i>i</i>] > heuristicUB[<i>i</i>]
7	skip to next <i>entry</i> ;
8	FOR <i>i</i> in players 1.. <i>n</i>
9	if <i>entry.rank</i> [<i>i</i>] ≤ bounds[<i>i</i>]
10	skip to next <i>entry</i> ;
11	addToHashTable(<i>false</i> , bounds, heuristicUB)
12	RETURN <i>false</i> ;
13	addToHashTable(<i>true</i> , bounds, heuristicUB)
14	RETURN <i>true</i> ;

Figure 9: Leaf-Value Table Pruning pseudo-code.

we want to replace the linear classifiers with more accurate classifiers, given a leaf-value table. Thus, we need an efficient way to both find and compute regions like in Figure 8 to determine when we can prune.

Theorem: The information stored in a leaf-value table is sufficient to make optimal pruning decisions.

Proof: We first assume that we have a bound vector $b = (b_1, b_2, \dots, b_n)$, where each bound in the vector originates on the path from the root of the tree to the current node. We are guaranteed that a player i with bound b_i will not change his move unless he can get some value v_i where $v_i > b_i$. Thus, if there exists a value $v = (v_1, v_2, \dots, v_n)$ where $\forall_i v_i > b_i$ then we cannot prune, because v is better than every bound in the search so far. Given a leaf-value table, we know every possible value in the game, and so we can explicitly check whether some v exists that meets the above conditions, and thus we can use a leaf-value table to make optimal pruning decisions. $\square$

Because a leaf-value table gives us an exact description of the boundary of the spaces where we can and cannot prune, the only question is how we can use this information efficiently. Given a bound vector, we need to quickly determine whether or not we can prune, because we expect to ask this question once for every node in the game tree.

For small games, we can build a lookup table enumerating all possible bound vectors and whether or not we can prune. This will provide constant time lookup, but for a table with t entries and n players, this table will be size $O(t^n)$ and take time $O(t^{n+1})$ to compute. Including heuristic information makes these tables even larger. But, most of these entries will never be accessed in any given game. Instead, we can dynamically compute entries as we need them, and store the

results in a hash table.

Pseudo-code for using a leaf-value table is in Figure 9. This procedure is called by a pruning algorithm to decide whether or not to prune given current bounds. A leaf-value table is pre-computed with each possible outcome of the game and the associated rank of that outcome for each player. We pass in the current bound vector, as well as any heuristic upper bounds on players’ scores. If an entry in the leaf-value table is inconsistent with the heuristic (line 6), we can ignore that entry, because that outcome cannot occur in the current sub-tree. If there is an entry in the table for which every player does better than their value in the bound vector, tested on lines 8-9, we reach lines 11-12, indicating we cannot prune.

Every time we attempt to prune, we will pay at most $O(\text{table size})$, but this cost is quickly amortized over the look-ups, and doesn’t add significant overhead. In a game with a large number of outcomes there are a variety of other methods that could be used to reduce the lookup cost. Additionally, we can always use the standard linear *maxsum* check, as it will always be correct if it indicates we can prune. The heuristic does not speed up the check for whether we can prune, but it can reduce the effective size of the leaf-value table, making it more likely that we do prune.

5 Experimental Results

5.1 Nodes Expanded in Three-Player Spades

We use the game of Spades as a test-bed for the techniques introduced in this paper. Our first experiment will illustrate that leaf-value tables are quite effective for pruning, and will also help demonstrate when they will be most effective. To do this, we compare the number of nodes expanded using (1) previous pruning techniques and (2) a variety of evaluation functions in the 3-player version of Spades. One of these evaluation functions is actually from a different bidding game, called “Oh Hell!”

For each method, we counted the total number of node expansions needed to compute the first move from 100 hands of Spades, where each player started with 9 cards and searched the entire game tree (27-ply). Our search used a transposition table, but no other search enhancements besides the pruning methods explicitly discussed. Bids for each player are pre-determined by a simple heuristic. For these trees, the leaf-value tables contain 55 entries. We present the results in Table 2.

The first column in the table is the average size of the game trees without pruning, 2.7 million nodes. This is determined by the cards players hold, not the evaluation function used, so we will compare the other tree sizes to this value.

The next two columns are the results using speculative pruning with a linear classifier, the previous best pruning technique. If we use a non-constant-sum (NCS) evaluation

	<i>Full Tree</i>	<i>NCS</i>	<i>MT</i>	<i>MoMB</i>	<i>mOT</i>	<i>smOT</i>	<i>WL</i>	<i>OH</i>
<i>nodes expanded</i>	2.7M	1.9M	1.1M	400k	37.2k	8.6k	788	40.7k
<i>reduction factor</i>	-	1.36	2.37	6.6	71	308	3362	65

Table 2: Overall reduction and average tree sizes in Spades.

	Full Tree	Best NZS	Zero-Sum	LVT
<i>nodes</i>	481k	225k	14k	22k
<i>reduction</i>	-	2.14	34.4	21.9

Table 3: Reduction and average tree sizes in 2-player Spades.

function, speculative pruning is able to reduce the average tree size by a factor of 1.36 to 1.9 million nodes. Maximizing tricks, *MT*, is the best-case evaluation function for speculative pruning, because it is constant-sum. In this case the average tree size is reduced to 1.1 million nodes.

Now, we replace speculative pruning’s linear classifier with leaf-value tables and use a non-zero-sum evaluation. The first function we use is *MoMB*, maximizing the number of opponents that miss their bid. This is quite similar to the strategy of maximizing your tricks, but the average tree size can be reduced further to 400k nodes, a 6.6 fold reduction.

The next evaluation function we use tries to minimize overtricks, *mOT*. This produces much smaller trees, 37.2k nodes on average, a 71 fold reduction over the full tree. A similar evaluation function, *smOT*, allows a slight margin for taking overtricks, because that is how we keep our opponent from making their bid, but tries to avoid too many overtricks. This evaluation function reduces the tree further to 8.6k nodes, over 300 times smaller than the original tree. The *smOT* evaluation is what was used with speculative maxⁿ for the *NCS* experiment. In the end, both algorithms calculate the exact same strategies, but with leaf-value tables we can do it hundreds of times faster.

Finally, we show a very simple evaluation function, *WL*. This function gives a score of 1 (*win*) if we make our bid and 0 (*loss*) if we miss it. In practice we wouldn’t want to use this evaluation function because it is too simple, but it does give an estimate of the minimum tree size. With this evaluation function the average tree has 788 nodes.

“Oh Hell!” (*OH*) is a similar game to Spades, however the goal of this game is to get as close to your bid as possible. In the context of this paper, we can just view this a different evaluation function in the game of Spades. Using this evaluation, the average tree size was 40.7k nodes, 65 times smaller than the full tree.

Besides showing the effectiveness of leaf-value tables, these experiments help illustrate two reasons why leaf-tables are effective for pruning in a non-constant-sum game. The first reason for large reductions is that the non-constant-sum evaluation may significantly reduce the number of unique outcomes in a game, which will be captured by a leaf-value table. The best example of this is the *WL* evaluation function. But, *smOT* also reduces the possible outcomes over *mOT*, and thus reduces the size of the game trees.

The other factor that is important for pruning is having a monotonic heuristic along with an evaluation function that is not monotonic with respect to the heuristic. Evaluation functions like *mOT* are non-monotonic in the number of tricks taken, because we initially want to take more tricks, and then, after making our bid, we don’t want to take any more tricks. This allows a monotonic heuristic to more tightly constrain

	Avg. Points	% Wins
LVT	263	62.3%
<i>prev. methods</i>	226	37.7%

Table 4: Average score over 100 games of Spades.

the search space, and thus increases pruning.

The *MoMB* evaluation function is both monotonic and only a slight simplification of the *MT* evaluation function, so we should and do see the least gains when using this evaluation function.

5.2 Nodes Expanded in Two-Player Spades

We conducted similar experiments in the two-player game of Spades. [Korf, 1991] described how deep pruning fails in multi-player games. It is not difficult to show that the same problem exists in two-player non-zero-sum games as we have described them here. The bottom line is that we cannot prune as efficiently as alpha-beta once we use a non-zero-sum evaluation function. In these experiments we did not apply every conceivable game-tree reduction technique, only transposition tables and basic alpha-beta pruning, so in practice we may be able to generate smaller two-player zero-sum game trees.

In the two-player Spades games, we searched 100 hands to depth 26 (13 tricks) using a variety of techniques. The results are in Table 3. The full tree averaged 481k nodes. Using a non-zero-sum evaluation function and the best previous methods produced trees that averaged 225k nodes. Using alpha-beta pruning and a zero-sum evaluation function reduced the trees further to 14k nodes on average. Using leaf-value tables (LVT) for pruning produced trees were slightly larger, 22k nodes. As referenced above, these trees are larger than those generated by alpha-beta because we cannot apply deep pruning, but they are still much smaller than previously possible given a non-zero-sum evaluation function.

5.3 Quality of Play From Extended Search

Finally, to compare the effect of additional search on quality of play, we then played 100 games of 3-player Spades, where multiple hands were played and the scores were accumulated until one player reached 300 points. Also, each complete game was replayed six times, once for each possible arrangement of player types and orderings, excluding games with all of one type of player. Each player was allowed to expand 2.5 million nodes per turn. One player used speculative pruning with leaf-value tables to prune, while the other used speculative pruning with a linear classifier. Hands were played “open” so that players could see each others cards. The results are in Table 4. The player using the leaf-value tables (LVT) was able to win 62.3% of the games and averaged 263 points per game, while the player using previous techniques averaged only 226 points per game.

5.4 Summary

We have presented results showing that leaf-value tables, when combined with previous pruning techniques, can effectively prune non-constant-sum games such as Spades or

Oh Hell!

Although we do not present experimental results here, we can predict the general nature of results in other games such as Hearts. In most situations in Hearts, our evaluation function will be constant-sum. But, there will be some situations where a non-constant-sum evaluation function is needed. Thus, if we use leaf-value tables for such a game, we will have the same gains as previous techniques in portions of the game that are constant-sum. But, when the game is non-constant sum, we will benefit from additional pruning, although the exact amount will depend on the particular situation.

6 Conclusions and Future Work

In this paper we have shown how an enumeration of possible leaf-values in a game tree, called a leaf-value table, can be used to change the linear classifier used in classical pruning algorithms to an arbitrary classifier needed for non-zero-sum games. This technique works particularly well in card games like Spades, where we see up to a 100 fold reduction of nodes expanded over the previous best results using a constant-sum evaluation function, along with gains in quality of play.

This work expands the limits of how efficiently we can search two-player and multi-player games. To a certain extent there is still an open question of how to best use limited resources for play. Recent results in applying opponent modeling to two-player games have not been wildly successful [Donkers, 2003]. In multi-player games, we have shown that deep search isn't always useful, if there isn't a good opponent model available [Sturtevant, 2004]. But, given a reasonable opponent model, this work allows us to use the best evaluation function possible and still search to reasonable depths in the game tree.

In the future we will continue address the broader question of what sort of opponent models are useful, and what assumptions we can make about our opponents without adversely affecting the performance of our play. The ultimate goal is to describe in exactly which situations we should use $\max^n$, and in which situations we should be using other methods. These are broad questions which we cannot fully answer here, but we these additional techniques will provide the tools to better answer these questions.

Acknowledgements

This research benefited from discussions with Jonathan Schaeffer, Michael Bowling, Martin Müller, Akihiro Kishimoto and Markus Enzenberger. The support of Alberta's Informatics Circle of Research Excellence (iCORE) is also appreciated.

A Game Transformations

In this paper we have often made distinctions between games or evaluation functions based on whether they are constant-sum or not. These distinctions can be blurred through minor transformations, however such transformations do not change the underlying nature of the game. In this appendix we explain this in more detail, but the details are not necessary for understanding the technical contributions of this paper.

First, we can take a game or evaluation function that is

naturally constant-sum and make it non-constant-sum by either adding a player which doesn't actually play in the game, but receives a random score, or by applying an affine transformation to one or more players' scores. Neither of these transformations, however, will change the *strategies* calculated by $\max^n$, because $\max^n$ makes decisions based on the relative ordering of outcomes, which an affine transform preserves, and because any extra players added will not actually make decisions in the game tree.

If we are unaware of either of these changes, previous pruning algorithms may treat the game as non-constant-sum and miss pruning opportunities. But, leaf-value tables are not affected by either of these changes. Leaf-value tables compute the ranking of all outcomes, so any affine transform applied to an evaluation function will be removed by this process. Because no bounds will ever be collected for any extra players in the game, as they are not actually a part of the game, pruning decisions are unchanged.

Secondly, we can take a game that is non-constant-sum and make it constant-sum by adding an extra player whose score is computed such that it makes the game constant-sum. Again, because this extra player never actually plays in the game, no pruning algorithm will ever collect bounds for this player, and it is equivalent to playing the original game. No extra pruning can ever be derived from such a change.

Adding an additional player may also make a non-linear classifier appear to be linear. But, because the extra player never plays, we will actually need to use the non-linear classifier to make pruning decisions.

Thus, while the difference between constant-sum games and non-constant-sum games can be blurred by simple transforms, the underlying game properties remain unchanged.

References

- [Carmel and Markovitch, 1996] Carmel, D., and Markovitch, S., Incorporating Opponent Models into Adversary Search, AAAI-96, Portland, OR.
- [Donkers, 2003] Donkers, H.H.L.M., Nosce Hostem: Searching with Opponent Models, PhD Thesis, 2003, University of Maastricht.
- [Korf, 1991] Korf, R., Multiplayer Alpha-Beta Pruning. Artificial Intelligence, vol. 48 no. 1, 1991, 99-111.
- [Luckhardt and Irani, 1986] Luckhardt, C.A., and Irani, K.B., An algorithmic solution of N-person games, Proceedings AAAI-86, Philadelphia, PA, 158-162.
- [Russell and Norvig, 1995] Russell, S., Norvig, P., Artificial Intelligence: A Modern Approach, Prentice-Hall Inc., 1995.
- [Sturtevant, 2004] Sturtevant, N.R., Current Challenges in Multi-Player Game Search, Proceedings, Computer and Games 2004, Bar-Ilan, Israel.
- [Sturtevant, 2003] Sturtevant, N.R., Last-Branch and Speculative Pruning Algorithms for $\max^n$, Proceedings IJCAI-03, Acapulco, Mexico.
- [Sturtevant and Korf, 2000] Sturtevant, N.R., and Korf, R.E., On Pruning Techniques for Multi-Player Games, Proceedings AAAI-2000, Austin, TX, 201-207.

Choosing between heuristics and strategies: an enhanced model for decision-making *

Shavit Talman, Rotem Toister, Sarit Kraus

Department of Computer Science, Bar-Ilan University

Ramat-Gan, 52900, Israel

{toistet, talmans, sarit}@cs.biu.ac.il

Abstract

Often an agent that has to solve a problem must choose which heuristic or strategy will help it the most in achieving its objectives. Sometimes the agent wishes to obtain additional units of information on the possible heuristics and strategies in order to choose between them, but it may be costly. As a result, the agent's goal is to acquire enough units of information in order to make a decision while incurring minimal cost. We focus on situations where the agent must decide in advance how many units it would like to obtain. We present an algorithm for choosing between two options, and then formulate three methods for the general case where there are $k > 2$ options to choose from. We investigate the 2-option algorithm and the general k -option methods effectiveness in two domains: the 3-SAT domain, and the CT computer game. In both domains we present the experimental performance of our models. Results will show that applying the 2-option algorithm is beneficial and provides the agent a substantial gain. In addition, applying the k -option method in the domains investigated results in a moderate gain.

1 Introduction

When making decisions, automated agents need to decide which heuristic function or strategy would best assist them in achieving their objectives. In particular, when an agent possesses several alternatives to tackle a problem, and the best alternative is unknown in advance, its success depends on choosing the most beneficial alternative. In [Azoulay-Schwartz and Kraus, 2002] a formal model for solving a problem of choosing between alternatives in e-commerce have been presented. They formulated the problem in terms of units of information and agent's utility, and assumed that the agent should decide in advance on how many information units to obtain about each alternative. Moreover, the agent could not change its decision during the information obtaining process. Then they suggested an algorithm for acquiring

an optimal amount of information units, through which the best alternative could be determined.

In this paper, we generalize their model to suit domains that involve choosing between heuristics or strategies. Azoulay-Schwartz and Kraus' assumptions hold in this research as well: the agent chooses the best alternative in advance and is able to hold onto its decision for a long period of time. This approach is valid in many situations, especially when acquiring information is costly. In these cases, the agent would like to execute its decision-making process once and apply the process's output thereafter. Moreover, once the agent realizes the decision made does not produce satisfactory results at any given time, it may re-execute the decision-making process, and proceed with its new output. Nevertheless, we are more concerned with the applicative rather than the theoretical aspect of the model. Indeed, the model suggests many implementation challenges. We present two examples, from the heuristics domain and from the computer games domain. In both domains we compare the utility of the automated agent with and without using our algorithm. Results will show that an agent that applies our model is more likely to choose the best option among the group of possible options. Furthermore, it does not waste too many resources during the decision process, and thus, its overall utility increases.

The paper first briefly presents the model constructed by Azoulay-Schwartz and Kraus. It then introduces the algorithm for choosing between two options, which was formulated by Azoulay-Schwartz and Kraus, and thereafter the three generalized methods for cases where there are more than two options. The first of these three methods is a theoretical one which was formulated by Azoulay-Schwartz and Kraus. We generalized their ideas to form the other two methods, which are much more feasible in real-time environments. We proceed by describing the experimental design and analysis. In section 4 we describe the first domain we considered - the 3-SAT domain, and in the following section we describe the other domain, the CT game domain. In both sections we present the experiments we conducted and the outcomes of applying the 2-option algorithm and the k -option generalized models. In section 6 we compare our work with previous researches. The concluding section discusses the experimental results and suggests several avenues for future research.

*This work was supported in part by NSF grant no. IIS-0208608 and by ISF grant no. 1211-04. Sarit Kraus is also affiliated with UMIACS.

2 Model construction

A risk neutral agent has to choose a heuristic from k heuristics to solve M problems. After choosing alternative i , the agent will obtain a value of x_i which is unknown in advance. We assume that x_i is normally distributed, that is, $x_i \sim N(\mu_i, \sigma_i^2)$. In addition, the agent does not know μ_i , but it has some prior beliefs about its distribution. We further assume that μ_i also follows the normal distribution - $\mu_i \sim N(\zeta_i, \tau_i^2)$. We assume that σ_i , ζ_i and τ_i are known and reflect the agent's beliefs. Otherwise, if σ_i is unknown, the agent can use The student distribution. In addition, if ζ_i and τ_i are unknown, we assume that the agent can estimate their values based on its beliefs as to the possible values of μ_i . Furthermore, in future work we will investigate the effect of eliminating normal distribution assumptions.

The agent has n_i units of information about each alternative i , with an average value of $\bar{x}_i$. For instance, in the e-commerce example presented by Azoulay-Schwartz and Kraus, a customer would like to buy an item available from two suppliers. The customer collected n_1 and n_2 customer-impressions from friends or from the web about these suppliers, in order to be able to decide between them. The average impression was calculated to form $\bar{x}_1$ and $\bar{x}_2$.

The agent is able to obtain a combination of $comb = (m_1, \dots, m_k)$ additional units about the different alternatives. However, this operation is costly, either in time or in direct costs. Given $0 < \delta \leq 1$, the discount time factor, the cost model, the list of alternatives and the parameters for each alternative, the agent decides whether to proceed with the information it has so far, or to accumulate additional information units. One could simply use a greedy algorithm in order to find the optimal allocation of additional experiments. In each step the option which proved to be better so far is chosen, executed and its result is added to the data accumulated. The greedy algorithm stops when there is no additional sample that increases the expected utility. However, [Azoulay-Schwartz and Kraus, 2002] showed that the greedy algorithm is not optimal. The reason being that there may be situations where obtaining one unit of information about a particular alternative is not worthwhile since it will not lead to a change in the decision, but obtaining two and more may be worthwhile. In conclusion, first we will present an algorithm for $k=2$ heuristics (or strategies) and then generalize it for $k > 2$.

Algorithm 1 The HSC Algorithm

```

for  $m_A$  from 0 to  $\infty$  do
  for  $m_B$  from 0 to  $\infty$  do
    calculate Fchange.
    calculate Utility.
    if  $Utility(m_A - 1, m_B - 1) > Utility(m_A, m_B)$  and
        $Utility(m_A - 1, m_B - 1) > Utility(m_A - 2, m_B - 2)$  then
      return  $(m_A - 1, m_B - 1)$ .
Add  $m_i$  experiments to alternative  $i$ .
Choose the best alternative according to the new results

```

2.1 Choosing Between Two Heuristics

Suppose the agent has to decide between alternatives A and B. Currently, $\bar{x}_A > \bar{x}_B$. Since the agent is risk neutral, alternative A will be chosen if no additional information is obtained. Firstly, the probability of changing the winning option

is $Fchange(\mu_A, \mu_B, \sigma_A, \sigma_B, \bar{x}_A, \bar{x}_B, n_A, n_B, m_A, m_B) = Pr(Z > Z_\alpha)$, where Z is a random variable, having the standard normal distribution and Z_α represents the first value where B outperforms A (see [Azoulay-Schwartz and Kraus, 2002] for more information). Then, the expected benefits from obtaining additional m_A and m_B is a function of $Fchange$ and is denoted by $benefits(m_A, m_B)$. The agent's utility is $utility(m_A, m_B) = \int_{\mu_A} \int_{\mu_B} benefits(m_A, m_B) \cdot \delta^{m_A+m_B} - cost(m_A, m_B) d\mu_B d\mu_A$. The specifics of the *benefits* function depend on the cost model. Later we will show the specifications for three different cost models. The agent's goal is to find the pair m_A and m_B , that yields the highest $utility(m_A, m_B)$ value. By considering all possible combinations of information units about A and B, the optimal combination is achieved. Algorithm 1 presents the Heuristics-Strategies-Choosing (HSC) Algorithm for the two heuristics (or strategies) case.

Proposition 1. *The HSC algorithm stops and returns m_A and m_B that maximize the utility function.*

Proof. The intuition for the correctness of the algorithm lies in the form of the utility function. Since it consists of the multiplication of similar normal distribution functions, its form is Gaussian as well, with only one maximum point. As soon as the algorithm finds that point it stops and returns m_A and m_B . $\square$

2.2 Choosing Between Multiple Heuristics

There are $k > 2$ alternatives, and the agent can obtain up to $M-1$ units of information about each of them. We suggest three models:

1. The *Statistical model* in which we considered the agent utility function given a combination of information units as suggested by [Azoulay-Schwartz and Kraus, 2002]. Nevertheless, in order to find a combination of additional units for each alternative, one must solve a quadruple integral. As a result, the model proved to be inapplicable for our purposes;
2. The *Binary Tree model* where the alternatives construct the leaves' level of the tree. We apply the HSC algorithm for each pair. The winning alternative goes up a level in the tree. We repeat the procedure until the best alternative reaches the root. This model deviates from our initial assumption that all experiments are conducted prior to decision-making. The binary tree model is an intermediate approach between the greedy algorithm and the HSC algorithm. However, since the comparisons are done in pairs, a large number of experiments may be executed for alternatives that would not have been considered at all when regarding all the alternatives together;
3. The *Fixed number of experiments model* - in which we distribute a fixed number of experiments denoted by (N) between the different alternatives using the information we have so far. For example, suppose we have five alternatives to choose from. Alternative 1 produces the best results and alternative 5 produces the worst ones. Thus, it is worthwhile to execute more experiments of alternative 1 than of alternative 5, as less time will be wasted

on fewer alternatives while accumulating information on all possible alternatives. Following preliminary experiments, we set N to be four times the number of alternatives. Thus, it is large enough to accommodate experiments of all alternatives and nonetheless, economical in the number of additional experiments. Consequently, for the example of five alternatives $N=20$, and we suggest that the best alternative be assigned one third of N . The rest will be assigned two thirds of the remaining experiments: $m_1 = \lceil 20 \cdot \frac{1}{3} \rceil = 7$; $m_2 = \lceil 13 \cdot \frac{2}{3} \rceil = 5$; $m_3 = \lceil 8 \cdot \frac{2}{3} \rceil = 3$; $m_4 = \lceil 3 \cdot \frac{2}{3} \rceil = 2$ and $m_5 = \lceil 1 \cdot \frac{2}{3} \rceil = 1$. This ad-hoc approach will be refined in future work.

3 Experimental design and analysis

Our investigation using the HSC algorithm was conducted in two domains. The first, a classic NP-complete problem, i.e. the 3-SAT problem. To demonstrate the HSC algorithm's vast usage possibilities, it was employed in two different cost models:

1. Minimal Time (MT) scenario - the agent had to solve M formulas as quickly as possible;
2. Maximal Formulas (MF) scenario - the agent had T units of time to solve as many formulas as possible.

We chose three best-known search algorithms as the agent's possible heuristics: Greedy-SAT (GSAT), Simulated Annealing (SA) and GSAT with Random Walk. The agent had to perform its task by using one of these search algorithms. The second domain was a computer game, the CT game (for game specifications see [Grosz *et al.*, 2004]). Here, the agent's task was to maximize its game score. To this end, our agent had seven different strategies to employ against its opponent agent in the game, and it had to choose the best strategy against the opponent. The experiments compared the agent's achievements without using the HSC algorithm and its achievements after executing this algorithm. We also compared these results with methods where a large set of additional experiments had been conducted. ([Selman *et al.*, 1993] for instance conducted an unlimited number of experiments in order to choose the best heuristic). These methods found the best heuristic more frequently than our algorithm. Nevertheless, though the decisions made were slightly improved, given our cost model, the approach applied by Selman yielded a huge loss in the number of experiments executed. For space reasons, these results are not detailed here.

We first executed preliminary runs to prepare an offline database. The database comprised vital information, namely, which option is the best, and moreover, the mean quality of each option. Our hypothesis was that the execution of the HSC algorithm will indeed benefit the agent. Moreover, by executing a small number of additional experiments the agent's utility will increase.

4 The 3-SAT domain

We assumed that each Truth-assignment flip takes one unit of time. Thus, the MT scenario in this domain is a minimum problem: the agent must solve M formulas within a minimal number of flips. Without loss of generality, when comparing

two possible heuristics, we assumed that the better heuristic after n_A experiments is heuristic A . As a result, the agent would have chosen heuristic A for the M formulas. In that case, it would have taken the agent the average number of flips multiplied by the number of formulas, $M \cdot \mu_A$, to solve M formulas. After the HSC algorithm is executed, the total number of flips will be assembled from,

1. The number of flips in the $m_A + m_B$ extra experiments yielded by the HSC algorithm;
2. The number of flips to solve the remaining $M - m_A - m_B$ formulas in case heuristic B prevails;
3. The number of flips to solve the remaining formulas in case heuristic A still leads.

δ in this case is 1, since the agent solves the required formulas in the additional experiments. Accordingly, the utility function in this domain is $utility(m_A, m_B) = \int_{\mu_A} \int_{\mu_B} benefits(m_A, m_B) \cdot \delta^{m_A+m_B} - cost(m_A, m_B) d\mu_B d\mu_A = \int_{\mu_A} \int_{\mu_B} M \cdot \mu_A - Fchange(M - m_A - m_B) \cdot \mu_B - (1 - Fchange)(M - m_A - m_B) \cdot \mu_A - (m_A \mu_A + m_B \mu_B) d\mu_B d\mu_A = \int_{\mu_A} \int_{\mu_B} (M - m_A - m_B) \cdot Fchange \cdot (\mu_A - \mu_B) - m_B \cdot (\mu_B - \mu_A) d\mu_B d\mu_A$. The agent searches for m_A and m_B that maximize this equation.

The MF scenario in the domain is a maximum problem: the agent must solve as many formulas as possible within T flips. Considering heuristic A is the better heuristic after n_A experiments, the agent would have solved T/μ_A 3-SAT formulas. However, if it would have used the HSC algorithm, the total number of 3-SAT formulas would have been the summation of:

1. The number of formulas solved during the extra experiments yielded by the HSC algorithm, $m_A + m_B$;
2. The number of formulas to be solved using heuristic B , if it prevails, $Fchange \cdot [T/\mu_B]$;
3. The number of formulas to be solved if heuristic A is not changed, $(1 - Fchange) \cdot [T/\mu_A]$.

Here, δ is 1 as well, and $cost(m_A, m_B)$ consists of the number of flips lost in each case, namely $cost(m_A, m_B) = Fchange \cdot [(m_A \mu_A + m_B \mu_B)/\mu_B] + (1 - Fchange) \cdot [(m_A \mu_A + m_B \mu_B)/\mu_A]$. Lastly, the HSC algorithm will yield m_A and m_B that will maximize $utility(m_A, m_B) = \int_{\mu_A} \int_{\mu_B} m_A + m_B + Fchange \cdot \frac{T}{\mu_B} + (1 - Fchange) \cdot \frac{T}{\mu_A} - Fchange \cdot \frac{m_A \mu_A + m_B \mu_B}{\mu_B} - (1 - Fchange) \cdot \frac{m_A \mu_A + m_B \mu_B}{\mu_A} d\mu_B d\mu_A = \int_{\mu_A} \int_{\mu_B} m_B - \frac{\mu_B m_B}{\mu_A} + Fchange \cdot [\frac{T - \mu_A m_A}{\mu_B} - \frac{T - \mu_B m_B}{\mu_A} + m_A - m_B] d\mu_B d\mu_A$.

4.1 Implementation issues

We constructed 305 different 3-SAT formulas, each consisting of 100 different variables and 430 clauses. The formulas were tested in advance for the existence of a valid truth assignment. The GSAT algorithm restarted with a new random truth assignment after 5,000 flips, and the total number of restarts was set at 18. The temperature of the SA algorithm was set at 2%, and the experiment was stopped after 100,000 unsuccessful flips. The Random Walk with GSAT was implemented using three different probabilities of Walk: 50%,

Heuristic	GSAT	Simulated Annealing	Random 50%	Random 60%	Random 80%
Flips #	26.2	18.3	5.1	4.5	16.0

Table 1: Offline results of number of flips (in thousands)

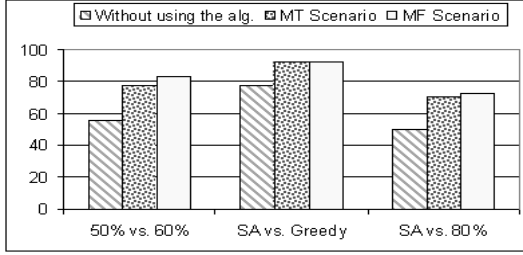


Figure 1: 3-SAT % of choosing the best heuristic without the HSC algorithm, with the algorithm in the MT and with the algorithm in the MF scenarios

60% and 80% (The three different heuristics of this algorithm will be denoted by Random 50%, 60% and 80%). In each experiment the maximal number of flips was set at 15,000 and the number of restarts was set at five. We established all the algorithms' parameters such that, on the one hand, they will have a reasonable potential to solve any formula, but on the other, their execution time will remain low.

The preliminary experiments executed each of the five heuristics on the 305 3-SAT formulas. Table 1 presents the average number of flips each heuristics obtained. The parameters in the equations of section 2 were estimated using the data accumulated during the preliminary experiments stage, since this data comprises our whole population. Thus, the a-priori parameters ζ_A and ζ_B were estimated with the mean of all the offline results of all heuristics (55.2), and τ_A and τ_B were estimated by their standard deviation (22.14). In addition, σ_A and σ_B were estimated by heuristic A and heuristic B standard deviations, respectively. A sensitivity analysis of these parameters determined that the results were not sensitive to changes in the parameters. In the analyses we varied the values of σ_A and σ_B , of ζ_A and τ_A and of ζ_B and τ_B and re-executed the experiments. Except for several extreme situations we obtained similar results for the specific values described above.

Furthermore, in each experiment n_A and n_B were set to five, M to 300, and T to 200,000. That is, in each experiment of both scenarios, five formulas chosen randomly from the 305 formulas mentioned above, were solved first, and were used as the agent's preliminary units of information. Then, it had to decide with which heuristic to proceed solving the remaining 300 formulas. We allowed the agent only five units of prior information since this is often the case in the real world - agents need to base their decision on only few observations due to uncertainty in their environment or due to a cost associated with the information.

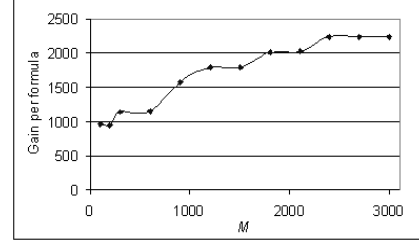


Figure 2: 3-SAT MT scenario influence of M

4.2 2-heuristic experimental results

According to the results presented in table 1 we compared (1) Random 60% to Random 50%; (2) SA to GSAT; (3) Random 80% to SA. Other pairs, although feasible, would not effectively demonstrate the HSC algorithm's performance: the difference between their means is too large for the algorithm to improve. From empirical results we know that the algorithm will not advise on further experiments in those cases, and simply yield the better heuristic according to the information the agent has. Thus, it will not endure any loss for these other pairs. The offline results revealed that the best heuristic in each pair was Random 60%, SA and Random 80%, respectively. The total number of experiments in each scenario was 120, 40 experiments per pair.

In the MT scenario, when the agent did not use the HSC algorithm, it always chose the heuristic with the better 5-game average. In contrast, when the agent applied the HSC algorithm, it mostly executed more experiments for both heuristics, and then either changed its mind or continued with the better 5-game heuristic. On average, only 12 additional experiments were executed for each heuristic. Figure 1 summarizes the percentage of experiments in which the agent chose the best heuristic, with and without the HSC algorithm across the three heuristic pairs. As we expected, the algorithm improved the agent's decision-making, and directed it to the best heuristic in 80% of the experiments. Moreover, without the HSC algorithm, the agent would have chosen the best heuristics only in 61% of the experiments. This improvement resulted in a gain in the number of flips: the average number of flips for 300 3-SAT formulas without the algorithm (4.20 million) was significantly higher than with the HSC algorithm (4.06 million)(Wilcoxon $p=0.07$).

To show the influence of M on the average gain, we executed 120 additional experiments, with 12 different values for M . Figure 2 summarizes the average gain per formula solved for M varying between 100 and 3000. As expected, the average gain is in linear relation to the size of M , since the HSC algorithm poses a greater potential benefit as the number of formulas increases.

The agent's task in the MF scenario was to solve as many formulas as possible within 200,000 flips. Here, on average nine additional experiments were executed for each heuristic. Figure 1 presents the percentage of experiments in which the agent chose the best heuristic, with and without using the HSC algorithm (note that the percent of without the HSC algorithm is the same as in the MT scenario). In this scenario

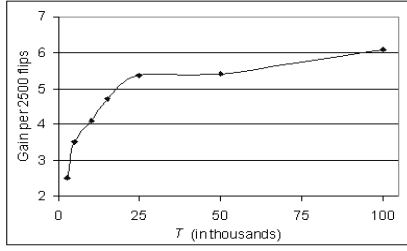


Figure 3: 3-SAT MF scenario influence of T

as well, the agent succeeded in choosing the best heuristic more frequently by using the algorithm (83% vs 61%). Consequently, it managed to solve six more formulas on average: 212 formulas with the HSC algorithm in contrast to 206 without it. To demonstrate the influence of T , we executed an additional 70 experiments for seven different values of T varying from 2,500 to 100,000. As the average gain is dependent on T (the larger the T the larger the gain), we normalized the gain by calculating the average gain per 2,500 flips. Thus, when T was set to 5,000 the average gain was divided by 2, and when it was set to 100,000 the average gain was divided by 40. Figure 3 summarizes the average gain per 2,500 flips. It supports our hypothesis that the gain of using the HSC algorithm increases as T increases. It seems that the increase is very steep at the beginning, but when $T > 25,000$ it becomes moderate.

4.3 k -heuristic experimental results

We tested the generalized k -heuristic model using the binary tree algorithm and the fixed number of experiments algorithm. The binary tree was built according to the pairs we described in section 4.2. That is, the binary tree's leaves were Random 50% and Random 60%, then GSAT and SA and finally Random 80%. The fixed number of experiments procedure was executed according to the description presented in 2.2. We repeated the procedures 40 times for both algorithms.

Table 2 summarizes the number of times (of the forty experiments) the agent chose each heuristic with and without each algorithm for the MT scenario. As expected, both algorithms improved the agent's decision-making, in the binary tree algorithm by 27.5% and in the fixed number of experiments algorithm by 25%. Nevertheless, the gain in the agent's utility was not very impressive in the binary tree case: on average 1,500 flips were lost per experiment due to the execution of heuristics that although seemed promising in their pair, were in fact time consuming in the overall aspect. On the other hand, in the fixed number of experiments case 1,522 flips were gained per experiment (the average total number of flips per experiment was 4,800). The number of additional experiments were 14 for each heuristic on average. Due to the disappointing results of the binary tree method, for the MF scenario we executed only the fixed number of experiments method. Here, on average 1.6 formulas more were solved per experiment when applying the algorithm.

	Greedy	SA	Random 80%	Random 50%	Random 60%
Without	1	2	2	16	19
With	0	0	1	9	30

	Greedy	SA	Random 80%	Random 50%	Random 60%
Without	1	2	2	16	19
With	0	0	0	11	29

Table 2: MT Scenario chosen heuristic distribution, with and without applying the HSC algorithm: (top) The binary tree results; (bottom) The fixed number of experiments results

5 The CT domain

We investigated the HSC algorithm in a two-player negotiation game that uses the CT game [Grosz *et al.*, 2004]. In this game each player has a goal placed on the game board and certain resources to help it reach the goal. The players can exchange resources, and at the end of the game are assigned a score that corresponds to their performance throughout the game. During the game the agents may negotiate on resources, commit to resource exchanges and execute exchanges. However, the commitments made by an agent are not enforceable, and it might decide to back down on a commitment or even deceive an opponent agent by committing to an exchange it does not intend to keep.

In [Talman *et al.*, 2005], an automated agent able to play repeated CT games was developed. The agent characterizes itself and its opponent in terms of cooperation and reliability. The cooperation trait measures the willingness of an agent to share its resources with others, whereas the reliability trait measures the agent's willingness to keep its commitments in the game. Accordingly, the agent is capable of employing seven strategies, differing in the level of cooperation and reliability the strategy dictates. Each strategy is suitable to a different type of an opponent, but the optimal matching scheme is unknown. For example, it may prove beneficial to play a low-reliability strategy against a highly-cooperative opponent by deceiving it. On the other hand, perhaps a more logical strategy against such an opponent will be a high-cooperation strategy which promotes reciprocity in the game, and may benefit the agent in the long run. Thus, in order to maximize the agent's score in the game, it must determine which strategy best-suits each opponent type. Each strategy trait can be low (L), medium (M) or high (H), and a strategy will be referred to by its cooperation-reliability level, such as a low-cooperation medium-reliability strategy (or LM strategy). Therefore, the seven possible strategies in the game are LL, LM, LH, MM, MH, HM and HH. The remaining two strategies, ML and HL are not applicable as an agent applying low reliability strategy has by definition a low cooperation level - when an agent almost never keeps its commitments, it is not willing to share its resources.

We expect that our suggested model will assist the agent in its decision-making, which will result in higher score in the game. Following n_i games of each strategy the agent executes the HSC algorithm and determines which strategy best-

Strategy	LL	LM	LH	MM	MH	HM	HH
Avg. Score	80.9	85.7	68.6	69.4	69.5	69	69.2

Table 3: Average score of each strategy against an LH opponent in the CT domain

suits each opponent type. The model adjusted to the CT game is similar to the 3-SAT model except for the introduction of the time discount factor (δ). Each game the agent plays with a non-optimal strategy results in a lower score. Hence, each experiment added by the HSC algorithm brings about a time cost of δ . In contrast, $cost(m_A, m_B) = 0$ in this domain: the execution of $m_A + m_B$ additional games does not incur an additional cost associated with m_A and m_B . In conclusion, for two strategies A and B (with A as the current winning strategy) having a mean score of μ_A and μ_B , respectively, the agent searches for m_A and m_B that maximize the utility function: $\int_{\mu_A} \int_{\mu_B} Fchange(\mu_B - \mu_A) \cdot \delta^{m_A+m_B} d\mu_B d\mu_A$. In addition, we define the agent’s gain from applying the HSC algorithm as follows: let $Score_A$ be the score of the strategy the agent chose before the algorithm and $Score_B$ be the score of the strategy it chose after using the algorithm. Then, its total gain from the algorithm is obtained by $(Score_B - Score_A) \cdot \delta^{m_A+m_B}$.

5.1 Implementation issues

For our study, we constructed 300 game scenarios, which varied the game boards, the resources each player was assigned and the dependency relationship between the two players. For evaluation purposes the HSC algorithm was executed to establish which strategy best suits an LH type opponent (best matches for the other opponent types are carried out in a similar manner). To this end, we first executed all 300 games, in which the agent played all seven strategies against an LH type opponent. The results served as the CT domain offline database as shown in table 3. Interestingly, the LM strategy produced the best results. In addition, we set δ at 0.90, to allow a certain amount of exploration. We later varied δ and investigated its influence on the algorithm’s performance.

5.2 2-heuristic experimental results

In light of the results in table 3, we compared strategies: (1) LL and LH; (2) LM and MH; (3) MM and HM. Other pairs can be compared in a similar manner as long as the difference between them is not too diverse for the HSC algorithm to introduce an improvement nor a loss. Each comparison was executed 40 times, totaling 120 experiments. In each experiment, our agent played five 2-player CT games against an LH opponent applying each strategy. Then, the agent executed the HSC algorithm, which yielded the additional number of games of each strategy the agent needs to play. On average only three games of each strategy were additionally executed.

Figure 4 summarizes the percentage of experiments in which the agent chose the best strategy, with and without using the HSC algorithm across the pairs of the three strategies. By using the HSC algorithm, the agent was able to apply the better strategy in 72% of the games, whereas without the HSC algorithm it would have applied the better strategy in only 60% of the games. Nevertheless, as we anticipated, when the

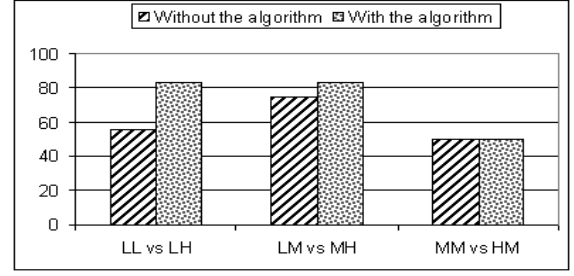


Figure 4: CT % of choosing the best strategy with and without the HSC algorithm

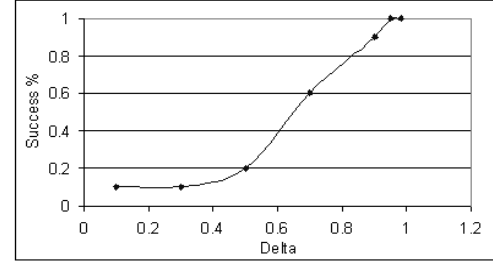


Figure 5: CT % of correct re-evaluations for different δ s

difference between the two alternatives was negligible as in the MM vs HM case, the algorithm did not improve the performance. This is because the two strategies are almost identical. Thus, applying the algorithm resulted in no gain since it rarely decided to perform additional experiments. However, no loss was incurred as well (the average number of additional experiments was 3 for each alternative). Accordingly, the agent’s gain was on average 0.8. This gain is somewhat low because it is multiplied by $\delta^{m_A+m_B}$.

Naturally, the time discount factor has a great influence on the HSC algorithm’s performance. We envisioned that if we increase δ the HSC algorithm will shift more frequently towards the better strategy. To test our hypothesis, we randomly selected ten experiments from the experiments described above, across the three pairs, in which the worse strategy would have been chosen by the agent according to the preliminary experiments. We then re-evaluated the HSC algorithm’s performance in those games with δ varying between 0.10 and 0.98. Figure 5 presents the percentage of experiments the HSC algorithm pointed out the better strategy, for each possible δ .

5.3 k-heuristic experimental results

We tested the generalized k -heuristic model using the binary tree algorithm and the fixed number of experiments algorithm. In the binary tree, the pairs were established according to the settings described in 5.2: LL vs LH, followed by MM vs HM, then LM vs MH, and finally we added the HH strategy. Then, we repeated the HSC algorithm bottom up. The fixed number of experiments procedure was executed according to the description in 2.2. We repeated the procedures 40 times for both algorithms.

The results in this domain were similar to those of the 3-SAT domain: both algorithms improved the agent's decision-making while keeping the number of additional experiments to a minimum (on average, 11 additional games per strategy) - the binary tree algorithm by 30% and the fixed number of experiments algorithm by 27.5%. The agent's gain was 0.015 in both cases. Again, this figure seems low, but any positive figure indicates a gain in performance due to the algorithm.

6 Related work

Our problem is different from the k-armed bandit problem [Tins, 1989] since once the agent decides which alternative to use it cannot change its decision. In the k-armed bandit problem, an item is chosen repeatedly and the decision of which item to select may change over time. Another approach is the statistical one. In [Pizarro *et al.*, 2000] the statistical ANOVA was employed for choosing the best model for a neural network. The procedure enabled them to isolate a subset of models whose mean error was the smallest, and subsequently the simplest model was chosen (Occams razor criteria). The main difference between that work and ours is that they were not interested in minimizing the number of additional experiments since no cost was involved. The same approach can be found in heuristics-related works. [Selman *et al.*, 1993], for instance, conducted a large number of experiments in order to identify the best heuristic. We suggest a method that will minimize the number of experiments to be executed, while providing the best option in a high percentage of the cases. In another work [Tseng and Gmytrasiewicz, 1999] developed an information-gathering system that uses the information value to guide the process. However, they assume that the number of possible answers for a query in the gathering process is finite while we consider a continuous set of possible answers. Moreover, they consider a myopic sequential procedure for the information gathering process. Thus, their solution is not optimal: they only consider the nearest step of information gathering, assuming that in each step the agent can decide about the next information to be obtained. [Grass and Zilberstein, 2000] developed a decision theoretic approach that uses an explicit representation of the user's decision model in order to plan and execute information-gathering actions. However, their system is based on information sources that return perfect information about the asked query.

In the world of strategies, researchers also have been more concerned with finding the most beneficial strategy in a given domain rather than formulating a general model. For example, in [Carlsson and Johansson, 1998] three types of strategies - generous, even-matched and greedy - were investigated as concepts for analyzing games. They granted the participating agents a strategy and examined their performances in two types of games. Their aim was to determine which kind of strategy was preferable and in which environments.

7 Conclusions and future work

In this paper, we generalized a model developed in [Azoulay-Schwartz and Kraus, 2002], which considered the problem of choosing between alternatives in e-commerce. We have presented an adjusted model which was designed to improve au-

tomated agents' decision-making. We evaluated this model in two different domains - an NP-complete problem and a computer-game framework. In each domain we executed a large number of experiments for several aspects of the domain. We have shown that agents that apply the HSC algorithm have improved their decision making by at least 20%, while executing a minimal number of additional experiments. Moreover, we have demonstrated how to adjust the HSC algorithm to suit these different domains and to perform well when solving both minimum and maximum problems.

Our future goal is to generalize the fixed number of experiments method. In this work we established its parameters empirically, and although it produced satisfactory results, we are interested in investigating its feasibility in additional domains. We also intend to examine the normal distribution assumption sensitivity. We assume that when the information units' distribution and the a-priori distribution are symmetric with one maximum, our assumptions hold. Nevertheless further analysis is required to ascertain this assumption.

References

- [Azoulay-Schwartz and Kraus, 2002] R. Azoulay-Schwartz and S. Kraus. Acquiring an optimal amount of information for choosing from alternatives. In *Proceedings of the 6th International Workshop on Cooperative Information Agents VI*, pages 123–137, 2002.
- [Carlsson and Johansson, 1998] B. Carlsson and S. Johansson. Generous and greedy strategies. In *Proceedings of Complex Systems*, 1998.
- [Grass and Zilberstein, 2000] J. Grass and S. Zilberstein. A value-driven system for autonomous information gathering. *J. of Intel. Information Systems*, 14:5–27, 2000.
- [Grosz *et al.*, 2004] B. Grosz, S. Kraus, S. Talman, B. Stosel, and M. Havlin. The influence of social dependencies on decision-making: Initial investigations with a new game. In *Proceedings of AAMAS*, pages 782–789, 2004.
- [Pizarro *et al.*, 2000] J. Pizarro, E. Guerrero, and P.L. Galindo. A statistical model selection strategy applied to neural networks. In *ESANN '2000*, pages 55–60, 2000.
- [Selman *et al.*, 1993] B. Selman, H. A. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *the 2nd DIMACS Challenge on Cliques, Coloring and Satis.*, 1993.
- [Talman *et al.*, 2005] S. Talman, K. Gal, M. Hadad, and S. Kraus. Adapting to agents' personalities in negotiations. In *Proceedings of AAMAS*, 2005.
- [Tins, 1989] J. C. Tins. *Multi-armed Bandit Allocation Indices*. John Wiley & Sons, 1989.
- [Tseng and Gmytrasiewicz, 1999] C. Tseng and P. J. Gmytrasiewicz. Time sensitive sequential myopic information gathering. In *HICSS-32*, 1999.

Decision Diagrams for the Computation of Semiring Valuations

Nic Wilson

Cork Constraint Computation Centre

Department of Computer Science, University College Cork, Ireland

n.wilson@4c.ucc.ie

Abstract

This paper describes a new approach to computation in a semiring-based system, which includes semiring-based CSPs (in particular weighted CSPs, fuzzy CSPs and standard CSPs) as well as Bayesian networks. The approach to computation is based on what we call semiring-labelled decision diagrams (SLDDs). These can be generated in a similar way to a standard search tree (decision tree) for solving a CSP, but some nodes are merged, creating a more compact representation; for certain classes of CSPs, the number of nodes in the resulting network will be a tiny fraction of the number of nodes in the corresponding search tree. A method is given for generating an SLDD that represents e.g., a particular instance of a semiring-based CSP; it is shown how this can be used to perform various computations of interest, such as solving a semiring-based CSP, finding optimal solutions, determining the possible values of each variable and counting solutions of a CSP.

1 Introduction

Suppose we are performing a depth-first search for solutions of a (not necessarily binary) CSP, where we record the nodes of the search (decision) tree that we've visited so far. Solutions correspond to complete paths in the tree, i.e., directed paths from the root node to a node corresponding to a complete assignment. We are at a node with associated partial assignment u of set of variables U , which we have checked is consistent (in the CSP sense of it satisfying all constraints involving only variables in U). Suppose we manage somehow to find another assignment u' of U , corresponding to a node previously visited, which is also consistent, and which is *fully interchangeable*, that is (extending [Freuder, 1991]) *for all assignments w of the remaining variables W , uw is a solution of the CSP if and only if $u'w$ is a solution.*

There is then no need to expand the subtree below u , as it is equivalent to the one below u' which we have already expanded. Instead we can merge the nodes corresponding to u and u' . The resulting structure will no longer be a tree—so can be described as a *decision diagram* (as opposed to a *decision tree*)—but it will still represent all solutions, as solu-

tions still correspond to complete directed paths; one can also prune the representation to allow backtrack-free generation of any solution. If we can find many such equivalences through interchangeability then the number of nodes in the decision diagram will be a tiny fraction of the number of nodes in the corresponding decision tree. Indeed, the number of solutions can even sometimes be exponential in the size of such a decision diagram, as the merging of nodes ‘factorises’ the solution set (so the size of the decision tree will be exponential in the size of the decision diagram). This kind of representation, in the form of binary decision diagrams [Bryant, 1986; 1995], has proved very effective in certain domains, such as CAD applications.

Testing full interchangeability is often expensive, but one can instead use the sufficient condition, *neighbourhood interchangeability*, that is, for all the constraints c whose scope intersects with both U and $V - U$, $c_u = c_{u'}$, where e.g., c_u (which involves no variables in U) is the ‘slice’ of c given by instantiating variables U to u . This condition can be tested quickly, in time proportional to the size of the (sliced) constraints. Furthermore, if one constructs these slices for each node visited, and indexes them appropriately, then checking whether we have previously generated neighbourhood-interchangeable u' , can be achieved efficiently.

This computational approach can be generalised to semiring-based systems (Section 2), including semiring-based CSPs (in particular weighted CSPs, fuzzy CSPs and standard CSPs) as well as Bayesian networks. Each edge in the decision diagram is labelled with a value in the semiring, generating (Section 3) what we call *semiring-labelled decision diagrams (SLDDs)*. We show how this can be done in such a way that the SLDD represents the semiring-based system, i.e., the product of semiring values of edges in a complete path is equal to the semiring value of the complete assignment associated with the path. The efficiency of this approach to computation is very much dependent on how compact the SLDD is, especially compared to the corresponding search tree. If the SLDD is compact, then it can be efficiently constructed and the operations are efficient. We give examples of situations where the SLDD is compact (even linear size) despite the number of solutions being exponential in the number of variables.

Sections 4 and 5 show how the semiring-labelled decision diagram can be used to perform various computations of in-

terest, including solving, optimising and generating inferred constraints, and projections needed in inference in Bayesian networks and solving semiring-based CSPs.

The formalism SLDDs defined in this paper is a development of the finite-state automata representations for weighted constraints in [Amilhastre *et al.*, 2002] (where the soft constraints are unary). The latter, which can be considered as implementing a kind of *dynamic programming* [Bellman, 1957], has been shown to be applicable as a compilation technique on a substantial real-world problem.

The main contributions of the paper are extending this computational technique of [Amilhastre *et al.*, 2002] in a number of significant ways: our technique, SLDDs, can be applied to computation in many formalisms for preferences and uncertainty: in particular, to a very general class of soft CSPs, semiring-based CSPs; we allow soft constraints to be of arbitrary *arity*, as well as both extensional and intensional representations of constraints; it applies also to computation in Bayesian networks, and for counting the solutions of a CSP. Our construction of the representation is also new, and is based on backtracking search, where certain nodes in the search tree are merged (implicitly performing a kind of formalised caching); this enables constraint programming techniques to be used, making SLDDs a combination of a form of dynamic programming and search-based approaches used in constraint programming.

2 Semiring Valuations

Semirings consist of a set with two operations $\otimes$ and $\oplus$, which are both associative and commutative and such that $\otimes$ distributes over $\oplus$. Here we also assume a unit element and a null element. Tuple $\mathcal{A} = \langle A, \mathbf{0}, \mathbf{1}, \oplus, \otimes \rangle$ is said to be a *semiring* if A is a set containing different elements $\mathbf{0}$ and $\mathbf{1}$ and $\oplus$ and $\otimes$ are operations on A satisfying the following properties: $\otimes$ is associative and commutative with identity $\mathbf{1}$, $\oplus$ is associative and commutative with identity $\mathbf{0}$, which is also a null element (i.e., for all $\alpha \in A$, $\alpha \otimes \mathbf{0} = \mathbf{0}$), and $\otimes$ distributes over $\oplus$ i.e., and for all $\alpha, \beta, \gamma \in A$, $\alpha \otimes (\beta \oplus \gamma) = (\alpha \otimes \beta) \oplus (\alpha \otimes \gamma)$.

Semiring valuations. Let V be a finite set of variables. For each $X \in V$ let $D(X)$ be the domain (i.e., the set of possible assignments) of X . For $U \subseteq V$ let the set of partial tuples $D(U) = \prod_{X \in U} D(X)$ be the set of possible assignments to set of variables U . A complete assignment is an element of $D(V)$. For $u \in D(U)$ and $W \subseteq U$, $u^{\downarrow W}$ is the projection of u onto variables W , so that for all $X \in W$, $u^{\downarrow W}(X) = u(X)$.

Let $\mathcal{A} = \langle A, \mathbf{0}, \mathbf{1}, \oplus, \otimes \rangle$ be a semiring. An $\mathcal{A}$ -valuation c associates a semiring value with each of a particular set of partial tuples: c is a function from $D(V_c)$ to A , where set of variables $V_c \subseteq V$ is called the *scope* of c . If $u \in D(U)$ is an assignment to set of variables U containing V_c , we may write $c(u)$ as an abbreviation for $c(u^{\downarrow V_c})$. The semiring operations allow one to define a combination and a projection operation on semiring valuations. The combination $c \otimes c'$ of $\mathcal{A}$ -valuations c and c' is defined to have scope $V_c \cup V_{c'}$, and by, $(c \otimes c')(u) = c(u) \otimes c'(u)$, i.e., $c(u^{\downarrow V_c}) \otimes c'(u^{\downarrow V_{c'}})$, for each assignment u of $V_c \cup V_{c'}$. $\otimes$ is a commutative and associative operation on $\mathcal{A}$ -valuations (on V). If C is a multiset of

$\mathcal{A}$ -valuations, we may write $\bigotimes C$ to mean $\bigotimes_{c \in C} c$. Let c be an $\mathcal{A}$ -valuation, and let U be a subset of V_c . The projection $c^{\downarrow U}$ of c onto U is defined by, for $u \in D(U)$, $c^{\downarrow U}(u)$ is equal to the semiring summation of $c(w)$ over all $w \in D(V_c)$ such that $w^{\downarrow U} = u$.

Many computational problems can be expressed as computing a projection of a combination $C = \bigotimes C$ of a multiset C of semiring valuations (and a straightforward approach to such a computation is exponential). In particular, computations in semiring-based CSPs [Bistarelli *et al.*, 1997; 1999; 2002] are of this form. This is a general formalism for soft constraints, which includes weighted CSPs, fuzzy CSPs, probabilistic CSPs, lexicographic CSPs and set-based CSPs. These use a special type of semiring, where $\oplus$ is idempotent and $\mathbf{1}$ is an absorbing element of $\oplus$.

Finite CSPs can be expressed as a set C of $\mathcal{A}$ -valuations, using $\mathcal{A} = \langle \{0, 1\}, 0, 1, \max, \times \rangle$, so that $\otimes = \times$ (i.e., $\min$) and $\oplus$ is $\max$. A constraint on set of variables U is represented as a semiring valuation $c : D(U) \rightarrow \{0, 1\}$, where for assignment u of U , $c(u) = 1$ if and only if u satisfies the constraint. The CSP has a solution if and only if $(\bigotimes C)^{\downarrow \emptyset} = 1$.

To perform computations in Bayesian networks, we use the semiring $\mathcal{A} = \langle \mathbb{R}^+, 0, 1, +, \times \rangle$, where $\mathbb{R}^+$ is the set of non-negative real numbers. As is well known, computation in Bayesian networks [Pearl, 1988; Shenoy and Shafer, 1990] can be performed by computing projections of combinations of such $\mathcal{A}$ -valuations, where each $\mathcal{A}$ -valuation represents a conditional probability table of a variable given its parents in the Bayesian network. In particular, the marginal probability that variable X is assigned to x is given by $C^{\downarrow \{X\}}(x)$. This semiring can also be used to count the solutions in a CSP as $C^{\downarrow \emptyset}$, by restricting the input semiring values to being 0 or 1.

3 Constructing an SLDD Representing a Combined Semiring Valuation

We will show how to construct a structure,¹ called a *semiring-labelled decision diagram (SLDD)*, that represents, in a particular sense, the combination $C = \bigotimes_{c \in C} c$ of a multiset C of $\mathcal{A}$ -valuations, where $\mathcal{A} = \langle A, \mathbf{0}, \mathbf{1}, \oplus, \otimes \rangle$ is a given semiring. The SLDD is a directed acyclic graph, where the edges are labelled with elements in A . As we'll show in Sections 4 and 5, this representation can then be used to perform certain important computations.

The construction of the SLDD is very similar to that of a search tree generated by chronological backtracking; the difference is that when we generate an edge we use a certain condition *forward neighbourhood interchangeability* (defined below, extending the definition of neighbourhood interchangeability in the introduction) to test if we can avoid creating a new node, but instead connect the edge to an already existing node.

Let $U \subseteq V$ be a set of variables. Define a valuation $c \in C$ to be *active* (with respect to U) if its scope intersects both U and $V - U$. For active valuation c , define, for $u \in D(U)$, $\mathcal{A}$ -valuation c_u to be the 'slice' of c given by assigning U to

¹A more general definition of SLDDs is given in [Wilson, 2004], with a corresponding version of Theorem 1.

u ; that is, the scope V_{c_u} of c_u is $V_c - U$, and for all assignments of t to V_{c_u} , $c_u(t)$ is defined to be $c(ut)$ (where ut is the concatenation of tuples u and t). Assignments u and u' to set of variables U are said to be *forward neighbourhood interchangeable* (with respect to C) if for all active $\mathcal{A}$ -valuations c , the slices c_u and $c_{u'}$ are equal. Roughly, what this implies is that the subproblems corresponding to u and u' are the same, so there is no need to solve them more than once.

A semiring-labelled decision diagram (over set of variables V) is defined to be an $\mathcal{A}$ -decision diagram (over V) for some semiring $\mathcal{A} = \langle A, \mathbf{0}, \mathbf{1}, \oplus, \otimes \rangle$. An $\mathcal{A}$ -decision diagram consists of a directed acyclic graph with a unique earliest (parent-less) node **source** and a unique latest (childless) node **sink**. Each edge λ is labelled with a value $\alpha(\lambda)$ in the semiring (defined below). A complete path is defined to be a maximal (directed) path, that is, a path from **source** to **sink**.

Nodes and edges are both labelled with various pieces of information, as described below. Associated with each node r (other than **sink**) is a variable X_r , which is the variable that is about to be instantiated. Also U_r is the set of variables associated with nodes on any (directed) path from **source** to r .

We start off by creating the node **source**, and define $U_{\text{source}} = \emptyset$, and u_{source} to be the trivial assignment to the empty set of variables. We also choose some variable $Y \in V$, and set $X_{\text{source}} = Y$. The construction process works by choosing a node already constructed, and constructing the (directed) edges coming from the node, and the nodes at the end of the edges. At each point we choose a node $r \neq \text{sink}$ which has currently no children (i.e., no edges emanating from it). If there is no such node, the construction is complete.

For each assignment x to X_r we create a (directed) edge λ from r with associated assignment $X_r = x$, (and define $X_\lambda = X_r$) and set of variables $U_\lambda = U_r \cup \{X_r\}$, and assignment u_λ to U_λ which is u_r extended with assigning x to X_r . The semiring value $\alpha(\lambda)$ is defined to be $\bigotimes_{c \in C} c(u_\lambda)$ where the semiring product is over all $c \in C$ which are ‘just instantiated’, i.e., such that $X_\lambda \in V_c \subseteq U_\lambda$.

If $U_\lambda = V$, so that all the variables have been instantiated, we connect the end of λ to **sink**. If $\alpha(\lambda) = \mathbf{0}$ we also set the end of λ to be **sink**. If $\alpha(\lambda) \neq \mathbf{0}$, and we can find previously created node r' such that u_λ and $u_{r'}$ are forward neighbourhood interchangeable (this requires also $U_\lambda = U_{r'}$), we set the end of λ to be r' . We call this ‘merging’, since it corresponds to merging nodes in a decision tree. Otherwise we create a new node r' , set $u_{r'} = u_\lambda$, $U_{r'} = U_\lambda$ and choose some $X_{r'} \in V - U_{r'}$. To aid future ‘merging’, we also store with the node r' the slices $c_{u_{r'}}$, for active valuations c .

Following a (directed) path thus corresponds to an assignment to a set of variables; also a complete assignment $v \in D(V)$ determines a *complete path* (i.e., a (directed) path from **source** to **sink**) π_v , by following, from each node r , the edge associated with value $v(X_r)$.

The semiring value $\alpha(\pi)$ associated with a path is the semiring product of the semiring values associated with each edge in the path, i.e., $\bigotimes_{\lambda \in \pi} \alpha(\lambda)$. This might be thought of as the cost of following path π . For $\mathcal{A}$ -decision diagram S

define the $\mathcal{A}$ -valuation c_S by, for $v \in D(V)$, $c_S(v) = \alpha(\pi_v)$. The SLDD is said to *represent* $\mathcal{A}$ -valuation c_S .

Theorem 1 *Let $\mathcal{A}$ be a semiring and let C be a multiset C of $\mathcal{A}$ -valuations over variables V . Let S be an $\mathcal{A}$ -decision diagram S constructed as above. Then S represents $\mathbf{C} = \bigotimes C$ and $\mathbf{C}^{\downarrow \emptyset} = \sum_{\pi} \alpha(\pi)$, where the sum is over all complete paths in S .*

In the next section it is shown how to compute such sums efficiently.

Note that ‘dynamic variable ordering’ can be used, i.e., the order of variables can differ between paths. We can also adapt constraint programming techniques to improve the efficiency. We can use propagation e.g., maintaining arc consistency for the zeros [Wilson, 2004]. Furthermore, the SLDD can, like a search tree, be generated in a depth-first fashion: this is natural when we’re attempting to solve a CSP; the construction can be terminated when a solution is found.

To facilitate merging, for partial tuple u we can represent the collection of slices $\{c_u : c \text{ active}\}$ as a string $\sigma(u)$ of elements in the semiring. Then, for $u, u' \in D(U)$, tuples u and u' are forward neighbourhood interchangeable if and only if $\sigma(u) = \sigma(u')$. For set of variables U , we can incrementally construct a *trie* (a tree of strings) storing each $\sigma(u_r)$ for all nodes r with $u_r \in D(U)$; this enables efficient generation of each edge of the SLDD. For valuations represented extensionally, define $\text{size}(C)$ to be $\sum_{c \in C} \text{size}(c)$, where $\text{size}(c)$ is the cardinality of the scope of c multiplied by the number of non-zero tuples in c . An upper bound for the time needed to construct the SLDD is then proportional to $\text{size}(C) \times \text{size}(S)$, where $\text{size}(S)$ is the number of edges of the SLDD. In particular, if the size of the SLDD is polynomial in n , for some parametrised family of problems depending on the number of variables n , then constructing the SLDD is polynomial.

When is the SLDD compact?

The crucial factor in the determining the efficiency of our approach is the size of the SLDD; if the SLDD is compact, then it can be efficiently constructed and the operations are efficient. The SLDD will tend to be compact (compared to the associated search tree) in situations where many different partial instantiations lead to equivalent subproblems (and so a standard search-based technique would solve the same subproblem many times). There are many kinds of situations where the problem structure causes this to happen.²

We give a class of examples to illustrate that the constructed semiring-labelled decision diagram can be compact, even if it represents an exponential number of non-zero complete assignments (e.g., solutions of a CSP). In these cases the size of the SLDD is linear in the number of variables, whereas the size of the corresponding search tree is exponential.

Example. Let $V = \{X_1, \dots, X_n\}$, where the size of each domain is at most d . Let C be a multiset of $\mathcal{A}$ -valuations, for some semiring $\mathcal{A}$, such that the scope of each valuation

²Even for random (binary) CSPs, it has been demonstrated experimentally that the size of the SLDD representation can be several orders of magnitude smaller than the size of the corresponding search tree (this happens when the problems are sparse and loose).

involves variables which are at most p apart (where p is constant and fairly small): for all $c \in C$, $\max \{i : X_i \in V_c\} - \min \{i : X_i \in V_c\} \leq p$. This might arise, for example, in a problem where there is a temporal component, with the index i related to time. We can generate an $\mathcal{A}$ -decision diagram using the fixed variable ordering $X_1, \dots, X_n$ (i.e., the variables appear in that order in all paths). Let j be such that $p \leq j < n$, and consider any assignments u and u' to set of variables $U = \{X_1, \dots, X_j\}$. If u and u' agree on their last p variables, i.e., $u(X_i) = u'(X_i)$ for all i such that $j - p < i \leq j$, then they are forward neighbourhood interchangeable, since for any active $\mathcal{A}$ -valuation c , u and u' agree on the scope of c , so $c_u = c_{u'}$. This implies that the construction of the SLDD will produce at most d^p nodes at ‘level’ j , and an upper bound for the size of the SLDD is hence nd^{p+1} ; this is linear in n , since d and p are constants. For the CSP case, the number of solutions will often be exponential in n , and hence exponential in the size of the SLDD. Many operations, such as counting the number of solutions, or finding an optimal solution for more general semirings (see Section 5) can be achieved in time linear in n , even though an exponential number of assignments are being reasoned about.

A similar argument leads to an upper bound on the size of the SLDD of the same form, but in terms of *pathwidth* [Bodlaender, 1993], for *arbitrary multisets of semiring valuations*. (A fixed variable ordering used to generate an SLDD gives rise to a corresponding path decomposition.)

Other examples. Certain problems involving permutations can also lead to relatively compact SLDDs; for example, counting the number of linear orders which extend a given partial order [van Dongen, 2004]. The structure of the problem ensures that the SLDD decomposes the problem (similarly to a dynamic programming approach), reducing a factorial problem to an exponential one—which can be a huge reduction in time complexity, but at the cost of large space requirements. Similar remarks apply to solving a *rehearsal problem* (CSPLib <http://www.csplib.org/> problem number 39). Compact structures with a form similar to an SLDD can also be used for enforcing generalised arc consistency for many forms of global constraints e.g., [Pesant, 2004].

4 Propagation of Semiring Values in SLDD

The purpose of this section is to show how to efficiently perform various technical computations in an SLDD, in particular, the sum of the semiring values over (i) all complete paths, (ii) all complete paths which pass through a particular node, (iii) all complete paths which pass through a particular edge. The algorithms are immediate generalisations of a classic shortest path algorithm (and the application to the weighted CSP semiring $\langle \mathcal{N} \cup \{0, \infty\}, \infty, 0, \min, + \rangle$ reduces to this shortest path algorithm).

Suppose we have an $\mathcal{A}$ -decision diagram, with semiring $\mathcal{A} = \langle A, 0, 1, \oplus, \otimes \rangle$. We associate with each node r two semiring values $f(r)$ and $g(r)$, which are defined inductively. Define $f(\text{source}) = 1$, and, working forwards from

source, for each $r \neq \text{source}$, define

$$f(r) = \bigoplus_{r'' \rightarrow r} (f(r'') \otimes \alpha(r'' \rightarrow r)).$$

where the semiring summation is over all edges r'' which point to r . Because the SLDD is a directed acyclic graph with **source** being the unique parent-less node, this defines $f(r)$ for each r unambiguously. Symmetrically, define $g(\text{sink}) = 1$, and, working backwards from **sink**, define, for $r \neq \text{sink}$, $g(r) = \bigoplus_{r \rightarrow r'} (\alpha(r \rightarrow r') \otimes g(r'))$, where the semiring summation is over all edges $r \rightarrow r'$ emanating from r . Furthermore, for each node r and edge $r \rightarrow r'$, define $h(r) = f(r) \otimes g(r)$, and $h(r \rightarrow r') = f(r) \otimes \alpha(r \rightarrow r') \otimes g(r')$.

Proposition 1 *Let r be a node and let λ be an edge in the SLDD. If $r \neq \text{source}$ then $f(r) = \bigoplus_{\pi} \alpha(\pi)$ where the semiring summation is over all paths π from **source** to r . In particular, $f(\text{sink})$ is the semiring sum of $\alpha(\pi)$ over all paths from **source** to **sink**. Similarly, if $r \neq \text{sink}$, $g(r) = \bigoplus_{\pi} \alpha(\pi)$, where the semiring summation is over all paths from r to **sink**. Also, if $r \notin \{\text{source}, \text{sink}\}$, $h(r) = \bigoplus_{\pi} \alpha(\pi)$ where the semiring sum of $\alpha(\pi)$ is over all complete paths passing through node r . Furthermore, $h(\lambda)$ is equal to the semiring sum of $\alpha(\pi)$ over all complete paths which include the edge λ .*

This proposition generalises proposition 8 of [Amilhastre et al., 2002], and the definitions of f and g are generalised forms of functions in definition 10 of [Amilhastre et al., 2002].

Note that the number of semiring operations needed to compute functions f , g and h is *linear in the size of the SLDD*.

5 Computations Using an SLDD

In this section we show how the SLDD representation can be used to perform important computations: projections and finding optimal solutions. In [Wilson, 2004] we also showed how the SLDD can be used for randomly picking a solution of a CSP with uniform probability (by using an algorithm with a very similar structure to that described below for finding an optimal solution). It is assumed that we have an SLDD representation of a function $\mathbf{C} : D(V) \rightarrow A$, based on a semiring $\mathcal{A} = \langle A, 0, 1, \oplus, \otimes \rangle$; we also assume that we have computed the associated functions f , g and h , as described in the last section.

5.1 Performing Projections

As mentioned above, computing the projection of the combination of a multiset of semiring valuations is important for many different applications. It gives what is called in [Bistarelli et al., 1997] the ‘solution’ of a semiring-based constraint problem, and can be used for generating implied soft constraints in an idempotent semiring-based CSP [Bistarelli et al., 2002]. Projecting to the empty set of variables can be used, for example, for computing the weight of the best solution in a weighted CSP, or for counting solutions of a CSP. Projection to a singleton set determines the possible values

of a variable for CSPs, and analogously for other semiring-based CSPs; for the Bayesian network semiring it computes a marginal of the Bayesian network.

The lemma below shows how the projection of $\mathbf{C}$ onto the empty set and to singleton sets can be computed efficiently. It follows easily from proposition 1.

Lemma 1 Suppose an $\mathcal{A}$ -decision diagram on set of variables V (with associated functions f, g and h) represents $\mathcal{A}$ -valuation $\mathbf{C}$. Let $X \in V$ be a variable, and $x \in D(X)$ be a value of X . Then (i) $\mathbf{C}^{\downarrow\emptyset} = f(\text{sink}) = g(\text{source})$; and (ii) $\mathbf{C}^{\downarrow\{X\}}(x) = \bigoplus_{\lambda \in \Lambda_x} h(\lambda)$, where Λ_x is the set of edges associated with assignment $X = x$.

So to compute $\mathbf{C}^{\downarrow\{X\}}(x)$ requires $|\Lambda_x| - 1$ semiring summations. To compute the projections onto every single variable, i.e., computing $\mathbf{C}^{\downarrow\{X\}}(x)$ for all variables X and all its values x , requires a semiring summation for each edge in the SLDD, so can be computed in time linear in the size of the SLDD.

Adding unary semiring valuations. Suppose we receive a new set of semiring valuations, all of which are unary (i.e., the scope of each is a single variable) which we want to combine with the previous combination $\mathbf{C}$. It can be seen that the same graphical structure can be used, but where we just change the semiring values on the edges. This can be used for computing more general projections; it also has applications in dynamic soft CSPs [Amilhastre *et al.*, 2002], and for conditioning, and hence inference, in a Bayesian network, and computing most probable explanations [Pearl, 1988].

5.2 Finding optimal solutions

In this subsection we only consider semirings $\mathcal{A} = \langle A, 0, 1, \oplus, \otimes \rangle$ with a special property: that for all $\alpha, \beta \in A$, either $\alpha \oplus \beta = \alpha$ or $\alpha \oplus \beta = \beta$. Semirings $\mathcal{A}$ satisfying this property are said to satisfy the *addition-is-max* property. We can then define a total order $\geq$ on A given by $\alpha \geq \beta$ if and only if $\alpha \oplus \beta = \alpha$. Any valuation structure, as defined in definition 11 of [Bistarelli *et al.*, 1999], gives rise to a semiring satisfying addition-is-max, by using the order relation to define $\oplus$ (note that we are writing the order the opposite way round). Hence the approach in this section applies to any valued CSPs [Bistarelli *et al.*, 1999], including weighted CSPs, fuzzy CSPs and lexicographic CSPs, as valued CSPs can be represented as multisets of $\mathcal{A}$ -valuations where $\mathcal{A}$ satisfies addition-is-max.

Complete assignment $v \in D(V)$ is defined to be optimal if $\mathbf{C}(v) \geq \mathbf{C}(v')$ for all $v' \in D(V)$. This happens if and only if $\mathbf{C}(v) = \mathbf{C}^{\downarrow\emptyset}$. To generate any optimal complete assignment $v \in D(V)$ is very fast: linear in n , the number of variables: we start with node `source` as the current node, and iteratively pick a child of the current node, satisfying the following condition, until we reach `sink`: we pick any child r' of current node r with maximum value of $\alpha(\pi) \otimes \alpha(r \rightarrow r') \otimes g(r')$ (among children of r), where π is the path chosen so far, from `source` to r . The associated complete assignment will be optimal, and conversely, any optimal complete assignment can be generated in this way.

If we are only interested in optimal solutions, we could eliminate each edge λ with $h(\lambda) < \mathbf{C}^{\downarrow\emptyset}$ (by connecting such

λ to `sink` and resetting $\alpha(\lambda)$ to 0), since such an edge is not part of any optimal complete path. For certain addition-is-max semirings (in particular, a fuzzy semiring with $\otimes$ equalling $\min$, or alternatively, if $\otimes$ is strictly monotonic), the resulting network has the property that every complete path is optimal (ignoring edges λ with $\alpha(\lambda) = 0$). This then gives considerable flexibility in generating optimal solutions in an interactive setting, such as guiding a user in solving a configuration problem.

6 Discussion

As well as the representation of [Amilhastre *et al.*, 2002], which they directly generalise (see Section 1), SLDDs are related to several other computational approaches, including search-based approaches for solving CSPs, and join tree/hypertree-based decomposition methods, and AND/OR search graphs. We very briefly discuss these relationships below.

Our construction of the SLDD in Section 3 is different from the way that the representation in [Amilhastre *et al.*, 2002] is generated or similar representations such as Binary Decision Diagrams [Bryant, 1986] and that of [Vempaty, 1992]. The latter involve building up the representation by adding the constraints incrementally, giving representations of subsets of the constraints. SLDDs could also be generated in a similar way; however, the construction in Section 3 takes all the constraints (semiring valuations) into account at once. This can sometimes be advantageous; in particular, when searching for a single solution of a CSP, the SLDD need have time complexity at most polynomially worse than a standard CSP search approach (such as chronological backtracking whilst maintaining arc consistency), whereas the experimental results of [Pan and Vardi, 2004] seem to suggest that a BDD approach can be exponentially worse than a search approach.

SLDDs often have large storage requirements, but they can involve very much fewer nodes (and need never involve more nodes) than a search tree approach. As illustrated by the example in Section 3 (and by the application in [Amilhastre *et al.*, 2002]), certain problems with a very large number of solutions (i.e., non-zero complete assignments) can have an SLDD of manageable size, leading to efficient computation; a search-based method will often then not be feasible if one wants to compile the set of solutions, or count the number of solutions.

Perhaps the most-studied general approach for this kind of problem is the join tree (or hypertree decomposition) approach, e.g., the general framework of Shenoy and Shafer [Shenoy and Shafer, 1990] Bucket Elimination [Dechter, 1999], and non-serial dynamic programming [Bertele and Brioschi, 1972]. Whether a join tree approach or a decision diagram approach is more efficient depends very much on the form of the problem; a join tree approach may be more efficient if there is appropriate topological structure, but little structure which is more value-specific; a decision diagram approach looks liable to be more efficient if there is a good deal of value-specific structure (e.g., of the form of ‘context-specific independence’ [Boutilier *et al.*, 1996], defined in a Bayesian network context), leading to a compact decision di-

agram. See also discussion in [Wilson, 2004].

An SLDD is similar in form to an OR search graph, a special case of AND/OR search graphs in [Dechter and Mateescu, 2004]; the latter are also closely related to the d-DNNF formalism—see e.g., [Darwiche, 2002], and Case-Factor Diagrams [McAllester *et al.*, 2004]. SLDDs can be extended to also include AND nodes; these can be used when the sliced constraints associated with a node can be partitioned into (e.g., two) sets with non-overlapping scopes; there is no longer a unique final node `sink`, and the computation generalises that of function g , starting from the sink nodes and working backwards, with semiring product $\otimes$ being used to combine the semiring values of the branches of an AND node. This sometimes leads to a still more compact representation, but with reduced functionality (as not all the techniques in Section 5 extend).

Summary

This paper introduces and develops semiring-labelled decision diagrams (SLDDs); this computational tool, combining a form of dynamic programming with a constraint programming approach, can be used to solve a variety of important computational problems for (soft or ordinary) constraints and uncertainty. For problem instances with appropriate structure, the SLDD will be compact, and hence be an efficient approach, for example, for solving constraints problems, optimisation, and knowledge compilation.

Acknowledgements

I am grateful for valuable comments from the reviewers (in particular for the pathwidth bound) and Barry O’Sullivan, Stefano Bistarelli, Greg Provan, Marc van Dongen, Gene Freuder, Barbara Smith, Peter Stuckey, Peter MacHale and, especially, Alex Ferguson. This material is based upon works supported by the Science Foundation Ireland under Grant No. 00/PI.1/C075.

References

- [Amilhastre *et al.*, 2002] J. Amilhastre, H. Fargier, and P. Marquis. Consistency restoration and explanations in dynamic CSPs—Application to configuration. *Artificial Intelligence*, 135, 2002.
- [Bellman, 1957] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- [Bertele and Brioschi, 1972] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972.
- [Bistarelli *et al.*, 1997] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of the ACM (JACM)*, 44(2):201–236, 1997.
- [Bistarelli *et al.*, 1999] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-based CSPs and Valued CSPs: Frameworks, properties and comparison. *Constraints*, 4(3), 1999.
- [Bistarelli *et al.*, 2002] S. Bistarelli, U. Montanari, and F. Rossi. Soft concurrent constraint programming. In *Proc. 11th European Symposium on Programming (ESOP)*, Lecture Notes in Computer Science (LNCS), pages 53–67. Springer, 2002.
- [Bodlaender, 1993] Hans L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–21, 1993.
- [Boutilier *et al.*, 1996] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proc. UAI96*, pages 115–123, 1996.
- [Bryant, 1986] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.
- [Bryant, 1995] R. E. Bryant. Binary Decision Diagrams and beyond: enabling technologies for formal verification. In *Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design*, pages 236–243, 1995.
- [Darwiche, 2002] A. Darwiche. A logical approach to factoring Belief Networks. In *Proc. KR2002*, pages 409–420, 2002.
- [Dechter and Mateescu, 2004] R. Dechter and R. Mateescu. Mixtures of deterministic-probabilistic networks and their and/or search space. In *Proceedings of UAI04*, 2004.
- [Dechter, 1999] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113, 1999.
- [Freuder, 1991] E. C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *Proc. AAAI-91*, pages 227–233, 1991.
- [McAllester *et al.*, 2004] D. McAllester, M. Collins, and F. Pereira. Case-factor diagrams for structured probabilistic modeling. In *Proceedings of UAI04*, 2004.
- [Pan and Vardi, 2004] G. Pan and M. Vardi. Search vs. symbolic techniques in satisfiability solving. In *Proc. SAT 2004*, 2004.
- [Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.
- [Pesant, 2004] G. Pesant. A regular language membership constraint for finite sequence of variables. In *Principles and Practice of Constraint Programming - CP 2004*, pages 482–495, 2004.
- [Shenoy and Shafer, 1990] P. P. Shenoy and G. Shafer. Axioms for probability and belief function propagation. In *Uncertainty in Artificial Intelligence 4*, pages 575–610, 1990.
- [van Dongen, 2004] M. van Dongen. Computing the frequency of partial orders. In *Principles and Practice of Constraint Programming - CP 2004*, pages 772–776, 2004.
- [Vempaty, 1992] N. R. Vempaty. Solving constraint satisfaction problems using finite state automata. In *Proc. AAAI-92*, pages 453–458, 1992.
- [Wilson, 2004] N. Wilson. Decision diagrams for the computation of semiring valuations. In *Proc. 6th International Workshop on Preferences and Soft Constraints*, 2004.

A Simple Model to Generate Hard Satisfiable Instances

Ke Xu¹

Frédéric Boussemart²

Fred Hemery²

Christophe Lecoutre²

¹NLSDE Lab, School of Computers
Beihang University
Beijing 100083, China
kexu@nldse.buaa.edu.cn

²CRIL-CNRS FRE 2499,
Université d'Artois
Lens, France
<lastname>@cril.univ-artois.fr

Abstract

In this paper, we try to further demonstrate that the models of random CSP instances proposed by [Xu and Li, 2000; 2003] are of theoretical and practical interest. Indeed, these models, called RB and RD, present several nice features. First, it is quite easy to generate random instances of any arity since no particular structure has to be integrated, or property enforced, in such instances. Then, the existence of an asymptotic phase transition can be guaranteed while applying a limited restriction on domain size and on constraint tightness. In that case, a threshold point can be precisely located and all instances have the guarantee to be hard at the threshold, i.e., to have an exponential tree-resolution complexity. Next, a formal analysis shows that it is possible to generate forced satisfiable instances whose hardness is similar to unforced satisfiable ones. This analysis is supported by some representative results taken from an intensive experimentation that we have carried out, using complete and incomplete search methods.

1 Introduction

Over the past ten years, the study of phase transition phenomena has been one of the most exciting areas in Computer Science and Artificial Intelligence. Numerous studies have established that for many NP-complete problems (e.g., SAT and CSP), the hardest random instances occur, while a control parameter is varied accordingly, between an under-constrained region where all instances are almost surely satisfiable and an over-constrained region where all instances are almost surely unsatisfiable. In the transition region, there is a threshold where half the instances are satisfiable. Generating hard instances is important both for understanding the complexity of the problems and for providing challenging benchmarks [Cook and Mitchell, 1997].

Another remarkable progress in Artificial Intelligence has been the development of incomplete algorithms for various kinds of problems. And, since this progress, one important issue has been to produce hard satisfiable instances in order to evaluate the efficiency of such algorithms, as the approach that involves exploiting a complete algorithm in order

to keep random satisfiable instances generated at the threshold can only be used for instances of limited size. Also, it has been shown that generating hard (forced) satisfiable instances is related to some open problems in cryptography such as computing a one-way function [Impagliazzo *et al.*, 1989; Cook and Mitchell, 1997].

In this paper, we mainly focus on random CSP (Constraint Satisfaction Problem) instances. Initially, four “standard” models, denoted A, B, C and D [Smith and Dyer, 1996; Gent *et al.*, 2001], have been introduced to generate random binary CSP instances. However, [Achlioptas *et al.*, 1997] have identified a shortcoming of all these models. Indeed, they prove that random instances generated using these models suffer from (trivial) unsatisfiability as the number of variables increases. To overcome the deficiency of these standard models, several alternatives have been proposed.

On the one hand, [Achlioptas *et al.*, 1997] have proposed a model E and [Molloy, 2003] a generalized model. However, model E does not permit to tune the density of the instances and the generalized model requires an awkward exploitation of probability distributions. Also, other alternatives correspond to incorporating some “structure” in the generated random instances. Roughly speaking, it involves ensuring that the generated instances be arc consistent [Gent *et al.*, 2001] or path consistent [Gao and Culberson, 2004]. The main drawback of all these approaches is that generating random instances is no more quite a natural and easy task.

On the other hand, [Xu and Li, 2000; 2003], [Frieze and Molloy, 2003] and [Smith, 2001] have revisited standard models by controlling the way parameters change as the problem size increases. The alternative model D scheme of [Smith, 2001] guarantees the occurrence of a phase transition when some parameters are controlled and when the constraint tightness is within a certain range. The two revised models, called RB and RD, of [Xu and Li, 2000; 2003] provide the same guarantee by varying one of two control parameters around a critical value that, in addition, can be computed. Also, [Frieze and Molloy, 2003] identify a range of suitable parameter settings in order to exhibit a non-trivial threshold of satisfiability. Their theoretical results apply to binary instances taken from model A and to “symmetric” binary instances from a so-called model B which, not corresponding to the standard one, associates the same relation with every constraint.

The models RB and RD present several nice features:

- it is quite easy to generate random instances of any arity as no particular structure has to be integrated, or property enforced, in such instances.
- the existence of an asymptotic phase transition can be guaranteed while applying a limited restriction on domain size and on constraint tightness. For instances involving constraints of arity k , the domain size is required to be greater than the k^{th} root of the number of variables and the (threshold value of the) constraint tightness is required to be at most $\frac{k-1}{k}$.
- when the asymptotic phase transition exists, a threshold point can be precisely located, and all instances generated following models RB and RD have the guarantee to be hard at the threshold, i.e., to have an exponential tree-resolution complexity.
- it is possible to generate forced satisfiable instances whose hardness is similar to unforced satisfiable ones.

This paper is organized as follows. After introducing models RB and RD, as well as some theoretical results (Section 2), we provide a formal analysis about generating both forced and unforced hard satisfiable instances (Section 3). Then, we present the results of a large series of experiments that we have conducted (Section 4), and, before concluding, we discuss some related work (Section 5).

2 Theoretical background

A constraint network consists of a finite set of variables such that each variable X has an associated domain denoting the set of values allowed for X , and a finite set of constraints such that each constraint C has an associated relation denoting the set of tuples allowed for the variables involved in C .

A solution is an assignment of values to all the variables such that all the constraints are satisfied. A constraint network is said to be satisfiable (sat, for short) if it admits at least a solution. The Constraint Satisfaction Problem (CSP), whose task is to determine if a given constraint network, also called CSP instance, is satisfiable, is NP-complete.

In this section, we introduce some theoretical results taken from [Xu and Li, 2000; 2003]. First, we introduce a model, denoted RB, that represents an alternative to model B. Note that, unlike model B, model RB allows selecting constraints with repetition. But the main difference of model RB with respect to model B is that the domain size of each variable grows polynomially with the number of variables.

Definition 1 (Model RB) A class of random CSP instances of model RB is denoted $RB(k, n, \alpha, r, p)$ where:

- $k \geq 2$ denotes the arity of each constraint,
- n denotes the number of variables,
- $\alpha > 0$ determines the domain size $d = n^\alpha$ of each variable,
- $r > 0$ determines the number $m = r \cdot n \cdot \ln n$ of constraints,
- $1 > p > 0$ denotes the tightness of each constraint.

To build one instance $P \in RB(k, n, \alpha, r, p)$, we select with repetition m constraints, each one formed by selecting k distinct variables and $p \cdot d^k$ distinct unallowed tuples (as p denotes a proportion).

When fixed, α and r give an indication about the growth of the domain sizes and of the number of constraints as n increases since $d = n^\alpha$ and $m = rn \ln n$, respectively. It is then possible, for example, to determine the critical value p_{cr} of p where the hardest instances must occur. Indeed, we have $p_{cr} = 1 - e^{-\alpha/r}$ which is equivalent to the expression of p_{cr} given by [Smith and Dyer, 1996].

Another model, denoted model RD, is similar to model RB except that p denotes a probability instead of a proportion. For convenience, in this paper, we will exclusively refer to model RB although all given results hold for both models.

In [Xu and Li, 2000], it is proved that model RB, under certain conditions, not only avoids trivial asymptotic behaviors but also guarantees exact phase transitions. More precisely, with $\Pr$ denoting a probability distribution, the following theorems hold.

Theorem 1 If $k, \alpha > \frac{1}{k}$ and $p \leq \frac{k-1}{k}$ are constants then

$$\lim_{n \rightarrow \infty} \Pr[P \in RB(k, n, \alpha, r, p) \text{ is sat}] = \begin{cases} 1 & \text{if } r < r_{cr} \\ 0 & \text{if } r > r_{cr} \end{cases}$$

$$\text{where } r_{cr} = -\frac{\alpha}{\ln(1-p)}.$$

Theorem 2 If $k, \alpha > \frac{1}{k}$ and $p_{cr} \leq \frac{k-1}{k}$ are constants then

$$\lim_{n \rightarrow \infty} \Pr[P \in RB(k, n, \alpha, r, p) \text{ is sat}] = \begin{cases} 1 & \text{if } p < p_{cr} \\ 0 & \text{if } p > p_{cr} \end{cases}$$

$$\text{where } p_{cr} = 1 - e^{-\frac{\alpha}{r}}.$$

Remark that the condition $p_{cr} \leq \frac{k-1}{k}$ is equivalent to $ke^{-\frac{\alpha}{r}} \geq 1$ given in [Xu and Li, 2000]. Theorems 1 and 2 indicate that a phase transition is guaranteed provided that the domain size is not too small and the constraint tightness or the threshold value of the constraint tightness not too large. As an illustration, for instances involving binary (resp. ternary) constraints, the domain size is required to be greater than the square (resp. cubic) root of the number of variables and the constraint tightness or threshold value of the tightness is required to be at most 50% (resp. $\approx 66\%$).

The following theorem establishes that unsatisfiable instances of model RB almost surely have the guarantee to be hard. A similar result for model A has been obtained by [Frieze and Molloy, 2003] with respect to binary instances.

Theorem 3 If $P \in RB(k, n, \alpha, r, p)$ and k, α, r and p are constants, then, almost surely¹, P has no tree-like resolution of length less than $2^{\Omega(n)}$.

The proof, which is based on a strategy following some results of [Ben-Sasson and Wigderson, 2001; Mitchell, 2002], is omitted but can be found in [Xu and Li, 2003].

To summarize, model RB guarantees exact phase transitions and hard instances at the threshold. It then contradicts

¹We say that a property holds almost surely when this property holds with probability tending to 1 as the number of variables tends to infinity.

the statement of [Gao and Culberson, 2004] about the requirement of an extremely low tightness for all existing random models in order to have non-trivial threshold behaviors and guaranteed hard instances at the threshold.

3 Generating hard satisfiable instances

For CSP and SAT, there is a natural strategy to generate *forced* satisfiable instances, i.e., instances on which a solution is imposed. It suffices to proceed as follows: first generate a random (total) assignment t and then generate a random instance with n variables and m constraints (clauses for SAT) such that any constraint violating t is rejected. t is then a *forced* solution. This strategy, quite simple and easy to implement, allows generating hard forced satisfiable instances of model RB provided that Theorem 1 or 2 holds. Nevertheless, this statement deserves a theoretical analysis.

Assuming that d denotes the domain size ($d = 2$ for SAT), we have exactly d^n possible (total) assignments, denoted by $t_1, t_2, \dots, t_{d^n}$, and d^{2n} possible assignment pairs where an *assignment pair* $\langle t_i, t_j \rangle$ is an ordered pair of two assignments t_i and t_j . We say that $\langle t_i, t_j \rangle$ satisfies an instance if and only if both t_i and t_j satisfy the instance. Then, the expected (mean) number of solutions $E_f[N]$ for instances that are forced to satisfy an assignment t_i is:

$$E_f[N] = \sum_{j=1}^{d^n} \frac{\Pr[\langle t_i, t_j \rangle]}{\Pr[\langle t_i, t_i \rangle]}$$

where $\Pr[\langle t_i, t_j \rangle]$ denotes the probability that $\langle t_i, t_j \rangle$ satisfies a random instance. Note that $E_f[N]$ should be independent of the choice of the forced solution t_i . So we have:

$$E_f[N] = \frac{\sum_{1 \leq i, j \leq d^n} \Pr[\langle t_i, t_j \rangle]}{d^n \Pr[\langle t_i, t_i \rangle]} = \frac{E[N^2]}{E[N]}.$$

where $E[N^2]$ and $E[N]$ are, respectively, the second moment and the first moment of the number of solutions for random unforced instances.

For random 3-SAT, it is known that the strategy mentioned above is unsuitable as it produces a biased sampling of instances with many solutions clustered around t [Achlioptas *et al.*, 2000]. Experiments show that forced satisfiable instances are much easier to solve than unforced satisfiable instances. In fact, it is not hard to show that, asymptotically, $E[N^2]$ is exponentially greater than $E^2[N]$. Thus, the expected number of solutions for forced satisfiable instances is exponentially larger than the one for unforced satisfiable instances. It then gives a good theoretical explanation of why, for random 3-SAT, the strategy is highly biased towards generating instances with many solutions.

For model RB, recall that when the exact phase transitions were established [Xu and Li, 2000], it was proved that $E[N^2]/E^2[N]$ is asymptotically equal to 1 below the threshold, where almost all instances are satisfiable, i.e. $E[N^2]/E^2[N] \approx 1$ for $r < r_{cr}$ or $p < p_{cr}$. Hence, the expected number of solutions for forced satisfiable instances below the threshold is asymptotically equal to the one for unforced satisfiable instances, i.e. $E_f[N] = E[N^2]/E[N] \approx$

$E[N]$. In other words, when using model RB, the strategy has almost no effect on the number of solutions and does not lead to a biased sampling of instances with many solutions.

In addition to the analysis above, we can also study the influence of the strategy on the distribution of solutions with respect to the forced solution. We first define the distance $d^f(t_i, t_j)$ between two assignments t_i and t_j as the proportion of variables that have been assigned a different value in t_i and t_j . We have $0 \leq d^f(t_i, t_j) \leq 1$.

For forced satisfiable instances of model RB, with $E_f^\delta[N]$ denoting the expected number of solutions whose distance from the forced solution (identified as t_i , here) is equal to δ , we obtain by an analysis similar to that in [Xu and Li, 2000]:

$$\begin{aligned} E_f^\delta[N] &= \sum_{j=1}^{d^n} \frac{\Pr[\langle t_i, t_j \rangle]}{\Pr[\langle t_i, t_i \rangle]} \quad \text{with } d^f(t_i, t_j) = \delta \\ &= \binom{n}{n\delta} (n^\alpha - 1)^{n\delta} \left[\frac{\binom{n-n\delta}{k}}{\binom{n}{k}} + (1-p) \left(1 - \frac{\binom{n-n\delta}{k}}{\binom{n}{k}} \right) \right]^{rn \ln n} \\ &= \exp \left[n \ln n \left(r \ln \left(1 - p + p(1-\delta)^k \right) + \alpha\delta \right) + O(n) \right]. \end{aligned}$$

It can be shown, from the results in [Xu and Li, 2000] that $E_f^\delta[N]$, for $r < r_{cr}$ or $p < p_{cr}$, is asymptotically maximized when δ takes the largest possible value, i.e. $\delta = 1$.

For unforced satisfiable instances of model RB, with $E^\delta[N]$ denoting the expected number of solutions whose distance from t_i (not necessarily a solution) is equal to δ , we have:

$$\begin{aligned} E^\delta[N] &= \binom{n}{n\delta} (n^\alpha - 1)^{n\delta} (1-p)^{rn \ln n} \\ &= \exp \left[n \ln n \left(r \ln(1-p) + \alpha\delta \right) + O(n) \right]. \end{aligned}$$

It is straightforward to see that the same pattern holds for this case, i.e. $E^\delta[N]$ is asymptotically maximized when $\delta = 1$.

Intuitively, with model RB, both unforced satisfiable instances and instances forced to satisfy an assignment t are such that most of their solutions distribute far from t . This indicates that, for model RB, the strategy has little effect on the distribution of solutions, and is not biased towards generating instances with many solutions around the forced one.

For random 3-SAT, similarly, we can show that as r (the ratio of clauses to variables) approaches 4.25, $E_f^\delta[N]$ and $E^\delta[N]$ are asymptotically maximized when $\delta \approx 0.24$ and $\delta = 0.5$, respectively. This means, in contrast to model RB, that when r is near the threshold, most solutions of forced instances distribute in a place much closer to the forced solution than solutions of unforced satisfiable instances.

4 Experimental results

As all introduced theoretical results hold when $n \rightarrow \infty$, the practical exploitation of these results is an issue that must be addressed. In this section, we give some representative experimental results which indicate that practice meets theory even if the number n of variables is small. Note that different values of parameters α and r have been selected in order to illustrate the broad spectrum of applicability of model RB.

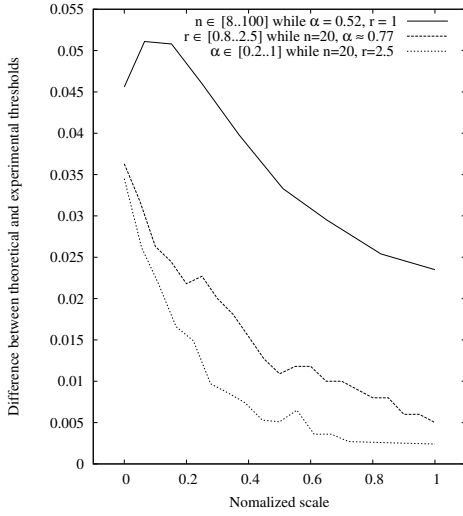


Figure 1: Difference between theoretical and experimental thresholds against α , r and n

First, it is valuable to know in practice, to what extent, Theorems 1 and 2 give precise thresholds according to different values of α , r and n . The experiments that we have run wrt Theorem 2, as depicted in Figure 1, suggest that all other parameters being fixed, the greater the value of α , r or n is, the more precise Theorem 2 is. More precisely, in Figure 1, the difference between the threshold theoretically located and the threshold experimentally determined is plotted against $\alpha \in [0.2, 1]$ ($d \in [2..20]$), against $r \in [0.8, 2.5]$ ($m \in [50..150]$) and against $n \in [8..100]$. Note that the vertical scale refers to the difference in constraint tightness and that the horizontal scale is normalized (value 0 respectively corresponds to $n = 8$, $\alpha = 0.2$ and $r = 0.8$, etc.).

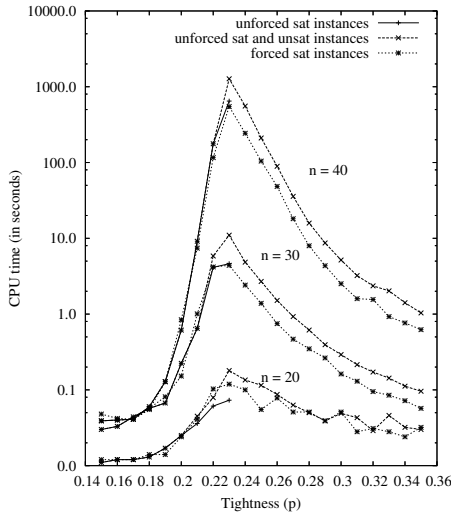


Figure 2: Mean search cost (50 instances) of solving instances in $RB(2, \{20, 30, 40\}, 0.8, 3, p)$

To solve the random instances generated by model RB, we

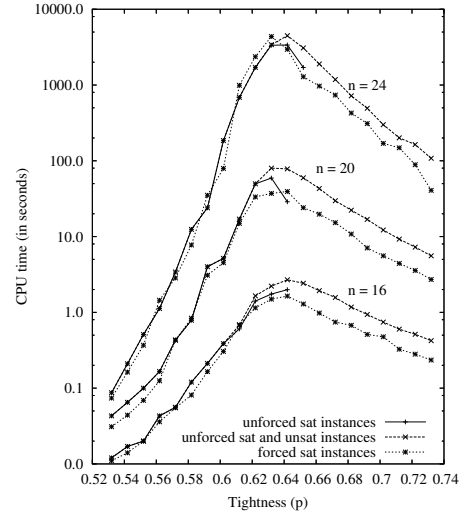


Figure 3: Mean search cost (50 instances) of solving instances in $RB(3, \{16, 20, 24\}, 1, 1, p)$

have used a systematic backtracking search algorithm (MAC) and a local search algorithm (tabu search). Both algorithms have been equipped with a search heuristic that learns from conflicts [Boussemart *et al.*, 2004].

We have studied the difficulty of solving with MAC the binary instances of model RB generated around the theoretical threshold $p_{cr} \approx 0.23$ given by Theorem 2 for $k = 2$, $\alpha = 0.8$, $r = 3$ and $n \in \{20, 30, 40\}$. In Figure 2, it clearly appears that the hardest instances are located quite close to the theoretical threshold and that the difficulty grows exponentially with n (note the use of a log scale). It corresponds to a phase transition (not depicted here, due to lack of space). A similar behavior is observed in Figure 3 with respect to ternary instances generated around the theoretical threshold $p_{cr} \approx 0.63$ for $k = 3$, $\alpha = 1$, $r = 1$ and $n \in \{16, 20, 24\}$.

As the number and the distribution of solutions are the two most important factors determining the cost of solving satisfiable instances, we can expect, from the analysis given in Section 3, that for model RB, the hardness of solving forced satisfiable instances should be similar to that of solving unforced satisfiable ones. This is what is observed in Figure 2.

To confirm this, we have focused our attention to a point just below the threshold as we have then some (asymptotic) guarantee about the difficulty of both unforced and forced instances (see Theorems 5 and 6 in [Xu and Li, 2003]) and the possibility of generating easily unforced satisfiable instances. Figure 4 shows the difficulty of solving with MAC both forced and unforced instances of model RB at $p_{cr} - 0.01 \approx 0.40$ for $k = 2$, $\alpha = 0.8$, $r = 1.5$ and $n \in [20..50]$.

To confirm the inherent difficulty of the (forced and unforced) instances generated at the threshold, we have also studied the runtime distribution produced by a randomized search algorithm on distinct instances [Gomes *et al.*, 2004]. For each instance, we have performed 5000 independent runs. Figure 5 displays the survival function, which corresponds to the probability of a run taking more than x backtracks, of a

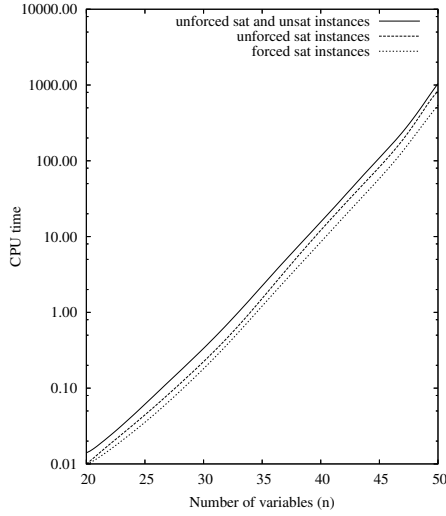


Figure 4: Mean search cost (50 instances) of solving instances in $RB(2, [20..50], 0.8, 1.5, p_{cr} - 0.01)$

randomized MAC algorithm for two representative instances generated at $p_{cr} \approx 0.41$ for $k = 2$, $\alpha = 0.8$, $r = 1.5$ and $n \in \{40, 45\}$. One can observe that the runtime distribution (a log-log scale is used) do not correspond to an heavy-tailed one, i.e., a distribution characterized by an extremely long tail with some infinite moment. It means that all runs behave homogeneously and, therefore, it suggests that the instances are inherently hard [Gomes *et al.*, 2004].

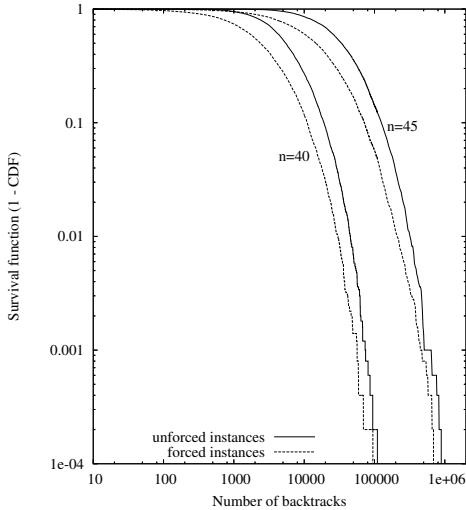


Figure 5: Non heavy-tailed regime for instances in $RB(2, \{40, 45\}, 0.8, 1.5, p_{cr} \approx 0.41)$

Then, we have focused on unforced unsatisfiable instances of model RB as Theorem 3 indicates that such instances have an exponential resolution complexity. We have generated unforced and forced instances with different constraint tightness p above the threshold $p_{cr} \approx 0.41$ for $k = 2$, $\alpha = 0.8$, $r = 1.5$ and $n \in [20..450]$. Figure 6 displays the search effort of a

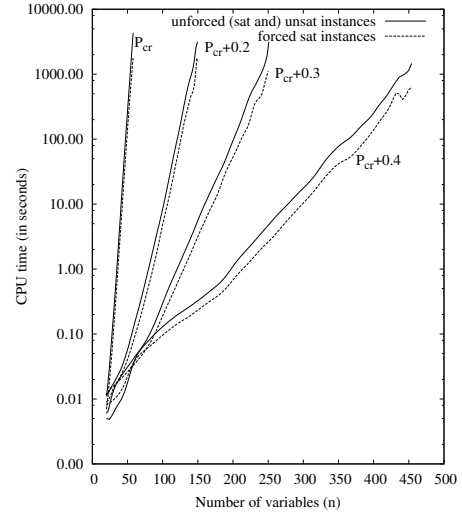


Figure 6: Mean search cost (50 instances) of solving instances in $RB(2, [20..450], 0.8, 1.5, p)$

MAC algorithm to solve such instances against the number of variables n . It is interesting to note that the search effort grows exponentially with n , even if the exponent decreases as the tightness increases. Also, although not currently supported by any theoretical result (Theorems 5 and 6 of [Xu and Li, 2003] hold only for forced instances below the threshold) it appears here that forced and unforced instances have a similar hardness.

Finally, Figure 7 shows the results obtained with a tabu search with respect to the binary instances that have been previously considered with MAC (see Figure 2). The search effort is given by a median cost since when using an incomplete method, there is absolutely no guarantee of finding a solution in a given limit of time. Remark that all unsatisfiable (unforced) instances below the threshold have been filtered out in order to make a fair comparison. It appears that both complete and incomplete methods behave similarly. In Figure 7, one can see that the search effort grows exponentially with n and that forced instances are as hard as unforced ones.

5 Related work

As a related work, we can mention the recent progress on generating hard satisfiable SAT instances. [Barthel *et al.*, 2002; Jia *et al.*, 2004] have proposed to build random satisfiable 3-SAT instances on the basis of a spin-glass model from statistical physics. Another approach, quite easy to implement, has also been proposed by [Achlioptas *et al.*, 2004]: any 3-SAT instance is forced to be satisfiable by forbidding the clauses violated by both an assignment and its complement.

Finally, let us mention [Achlioptas *et al.*, 2000] which propose to build random instances with a specific structure, namely, instances of the Quasigroup With Holes (QWH) problem. The hardest instances belong to a new type of phase transition, defined from the number of holes, and coincide with the size of the backbone.

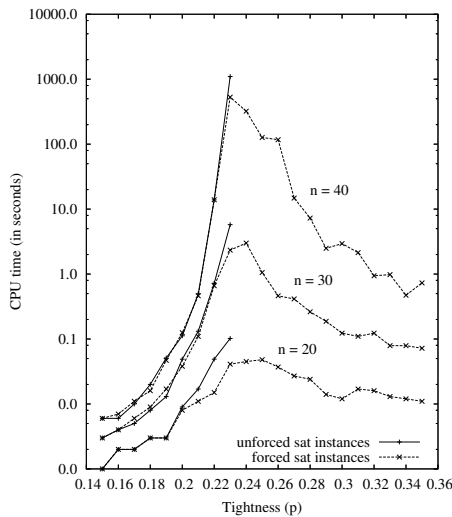


Figure 7: Median search cost (50 instances) of solving instances in $RB(2, \{20, 30, 40\}, 0.8, 3, p)$ using a tabu search

6 Conclusion

In this paper, we have shown, both theoretically and practically, that the models RB (and RD) can be used to produce, very easily, hard random instances. More importantly, the same result holds for instances that are forced to be satisfiable. To perform our experimentation, we have used some of the most efficient complete and incomplete CSP solvers. We have also encoded some forced binary CSP instances of class $RB(2, n, 0.8, 0.8/\ln \frac{4}{3}, p = p_{cr} = 0.25)$ with n ranging from 40 to 59 into SAT ones (using the direct encoding method) and submitted them to the SAT competition 2004². About 50% of the competing solvers have succeeded in solving the SAT instances corresponding to $n = 40$ ($d = 19$ and $m = 410$) whereas only one solver has been successful for $n = 50$ ($d = 23$ and $m = 544$).

Although there are some other ways to generate hard satisfiable instances, e.g. QWH [Achlioptas *et al.*, 2000] or 2-hidden [Achlioptas *et al.*, 2004] instances, we think that the simple and natural method presented in this paper, based on models with exact phase transitions and many hard instances, should be well worth further investigation.

Acknowledgments

The first author is partially supported by NSFC Grant 60403003 and FANEDD Grant 200241. Other authors have been supported by the CNRS, the “programme COCOA de la Région Nord/Pas-de-Calais” and by the “IUT de Lens”.

References

- [Achlioptas *et al.*, 1997] D. Achlioptas, L.M. Kirousis, E. Kranakis, D. Krizanc, M.S.O. Molloy, and Y.C. Stamatou. Random constraint satisfaction: a more accurate picture. In *Proc. of CP'97*, pp. 107–120, 1997.
- [Achlioptas *et al.*, 2000] D. Achlioptas, C. Gomes, H. Kautz, and B. Selman. Generating satisfiable problem instances. In *Proc. of AAAI'00*, pp. 256–301, 2000.
- [Achlioptas *et al.*, 2004] D. Achlioptas, H. Jia, and C. Moore. Hiding satisfying assignments: two are better than one. In *Proc. of AAAI'04*, pp. 131–136, 2004.
- [Barthel *et al.*, 2002] W. Barthel, A.K. Hartmann, M. Leone, F. Ricci-Tersenghi, M. Weigt, and R. Zecchina. Hiding solutions in random satisfiability problems: a statistical mechanics approach. *Physical review letters*, 88(18), 2002.
- [Ben-Sasson and Wigderson, 2001] E. Ben-Sasson and A. Wigderson. Short proofs are narrow - resolution made simple. *Journal of the ACM*, 48(2):149–169, 2001.
- [Boussemart *et al.*, 2004] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais. Boosting systematic search by weighting constraints. In *Proc. of ECAI'04*, pp. 146–150, 2004.
- [Cook and Mitchell, 1997] S.A. Cook and D.G. Mitchell. Finding hard instances of the satisfiability problem: a survey. *DIMACS Ser. Disc. Math. & TCS*, 35, 1997.
- [Frieze and Molloy, 2003] A.M. Frieze and M. Molloy. The satisfiability threshold for randomly generated binary constraint satisfaction problems. In *Proc. of Random'03*, pp. 275–289, 2003.
- [Gao and Culberson, 2004] Y. Gao and J. Culberson. Consistency and random constraint satisfaction models with a high constraint tightness. In *CP'04*, pp. 17–31, 2004.
- [Gent *et al.*, 2001] I.P. Gent, E. MacIntyre, P. Prosser, B.M. Smith, and T. Walsh. Random constraint satisfaction: flaws and structure. *Constraints*, 6(4):345–372, 2001.
- [Gomes *et al.*, 2004] C.P. Gomes, C. Fernández, B. Selman, and C. Bessiere. Statistical regimes across constrainedness regions. In *Proc. of CP'04*, pp. 32–46, 2004.
- [Impagliazzo *et al.*, 1989] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random number generation from one-way functions. In *Proc. of STOC'89*, pp. 12–24, 1989.
- [Jia *et al.*, 2004] H. Jia, C. Moore, and B. Selman. From spin glasses to hard satisfiable formulas. In *SAT'04*, 2004.
- [Mitchell, 2002] D.G. Mitchell. Resolution complexity of random constraints. In *Proc. of CP'02*, pp. 295–309, 2002.
- [Molloy, 2003] M. Molloy. Models for random constraint satisfaction problems. *SIAM Journal of computing*, 32(4):935–949, 2003.
- [Smith and Dyer, 1996] B.M. Smith and M.E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81:155–181, 1996.
- [Smith, 2001] B.M. Smith. Constructing an asymptotic phase transition in random binary constraint satisfaction problems. *TCS*, 265:265–283, 2001.
- [Xu and Li, 2000] K. Xu and W. Li. Exact phase transitions in random constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 12:93–103, 2000.
- [Xu and Li, 2003] K. Xu and W. Li. Many hard examples in exact phase transitions with application to generating hard satisfiable instances. CoRR Report cs.CC/0302001, 2003.

²<http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/benchmarks.htm>

A Novel Local Search Algorithm for the Traveling Salesman Problem that Exploits Backbones*

Weixiong Zhang and Moshe Looks

Department of Computer Science and Engineering
Washington University in Saint Louis
Saint Louis, MO 63130, USA
Email: zhang@cse.wustl.edu

Abstract

We present and investigate a new method for the Traveling Salesman Problem (TSP) that incorporates backbone information into the well known and widely applied Lin-Kernighan (LK) local search family of algorithms for the problem. We consider how heuristic backbone information can be obtained and develop methods to make biased local perturbations in the LK algorithm and its variants by exploiting heuristic backbone information to improve their efficacy. We present extensive experimental results, using large instances from the TSP Challenge suite and real-world instances in TSPLIB, showing the significant improvement that the new method can provide over the original algorithms.

1 Introduction

Given a set of cities and the distances between them, the traveling salesman problem (TSP) is to find a complete, minimal-cost tour visiting each city once. The TSP is a well-known NP-hard problem with many real-world applications, such as jobshop scheduling and VLSI routing [5]. The TSP has often served as a touchstone for new problem-solving strategies and algorithms; and many well-known combinatorial algorithms were first developed for the TSP, including the Lin-Kernighan local search algorithm [10]. In this paper, we consider the symmetric TSP, where the distance from a city to another is the same as the distance in the opposite direction.

Local search (LS) algorithms have been shown very effective for the TSP. In LS algorithms for the TSP, one defines a neighborhood structure of tours, in which tour T is a neighbor of tour T' if T can be changed into T' by some local perturbation, such as by exchanging a pair (2-Opt) or triplet (3-Opt) of edges between cities [9; 10]. Starting from a complete tour, LS repeatedly improves the current tour until it is the best among its neighbors; such a tour is called a local minimum. This process can be applied multiple times using different initial tours. Although LS algorithms do not guarantee the optimality of the best solution found, they have been routinely applied in practice, and are among the best methods for the TSP. The best known LS algorithm for the TSP is the Lin-Kernighan (LK) algorithm [10]. Since its inception three decades ago, this classic algorithm has inspired much research

on LS. Today, the best LS algorithms for the TSP are its variants, which include chained and iterated LK algorithms [11; 9; 6]. These algorithms can provide high-quality, near optimal solutions for problems with several thousand cities [9]. It is nontrivial to find solutions better than those from the Lin-Kernighan family. Nonetheless, improving these algorithms is of practical importance; even small improvements can have substantial financial impacts for many applications, as in manufacturing, where selected TSP tours need to be routinely traversed.

A major deviation of most modern LS variants from the LK algorithm is the use of starting tours that are closely related to the previous local minimum or the best tour found so far, rather than independently generated [9]. Such starting tours can be typically generated by perturbations to the final tours of previous runs using neighborhood structures different from the ones used by the main procedure. For instance, double-bridge 4-Opt moves have been used extensively for this purpose [9].

The intuition behind using chained starting tours in an iterated LS is that high-quality local minima tend to reside in a small vicinity of a neighborhood structure. Therefore, it is more effective and efficient to search for a better local minimum from a known one. This intuition is supported by the observation that local minima of a good LS method usually share many common partial structures [1; 2; 10]. Such observations have also led to the “big valley” hypothesis [1], which suggests that high-quality local minima tend to have many edges in common, forming a single cluster around the optimal tour(s), and that a better local minimum tends to have more common edges shared with an optimal solution than a worse local minimum.

Besides the iterated LS methods, common structures of local minima have been exploited in at least two other ways. The first is reduction [10], which first collects a small set of local minima and then locks in the edges appearing in all of them in the subsequent runs. This method has two effects: it can speed up the search, as the problem becomes smaller, and provides a means of directing search among a set of otherwise indistinguishable tours [10]. In [10], Lin and Kernighan reported experimental results on instances up to 318 cities using reduction. The second and more recent way to exploit common structures of local minima is called tour merging [2]. Given a set of local minima, this method constructs a graph containing the union of their edges. Thanks to the large number of shared edges among the local minima, the resulting graph is sparse. An optimal tour within the sparse graph is then uncovered as

*The research was supported NSF grant EIA-0113618.

an approximate solution, which is very often optimal. Our recent algorithm on maximum satisfiability (max-SAT) also explicitly exploited the information in a cluster of local minima and incorporated it in the Walksat algorithm [15], resulting in a significant performance improvement on diverse large max-SAT problems [16]. The current research was inspired by these previous results.

Our research was also motivated by the recent advances in characterizing typical case features of combinatorial problems by their phase transitions and backbones [7; 12]. A problem of fundamental importance and practical interest is to utilize inherent problem information, such as phase transitions and backbones, in a search algorithm to cope with problem difficulty. The research along this line is limited. Besides the published work of utilizing backbone information in local search for SAT and max-SAT [16], previously published results include exploiting phase transitions in tree search problems in an approximation algorithm [13] and applying heuristic backbone information in a systematic search for SAT [3].

Our new method explicitly exploits the structure of the local minima of LS algorithms, namely the possible backbone information embedded in the local minima, to improve the performance of the algorithm. Here, a backbone variable for a TSP refers to an edge between two cities that appears in all optimal TSP tours. Unlike the reduction and tour merging methods, our new heuristic does not freeze the common edges in all local minima in subsequent searches; it rather applies estimated backbone information to guide a local search to the region of the search space that is more likely to have better approximate, and hopefully optimal, solutions. Specifically, we treat local minima from a local search as if they were optimal solutions, and use edge appearance frequencies to estimate the probabilities of backbone variables. We then apply the estimated backbone probabilities to alter the perturbations made by the local search algorithm so that a variable having a higher backbone probability will be less likely to be swapped out of the current tour than a variable having a smaller backbone probability, and conversely, will be more likely to be swapped in.

The paper is organized as follows: in Section 2, we describe the general idea of backbone-guided local search for the TSP. We then in Section 3 discuss the backbone-guided LK algorithms. We present the experimental results in Section 4, discuss related work in Section 5, and conclude in Section 6.

2 Backbone Guided Local Search

If all backbone variables of a TSP were known, they could provide a useful clue to how edges between two cities should be swapped in or out during a local search. If an edge is a part of the backbone, i.e., it appears in all optimal solutions, obviously the edge should be swapped in if it is not included in the current tour. Moreover, we can extend the concept of backbone to backbone frequency of an edge, which is the percentage of optimal solutions that have the edge. This means that a backbone edge has a backbone frequency of one and an edge that does not appear in any optimal solutions has a backbone frequency of zero. Therefore, the backbone frequency of an edge is an indicator of how often that edge should be swapped in (or out) if it is (or is not) part of the current tour. This can be exploited as a heuristic for selecting edges in local search.

Unfortunately, exact backbone frequencies are hard to come by without solving the problem exactly. The idea to by-

pass this problem was inspired by the fact that many approximation algorithms for the TSP, the LK algorithm and its variants in particular, have superior performance. They can often reach local minima that are within a small percentage of a global optimum and have common structures shared with a global optimal solution, as discussed in [1; 9]. Therefore, we can treat local minima as if they were optimal to compute *pseudo-backbone frequencies* to approximate the true backbone frequencies. We call this general approach *backbone guided local search* or BGLS.

We define the pseudo-backbone frequency of an edge as the frequency of its appearances in the local minima sampled. Thus, if $\mathcal{S}$ is the set of local minima, and a given edge e appears in a subset $\mathcal{S}_e$ of $\mathcal{S}$, the pseudo-backbone frequency of e is simply $|\mathcal{S}_e|/|\mathcal{S}|$.

Which local minima to use will affect the quality of the pseudo-backbone frequencies. Ideally, we want the local minima to be an unbiased sample of all high-quality approximate solutions. One leading factor in reaching such an ideal is the set of initial tours: the more distinct the starting tours are, the more different the final tours will generally be. Therefore, even though local minima reached from greedily generated starting tours are superior to those reached from random starting tours, a pseudo-backbone constructed from the latter generally leads to better overall performance.

Pseudo-backbone information can be incorporated in LS algorithms to “bias” the search. In LS, moves are evaluated by the difference between the total cost of the edges to be removed from the tour and the total cost of the edges to be added. If this value is positive, the move is taken. In backbone guided search, we can make biased moves in two different ways; one only uses pseudo-backbone frequencies and the other combines pseudo-backbone frequencies and the distances between cities. Let $\mathcal{B}$ be the set of backbone edges of a TSP and $\mathcal{T}$ the set of local minima from which pseudo-backbone frequencies were computed. Let X and Y be the candidate set of edges to be removed and added, respectively, at a step of searching for a k -Opt move. We prefer to replace X by Y if X has a smaller possibility of having more backbone variables than Y . If we assume edges to be independent of each other, we prefer to replace X by Y if $\sum_{x_i \in X} P(x_i \in \mathcal{B}|\mathcal{T}) < \sum_{y_i \in Y} P(y_i \in \mathcal{B}|\mathcal{T})$, where $P(x_i \in \mathcal{B}|\mathcal{T})$ is the backbone frequency of edge x_i , computed from the set of local minima $\mathcal{T}$. This method has been shown effective on maximum satisfiability [16].

Unfortunately, local search based exclusively on local perturbations using merely pseudo-backbone frequencies is not very effective on the TSP. One possible factor contributing to this discrepancy between the TSP and max-SAT is the different sizes of their search spaces. The TSP has $2^{|E|} = O(2^{n^2})$ states in its search space, where $|E| = n(n-1)/2$ is the number of edges for an n city TSP. These states are embedded in a constraint structure, in which, for example, taking one edge preventing many other edges from being taken. In comparison, the search space of max-SAT has only 2^n states for n Boolean variables. The estimated backbone frequencies could thus be less reliable on the TSP than on max-SAT. The deficiency of local perturbations based purely on pseudo-backbone frequencies indicates that the actual intercity distances should not be ignored for the TSP. This constitutes one of the main differences between backbone-guided local search for the TSP and that for max-SAT.

In the LK algorithm, moves are evaluated by summing the costs of the edges to be removed from the current tour, and subtracting the sum of the costs of the edges to be added. The same evaluation principle can be applied in backbone guided LS, but with cost computed differently. Instead of taking the cost of a given edge to be the distance between two cities, w , it is taken to be $w \cdot (1 - p)$, where p is the pseudo-backbone frequency of that edge. Thus, the cost of an edge will decrease linearly in proportion to its pseudo-backbone frequency. The extreme cases are edges not in the pseudo-backbone, which have their original costs ($p = 0$), and edges in all local minima, which their costs set to zero ($p = 1$).

When this method is used, the results can be improved even more by carrying out regular LK search using original distances from the local minima found by the biased search. This is effective because the biased search actually searches in a new TSP instance that has been created from the original by applying a “pseudo-backbone transform” to its edge weights. Local minima in this new instance will not generally correspond to local minima in the original, so continued search of the original graph will improve the results, even after biased search has reached a local minimum.

Furthermore, pseudo-backbone information can be employed to generate starting tours. It is known that the effectiveness and speed of local search can be improved by using greedy starting tours [9]. The greedy tour construction begins by randomly picking a starting city, and adding the shortest edge exiting the city to the tour. Then, edges are greedily added one-by-one until the tour is complete. We can modify this process to naturally utilize the pseudo-backbone by redefining the “best” edge in terms of the pseudo-backbone-transformed weights, rather than the original weights. This gives us a greedy pseudo-backbone tour generation heuristic.

3 Backbone Guided LK and ILK

Applying the above ideas and considerations to LK, we have the backbone guided LK algorithm (BGLK). Similar to LK, BGLK runs many cycles, each of which starts from a new starting tour and reaches a local minimum. Different from LK, BGLK has two phases. The first is a learning phase that runs a fixed number of iterations of original LK with *random* starting tours. The local minima from these runs are used to compute pseudo-backbone frequencies. The second is a backbone-guided improvement phase where biased k -Opts are utilized. In addition, in the second phase, biased starting tours can be used as well to improve local minima and speed up the search. In our experiments, we found that setting aside 30% of the total runs for learning was generally effective.

A few issues must be dealt with to combine pseudo-backbone utilization with ILK, the iterated LK algorithm, in deriving iterative BGLK (IBGLK). First, pseudo-backbone must be constructed from “unbiased samples” of local minima. Since each local minimum found via ILK typically differs in only a small number of edges from its progenitor, a sampling of local minima from such a chained process is biased, and should not be averaged into the backbone. To deal with this, we construct a backbone in our experiments from 30 independent rounds, i.e., restarts, of ILK, each of which is allowed a smaller number of iterations, such as 1% of the total number of iterations. Another issue is that we would like to follow each round of BGLK by a round of regular search using the original distances. However, since BGLK uses “weighted” costs,

it is impossible to ignore potential k -Opts involving only unchanged cities, as when moving between successive rounds of ILK with only a single search method. Our solution to this has been to alternate between BGLK and regular LK at a higher level of granularity: one of these search mechanisms is employed until it “fails”, at which point we begin employing the alternate mechanism until it too “fails”, then switch back to the original, and the alternation repeats. In this context, we have defined failure at going through k rounds (restarts) of optimization without finding an improved tour. For our experiments, we have set $k = N/20$, where N is the number of cities.

4 Experimental Evaluation

We have carried out two sets of experiments to study different ways of applying pseudo-backbone information in BGLS. In our experiments we used random instances from the TSP Challenge suite [8], which includes problem classes of uniform Euclidean (Uniform), clustered Euclidean (Clustered), and distance matrix (Matrix). We have also used the large instances from the TSPLIB [14].

We compared BGLK and IBGLK against LK, ILK, the reduction method of [10], and the search space smoothing method of [4]. The smoothing method is related because it modifies the distances of TSP instances in an attempt to make it easier for local search to eventually reach local minima of higher quality. It uses a fixed formula, $d' = \bar{d} \pm |d - \bar{d}|^\alpha$, to transform distance from d to d' , where $\bar{d}$ is the average distance over all original distances, and α takes a series of decreasing values, typically 5, 4, $\dots$, 1 to help slowly move from an initially “smooth” instance, in which all distances look similar to one another, to the actual problem. We implemented this method in the LK algorithm for the test.

The particular version of LK-based algorithms that we used was implemented and provided by Johnson and McGeoch, described and analyzed in [9]. We leave the details of the implementation to its original description, while simply pointing out that all of our tests were carried out with the default settings for this implementation, namely length-20 neighbor lists for all levels of the search, don’t-look bits, and the 2-Level Tree tour representation. We incorporated the reduction method and the space smoothing method in LK to generate two variants. We have also followed Johnson and McGeoch [9] in configuring ILK, allowing it the most flips, $10N$, where N is the problem size. This is our baseline ILK algorithm, abbreviated as *ILK-1-run*. In addition, we used two different configurations of ILK in our experiments: the best of five runs of $2N$ iterations (i.e., ILK with four random restarts), which is named as *ILK-5-runs*, and the best of ten runs of N iterations, which is called *ILK-10-runs*. We configured IBGLK as follows. It first runs 30 rounds of ILK, each of which starts from a random tour and is allowed $N/10$ iterations. It then executes 70 rounds of ILK with biased moves and biased greedy initial tours, each of which is also allocated $N/10$ iterations. All algorithms used the same total number of flips to give a relatively fair comparison.

Our experiments were run on 1.6 and 2.0 GHz AMD Athlon machines with 2 gigabytes of memory. All runtimes are normalized for a 500 Mhz Alpha, as described in [9], so that our results and the previous results can be directly compared.

4.1 The TSP Challenge suite

We experimentally validate our claim that BGLS allows local search methods to reach better local minima, using problem

		N=1000	N=3162	N=10K	N=31K
U	Opti	0.74 —	0.70 —	—	—
	LK	1.38 (8.8)	1.56 (33.0)	1.79 (83.2)	1.83 (232.9)
	Redu	1.30 (5.2)	1.59 (18.1)	1.80 (43.8)	1.84 (117.0)
	Smoo	1.21 —	1.46 —	1.58 —	1.55 —
	BGLK	1.07 (12.1)	1.15 (53.4)	1.2 (147.0)	1.19 (441.2)
C	Opti	0.54 —	0.59 —	—	—
	LK	1.81 (226.9)	3.42 (625.0)	5.67 (1241.7)	5.63 (3013.3)
	Redu	1.51 (103.8)	3.36 (290.7)	5.19 (494.9)	6.01 (1132.8)
	Smoo	2.16 —	3.70 —	6.08 —	6.08 —
	BGLK	1.30 (322.3)	2.54 (1032.3)	4.90 (2120.8)	4.58 (5238.7)
M	Opti	0.02 —	0.00 —	0.00 —	—
	LK	2.45 (19.9)	3.87 (63.1)	5.30 (339.3)	—
	Redu	2.24 (16.0)	3.72 (49.1)	5.19 (270.3)	—
	Smoo	1.60 —	2.79 —	4.27 —	—
	BGLK	0.90 (30.9)	1.77 (79.0)	2.58 (405.0)	—

Table 1: Comparison of LK, reduction, smoothing and BGLK on problems from the Challenge suite on Uniform (U), Clustered (C) and matrix (M) problem instances. The numbers are tour costs over Held-Karp bounds in %, and normalized CPU times in seconds (in parentheses).

instances from the Challenge suite. We examine the results from four different perspectives.

In the first test, we compared BGLK against LK, LK with reduction, and LK with search space smoothing. The LK algorithm has 100 runs; the reduction method uses an initial probing stage of 30-run of LK, followed by 70-run of LK with reduction; the smoothing method has 20 runs total, each of which runs LK five times, for $\alpha = 5, 4, \dots, 1$. These algorithms all used greedy initial tours and had a total of 100 runs. BGLK executes an initial 30-run of LK, using random starting tours, to learn pseudo-backbone, and then 70 runs of LK with biased moves and biased initial tours. The sizes N of problem instances are 1000, 3162, 10000, and 31623, increased by a factor of $\sqrt{10}$, following the scheme proposed and used in [9]. For each problem class and size, 100 random problem instances were used for each algorithm. Table 1 shows the averaged results. For comparison, percent over the Held-Karp bounds for optimal tours is listed when known.

As shown, while the reduction and smoothing methods provide modest improvements over LK in certain instance classes and sizes, they are respectively the worst among all algorithms tested on the Euclidean and clustered Euclidean classes. In contrast, BGLK is consistently superior, outperforming all the other three methods across all instance classes and sizes. The improvement of BGLK over these algorithms varies from 0.3% to 1.7% on the largest problem instances of these classes. Note that the average running time of BGLK was no more than twice the time by LK. Interestingly, BGLK actually ran *faster* than LK on some instances, presumably due to its faster focus on promising areas. Note also that the reduction method’s running times significantly less than that of IL, while the smoothing method’s runtimes are essentially identical to LK’s.

In the second experiment, we compared IBGLK against ILK (ILK-1-run) and its two variants (ILK-5-runs and ILK-10-runs). We used the same problem instances as in the first experiment, and computed the results the same way as in that experiment. The results are shown in Table 2, where we adopted the same reporting scheme as in Table 1. As shown, the two ILK variations do not provide much improvement over the baseline ILK, while IBGLK is able to push significantly closer to optimal than ILK for most cases where ILK does not perform very well. For uniform Euclidean instances, as ILK does not reach the optimum for these instances, IBGLK has a good chance

		N=1000	N=3162	N=10K	N=31K
U	Optimal	0.74 —	0.70 —	—	—
	ILK-1-run	0.82 (58)	0.78 (294)	0.81 (1619)	0.89 (8371)
	ILK-5-run	0.77 —	0.80 —	0.80 —	0.85 —
	ILK-10-run	0.78 —	0.82 —	0.86 —	0.98 —
	IBGLK	0.77 (78)	0.77 (411)	0.77 (2311)	0.77 (11760)
C	Optimal	0.54 —	0.59 —	—	—
	ILK-1-run	0.55 (1469)	0.62 (6645)	0.84 (22859)	1.14 (88077)
	ILK-5-run	0.54 —	0.62 —	0.76 —	1.06 —
	ILK-10-run	0.54 —	0.63 —	0.73 —	1.01 —
	IBGLK	0.55 (1281)	0.62 (5764)	0.78 (23311)	0.94 (93759)
M	Optimal	0.02 —	0.00 —	0.00 —	—
	ILK-1-run	0.58 (106)	1.34 (860)	2.70 (8230)	—
	ILK-5-run	0.85 —	1.34 —	2.09 —	—
	ILK-10-run	0.89 —	1.94 —	3.44 —	—
	IBGLK	0.20 (139)	0.79 (1221)	1.48 (11360)	—

Table 2: Comparison of ILK (ILK-1-run), its variations (ILK-5-run and ILK-10-run), and IBGLK on problems from the Challenge suite. The legend and interpretation of the table are the same as those for Table 1.

to improve tour quality. IBGLK’s performance appears to remain constant (i.e., 0.77%) relative to the Held-Karp bound as the problem size increases, while ILK’s performance degrades. Although IBGLK runs longer than ILK, due to the extra time needed to construct and apply the pseudo-backbone, the overhead never exceeds 50% of ILK’s runtime, and it appears that IBGLK’s and ILK’s runtimes are asymptotically close to each other. For clustered problem class, IBGLK, ILK and its variations’ performances are compatible with one another. Interestingly, IBGLK actually runs *faster* than ILK on some of the clustered instances. For this instance class, IBGLK’s lack of significant improvement over ILK can be attributed to the fact that ILK is already effective and close to the optimum; there is little room for improvement. On the distance matrix class, similar to BGLK versus LK, IBGLK is significantly superior to ILK on each of the individual instances of different sizes, and reduces average tour costs by more than 0.6% on the largest ($N = 10,000$) instances tested. In addition, among ILK and its variants, ILK-1-run has the fastest running times, and its two variants are 5%-10% slower.

In the third experiment, we considered relatively small instances of 1,000 cities and compared the local minima from LK, ILK and IBGLK against the optimal solutions. We used 100 instances for each of the problem classes in the Challenge suite. For each instances and each algorithm, we averaged three runs with different random seeds. As the result shows, IBGLK generally produces solutions of higher quality on average. Interestingly, it seems that more powerful search methods produce not only better, but more diverse solutions, as quantified by the standard deviations of the Hamming distances between the structures of local minima and the optimal solutions (data not shown). Figure 1 graphically shows relative performances of ILK and IBGLK on individual instances in the test, indicating that IBGLK is favorable to ILK on most of the problem instances.

Additional insight can be gained by an inspection of the anytime behavior of these algorithms. As shown in Figure 2, BGLK’s anytime behavior is remarkably different from that of LK and its variants. To begin with, LK outpaces BGLK, since LK is being run from greedy starting tours, while BGLK must be run from random tours to construct a reliable pseudo-backbone (see Section 3). But once BGLK begins using the pseudo-backbone to guide the search, its rate of descent increases, and the tables are turned on their performance. Qual-

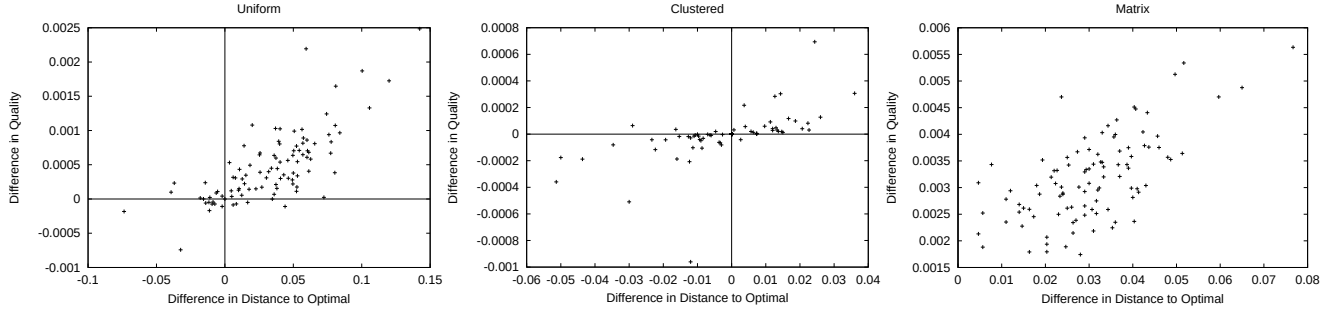


Figure 1: Comparison of local minima from ILK and IBGLK for 100 problems with 1,000 cities each. Points with $x > 0$ indicate that IBGLK's solutions are of higher quality, points with $y > 0$ indicate that IBGLK's solutions are closer to optimal. Both axes are normalized.

itatively similar anytime performance results have been found on large problem sizes and when comparing IBGLK with ILK.

4.2 Instances from the TSPLIB

Table 3 contains the results for ILK, its variants, and IBGLK, on all the instances in the TSPLIB that have at least 1,000 cities. On many of the instances, unfortunately, ILK is already at or near the optimum, so there is very little room for improvement. Again, in most places where further improvements are possible, IBGLK produces them. On this set of instances, IBGLK championed on 20 of them (the ones in bold and underlined), whereas ILK and its variants came first on 12 instances (with some overlaps). Note that on the largest instances (the ones with more than 10,000 cities), IBGLK produces significant gains over ILK. The runtimes were from our AMD Athelon machines and then normalized to a 500MHz Alpha as suggested in [9].

5 Related Methods and Discussions

The most closely related work is our previous work on backbone guided Walksat for maximum satisfiability [16]. The backbone guided local search developed here for the TSP follows the same principles developed there; and thus can be viewed as an innovative extension to the work in [16]. This research was also inspired by and builds upon the previous results of the “big valley” hypothesis on the clustering of local minima from the family of the Lin-Kernighan algorithm and its variants [1; 2; 10].

Two pieces of previous work resemble BGLK in some way. The first is the reduction method by Lin and Kernighan [10]. As discussed earlier, reduction “locks in” the common edges in all local minima. This has two effects: it can speed up the search, as the problem becomes smaller, and provides a means of directing search among a set of otherwise indistinguishable tours. The main limitation of this method is that it is brittle, depending on the quality of the “locked in” edges. If a “locked in” edge turns out not to be part of the backbone, no optimal tour will be found. Moreover, information such as “edge (a, b) appears in 90% of all local minima” cannot be utilized, thus it simply disregards potentially useful information of backbone frequencies. The results in [10], up to 318 cities, is too limited to provide a fair assessment of this method. Our results in Section 4 revealed that reduction is not competitive.

BGLS is more general than reduction by providing a remedy to the problem of brittleness and making use of information embedded in backbone frequencies. BGLS is a random strategy, in which edges that appear in all local minima may still be

Name	% over the Held-Karp bound					runtime in seconds	
	ILK-1	ILK-5	ILK-10	IBGLK	Optimal	ILK-1	IBGLK
dsj1000	<u>0.61</u>	0.62	0.62	0.62	0.61	2097	2066
pr1002	<u>0.89</u>	<u>0.89</u>	1.04	<u>0.89</u>	0.89	300	345
si1032	0.15	<u>0.08</u>	<u>0.08</u>	<u>0.08</u>	0.08	610	772
u1060	<u>0.65</u>	0.67	0.67	0.67	0.65	870	873
vm1084	1.37	<u>1.35</u>	<u>1.35</u>	<u>1.35</u>	1.33	403	403
pcb1173	0.97	0.98	<u>0.97</u>	<u>0.96</u>	0.96	161	213
d1291	1.31	1.23	<u>1.18</u>	1.26	1.18	1099	1086
rl1304	<u>1.55</u>	<u>1.55</u>	<u>1.55</u>	<u>1.55</u>	1.55	457	504
rl1323	1.66	1.66	<u>1.65</u>	<u>1.65</u>	1.65	470	582
nrv1379	0.47	0.51	<u>0.43</u>	0.49	0.43	274	440
fl400	<u>1.74</u>	<u>1.74</u>	<u>1.74</u>	<u>1.74</u>	1.74	22548	19317
u1432	0.44	0.38	0.38	<u>0.29</u>	0.29	319	617
fl577	1.68	<u>1.66</u>	<u>1.66</u>	<u>1.66</u>	1.66	9072	10430
d1655	1.19	0.92	0.95	<u>0.91</u>	0.91	1808	1869
vm1748	<u>1.35</u>	1.38	<u>1.35</u>	1.36	1.35	893	829
u1817	1.08	1.13	1.17	<u>1.06</u>	0.90	468	680
rl1889	1.74	1.64	<u>1.55</u>	1.77	1.55	1028	1110
d2103	<u>1.44</u>	<u>1.44</u>	<u>1.44</u>	<u>1.44</u>	1.44	3407	8669
u2152	0.96	<u>0.77</u>	0.79	0.81	0.62	597	808
u2319	0.13	0.13	0.13	<u>0.06</u>	0.02	1182	2091
pr2392	<u>1.27</u>	1.28	1.37	1.36	1.22	503	715
pcb3038	0.91	<u>0.88</u>	0.93	0.94	0.81	854	1178
fl795	<u>1.04</u>	<u>1.04</u>	1.05	1.07	1.04	45789	52460
fnl4461	0.66	0.67	0.69	<u>0.63</u>	0.55	1613	2505
rl5915	<u>1.58</u>	1.62	<u>1.58</u>	1.62	1.56	2963	4766
rl5934	1.53	<u>1.39</u>	1.56	1.54	1.38	3714	5250
pla7397	<u>0.61</u>	0.72	0.63	0.63	0.58	18338	20716
rl11849	<u>1.13</u>	1.17	1.21	1.15	1.02	9421	15301
usa13509	0.78	0.76	0.83	<u>0.72</u>	0.66	20829	26339
brd14051	0.61	0.63	0.66	<u>0.58</u>	—	28587	35048
d15112	0.66	0.68	0.72	<u>0.63</u>	0.52	21481	29709
d18512	0.63	0.64	0.70	<u>0.59</u>	—	21995	31916
pla33810	0.68	0.66	0.70	<u>0.65</u>	—	148387	150455
pla85900	0.56	0.58	0.62	<u>0.52</u>	—	256808	247597
# of bests	12	11	12	<u>20</u>			

Table 3: Comparison of all instances in TSPLIB with at least 1,000 cities. Numbers are tour costs over Held-Karp bounds in % and normalized runtimes in seconds (for a 500 Mhz Alpha).

swapped out of the current tour during the search, albeit with a slim possibility. This allows more tours be explored while still maintaining a focused search.

The second related work is search space smoothing [4], which was briefly described in Section 4. Gu and Huang only experimented with this idea with a 2-Opt algorithm and on small random Euclidean problems with no more than 100 cities. They did not compare their method to other techniques. Our results in Section 4 showed that smoothing outperforms LK on uniform and matrix instances. However, it is less efficient than BGLK and ILK.

Although both smoothing and backbone guided search all modify distances, they are fundamentally different. First, modifying distances is just one of the end products of the backbone guided search for the TSP. In fact, when applied to maximum satisfiability, backbone guided local search did not change

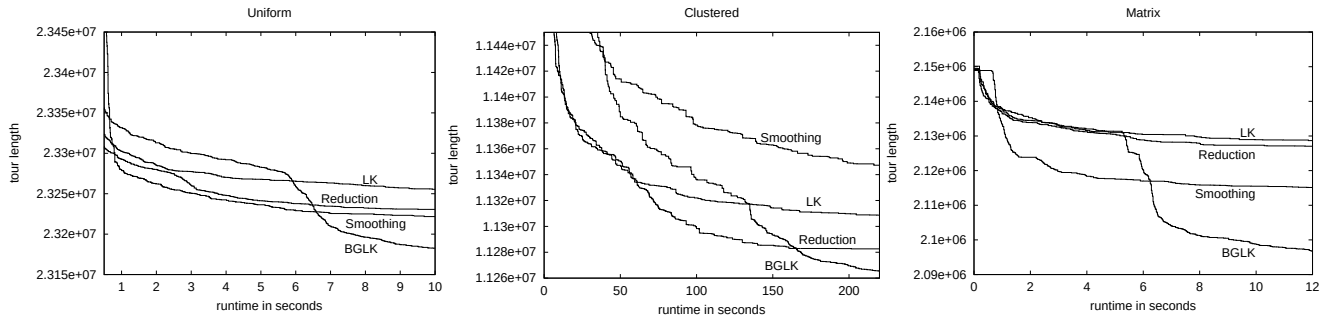


Figure 2: Anytime performance of LK, LK with reduction, LK with smoothing, and BGLK, for random instance from the TSP Challenge suite with 1,000 cities.

costs at all [16]. Estimated backbone information can be used not only to modify distances, but also in different places of a local search to make biased searches. For instance, it can be employed to generate biased initial starting tours. Secondly and more importantly, the backbone guided search does not treat every distance equally; some distances may be altered dramatically while others may remain intact to help exploit the intricate constraints among the cities.

6 Conclusion

We have presented and investigated a new method of applying backbone information to force a local search to make biased local perturbations, in the context of the TSP. Based on the “big valley” hypothesis, we developed a method for deriving heuristic backbone information by exploiting the local minima from the efficient Lin-Kernighan local search algorithm. We demonstrated its effectiveness with extensive experiments on various problem types and instances.

Our results on the TSP show that BGLK and IBGLK compare very favorably to LK and ILK, respectively, on the instances in the Challenge suite, up to 31,623 cities, and the instances from TSPLIB. For the uniform Euclidean class, BGLK and IBGLK outperform LK (and its variants) and ILK, respectively. On clustered Euclidean instances, BGLK outperforms LK by reducing tour costs by more than 1% on average on 31,623-city instances; and IBGLK and ILK have comparable performance. (Note that an improvement of a fraction of percent in solution quality is significant on large TSPs [9].) BGLK and IBGLK dominate their counterparts on tour quality on all instances in the distance matrix class we tested; on 10,000-city instances, BGLK and IBGLK find tours with costs at least 1.5% and 0.5% smaller than LK and ILK, respectively. On real instances from the TSPLIB, IBGLK finds nearly twice as many best solutions as three versions of ILK (i.e., 20 versus 12). Finally, BGLK and IBGLK have comparable running times with their counterparts, no more than twice as long as the original algorithms, while sometimes being faster.

In short, the main contribution of this work is an effective method of utilizing inherent problem features, backbones in particular, of the TSP in the LK algorithms and variants to improve their efficacy. A method of exploiting problem features in search is of fundamental interest and practical importance. The results on the TSP in this paper have indicated that the idea of backbone-guided local search is general and can lead to efficient heuristic algorithms for large, difficult problems. The source code of our algorithm will be available on the web.

References

- [1] K. D. Boese. *Models for Iterative Global Optimization*. PhD thesis, UCLA/Computer Science Department, 1996.
- [2] W. Cook and P. Seymour. Tour merging via branch-decomposition. *INFORMS J. of Computing*, to appear.
- [3] O. Dubois and G. Dequen. A backbone-search heuristic for efficient solving of hard 3-SAT formula. In *Proc. of IJCAI-01*, pages 248–53, 2001.
- [4] J. Gu and X. Huang. Efficient local search with search space smoothing: A case study of the traveling salesman problem (TSP). *IEEE Trans. SMC*, 24:728–735, 1994.
- [5] G. Gutin and A.P. Punnen, editors. *The Traveling Salesman Problem and Its Variations*. Kluwer, 2002.
- [6] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European J. Operations Research*, 126:106–130, 2000.
- [7] T. Hogg, B.A. Huberman, and C. Williams. Phase transitions and the search problem. *Artificial Intelligence*, 81:1–15, 1996.
- [8] D. S. Johnson. 8th dimacs implementation challenge: The TSP. <http://www.research.att.com/~dsj/chtsp>.
- [9] D. S. Johnson and L. A. McGeoch. Experimental analysis of heuristics for the STSP. In *The Traveling Salesman Problem and its Variations*, pages 369–443. Kluwer, 2002.
- [10] S. Lin and B.W. Kernighan. An effective heuristic algorithm for the Traveling Salesman Problem. *Operations Research*, 21:498–516, 1973.
- [11] O. Martin, S. W. Otto, and E. W. Felten. Large-step Markov chains for the traveling salesman problem. *Complex Systems*, 5:299–326, 1991.
- [12] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic ‘phase transitions’. *Nature*, 400:133–137, 1999.
- [13] J.C. Pemberton and W. Zhang. Epsilon-transformation: Exploiting phase transitions to solve combinatorial optimization problems. *Artificial Intelligence*, 81:297–325, 1996.
- [14] G. Reinelt. TSPLIB - a traveling salesman problem library. *ORSA J. Computing*, 3:376–384, 1991.
- [15] B. Selman, H. Kautz, and B. Cohen. Noise strategies for local search. In *Proc. of AAAI-94*, pages 337–43, 1994.
- [16] W. Zhang. Configuration landscape analysis and backbone guided local search: Part I: satisfiability and maximum satisfiability. *Artificial Intelligence*, 158(1):1–26, 2004.

KNOWLEDGE REPRESENTATION AND REASONING

Scalability Study of Peer-to-Peer Consequence Finding

P. Adjiman and P. Chatalic and F. Goasdoué and M.-C. Rousset and L. Simon

PCRI: CNRS & Université Paris-Sud (LRI) – INRIA (Futurs)

Bât. 490, 91405, Université Paris-Sud, Orsay Cedex, France

{adjiman,chatalic,fg,mcr,simon}@lri.fr

Abstract

In peer-to-peer inference systems, each peer can reason locally but also solicit some of its acquaintances, sharing part of its vocabulary. This paper studies both theoretically and experimentally the problem of computing proper prime implicates for propositional peer-to-peer systems, the global theory (union of all peer theories) of which is not known (as opposed to partition-based reasoning).

1 Introduction

Recently peer-to-peer (P2P) systems have received considerable attention because their underlying infrastructure is appropriate to scalable and flexible distributed applications over Internet. In a full P2P system, there is no centralized control or hierarchical organization: each peer is equivalent in functionality and cooperates with other peers in order to solve a collective task. First P2P systems were simple keyword-based file sharing systems like Napster (<http://www.napster.com>) and Gnutella (<http://gnutella.wego.com>), for which efficient lookup services (e.g., CHORD [Stoica *et al.*, 2001]) have been invented. Recently, schema-based peer data management systems like Edutella [Nedj *et al.*, 2002] or Piazza [Halevy *et al.*, 2003b] have been proposed. In those systems, query answering complexity is directly related to the expressivity of the formalism used to state the semantic mappings between peers schemas [Halevy *et al.*, 2003a]. The scalability of Piazza so far goes up to about 80 peers and relies on a wide range of optimizations (mappings composition, paths pruning [Tatarinov and Halevy, 2004]), made possible by the centralized storage of all the schemas and mappings in a global server.

In this paper, we make the choice of being fully distributed: there are neither super-peers (as in Edutella) nor a central server (as in Piazza). In addition, we aim at scaling up to thousands of peers. We consider P2P inference systems in which the local theory of each peer is a set of clauses defined upon a set of propositional variables. Each peer may share part of its vocabulary with some other peers. We investigate the reasoning task of finding consequences of a certain form for a given input formula expressed using the local vocabulary of a peer. This reasoning task is important in many applications (diagnosis, information integration), in which output

must be computed from input that is provided at query time, and cannot be reduced to satisfiability checking.

The problem of distributed reasoning considered in this paper is quite different from the problem of reasoning over partitions obtained by decomposition of a theory ([Dechter and Rish, 1994; Amir and McIlraith, 2000]). In that problem, a centralized large theory is given and its structure is exploited to compute its best partitioning, in order to optimize the use of a partition-based reasoning algorithm. In our problem, the whole theory (i.e., the union of all the local theories) is not known and the partition is imposed by the P2P architecture. Therefore, existing algorithms ([Amir and McIlraith, 2000; Dechter and Rish, 1994; del Val, 1999]) are not appropriate for our consequence finding problem.

Section 2 defines formally the P2P inference problem addressed in this paper. Section 3 describes the distributed consequence finding algorithm that we propose and states its properties. Section 4 accounts for a significant experimental study of the scalability of this approach. Section 5 describes related work and we conclude in Section 6.

2 Peer-to-peer inference: problem definition

A peer-to-peer inference system (P2PIS) is a network of peer theories. Each peer P is a finite set of formulas of a language $\mathcal{L}_P$. We consider the case where $\mathcal{L}_P$ is the language of clauses that can be built from a finite set of propositional variables $\mathcal{V}_P$, called the *vocabulary* of P . Peers can be semantically related by having common *shared variables* in their respective vocabularies. Each peer only knows its own local theory and that it shares some variables with some other peers of the P2PIS (its *acquaintances*). It does not necessarily know *all* the peers with which it shares variables. When a new peer joins a P2PIS it simply declares its acquaintances in the network, i.e., the peers it knows to be sharing variables with. A P2PIS can be formalized as an *acquaintance graph*.

Definition 1 Let $\mathcal{P} = \{P_i\}_{i=1..n}$ be a collection of clausal theories on their respective vocabularies $\mathcal{V}_{P_i}$, let $\mathcal{V} = \cup_{i=1..n} \mathcal{V}_{P_i}$. An *acquaintance graph* over $\mathcal{V}$ is a graph $\Gamma = (\mathcal{P}, \text{ACQ})$ where $\mathcal{P}$ is the set of vertices and $\text{ACQ} \subseteq \mathcal{V} \times \mathcal{P} \times \mathcal{P}$ is a set of labelled edges such that for every $(v, P_i, P_j) \in \text{ACQ}$, $i \neq j$ and $v \in \mathcal{V}_{P_i} \cap \mathcal{V}_{P_j}$.

A labelled edge (v, P_i, P_j) expresses that peers P_i and P_j know each other to be sharing the variable v . For a peer P and a literal l , $\text{ACQ}(l, P)$ denotes the set of peers sharing with P

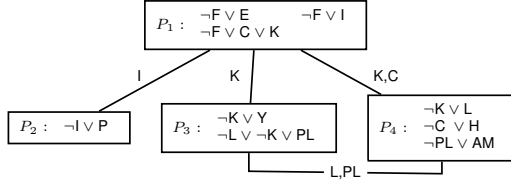


Figure 1: Acquaintance graph for the tour operator example

the variable of l . For each theory P , we consider a subset of *target variables* $TV_P \subseteq \mathcal{V}_P$, supposed to represent the variables of interest for the application. The goal is, given a clause provided as an input to a given peer, to find all the possible consequences of the input clause and the union of the peer theories, that belong to some *target language*. Given a set SP of peers, the target language $Target(SP)$ is the language of clauses (including the empty clause) involving only target variables of peers of SP . A shared variable must have the same target status in all the peers sharing it.

Definition 2 Let P be a clausal theory and q be a clause. A clause m is called a **prime implicate** of q w.r.t. P iff $P \cup \{q\} \models m$ and for any other clause m' , if $P \cup \{q\} \models m'$ and $m' \models m$ then $m' \equiv m$. m is called a **proper prime implicate** of q w.r.t. P iff it is a prime implicate of q w.r.t. P but $P \not\models m$.

Definition 3 Let $\Gamma = (\mathcal{P}, ACQ)$ be an acquaintance graph modeling a P2PIS, where $\mathcal{P} = \{P_i\}_{i=1..n}$ is a collection of clausal theories with respective target variables. The **consequence finding problem** is, given a peer P and a clause $q \in \mathcal{L}_P$ to find the set of proper prime implicates of q w.r.t. $\bigcup_{i=1..n} P_i$ which belong to $Target(\mathcal{P})$.

Since none of the peers of a P2PIS knows the union of the theories of the system, the consequence finding problem in a P2PIS is new and significantly different from the consequence finding problem in a single global theory (even partitioned). In a full P2P setting, each peer must run the same reasoning algorithm locally and independently, while being able to distribute part of the reasoning task that it controls to some of its acquaintances.

Example : Let us consider 4 peers (Figure 1). P_1 , describing a tour operator, expresses that its current far destinations (F) are either Kenya (K) or Chile (C). These far destinations are international destinations (I) and expensive (E). P_2 , only concerned with police regulations, expresses that a passport is required (P) for international destinations. P_3 focuses on sanitary conditions for travelers. It expresses that, in Kenya, yellow fever vaccination (Y) is strongly recommended and that a strong protection against paludism should be taken (PL) when accomodation occurs in Lodges (L). P_4 describes accommodation conditions : Lodges for Kenya and Hotels (H) for Chile. It also expresses that when anti-paludism protection is required, accommodations are equipped with appropriate anti-mosquito protections (AM). Shared variables are indicated on the edges of the acquaintance graph (Figure 1) and target variables are defined by : $TV_{P_1} = \{E\}$, $TV_{P_2} = \{P\}$, $TV_{P_3} = \{L, Y, PL\}$ and $TV_{P_4} = \{L, H, PL, AM\}$.

We now illustrate the behavior of the distributed consequence finding algorithm detailed in Section 3. When a peer receives a query, it first computes all local consequents of the

query. Those in $Target(P)$ are immediately returned. Then, those made of shared literals are splitted. For each shared literal, a subquery is propagated to the neighbor peers sharing the corresponding variable. When returned, consequents of the subqueries are respectively queued for future recombination. As soon as one answer has been returned for each subquery, they are recombined and transmitted back as new consequents to the querying peer. This process continues incrementally, as further consequents for the subqueries are returned.

For instance, suppose that F is transmitted to peer P_1 by the user. The consequents that are locally computed are E , I and $C \vee K$. Since $E \in Target(P_1)$, it is immediately returned as a *local consequent*. Since I is shared with P_2 , it is transmitted to P_2 , which produces the clause P . Since $P \in Target(P_2)$, it is transmitted back to P_1 and returned as a *remote consequent* of the initial query. The clause $C \vee K$, being made of shared variables, is splitted and C and K are transmitted separately to the concerned neighbors. C is transmitted to P_4 , which returns (only) H to P_1 , where it is queued for combination. Similarly, K is transmitted (independently) to P_4 and P_3 (both share K with P_1). On P_4 , L is produced locally. Since $L \in Target(P_4)$ it is returned as a first consequent of K to P_1 , where it is queued. On P_1 , after recombination, $H \vee L$ is then returned as a first *combined consequents* of the initial query. Since L is also shared between P_4 and P_3 , it is propagated on P_3 , where the clause $\neg K \vee PL$ is produced and, in turn, splitted. P_4 is then asked for PL and returns AM as its only consequent. P_1 is asked for $\neg K$. This happens while the complementary query K is still under process. We will see in Section 3 that when a same reasoning branch contains two complementary literals (which is detected using a history mechanism), it is closed and $\square$ is returned as a consequent. P_3 now combines AM (returned by P_4 for PL), and $\square$ (returned by P_1 for $\neg K$) as a new consequent of $\neg K \vee PL$, and thus, transmits AM back to P_4 as a new consequent of L . For lack of space, we do not detail all reasoning branches. The set of consequents of the initial query eventually produced is: $\{E, I, H \vee L, H \vee AM, H \vee Y, H \vee PL\}$. Among those answers, let us note that some of them (e.g., $H \vee Y$) involve target variables from different peers. Such implicates cannot be obtained by partition-based algorithms like in [Amir and McIlraith, 2000] which only compute consequents that belong to the target language of a single peer.

3 Distributed consequence finding algorithm

Our distributed and anytime consequence finding algorithm has been presented in [Adjiman *et al.*, 2004b]. For this paper to be self-contained, we describe the three main message passing procedures, which are implemented locally at each peer. They are triggered by the reception of a *query* (resp. *answer*, *final*) message, sent by a *Sender* peer to a receiver peer, denoted by *Self*, which executes the procedure.

Those procedures handle an *history* which is initialized to the empty sequence. An history *hist* is a sequence of triples (l, P, c) (where l is a literal, P a peer, and c a clause). An history $[(l_n, P_n, c_n), \dots, (l_1, P_1, c_1), (l_0, P_0, c_0)]$ represents a branch of reasoning initiated by the propagation of the lit-

eral l_0 within the peer P_0 , and the splitting of the clause c_0 : for every $i \in [0..n-1]$, c_i is a consequence of l_i and P_i , and l_{i+1} is a literal of c_i , which is propagated in P_{i+1} .

The reasoning is initiated by the user (denoted by a particular peer $User$) sending to a given peer P a message $m(User, P, query, \emptyset, q)$, which triggers the procedure $RECEIVEQUERYMESSAGE(m(User, P, query, \emptyset, q))$ that is locally executed by P . In the description of the procedures, we will use the notations:

- for a literal q , $\bar{q}$ denotes its complementary literal and $Resolvent(q, P)$ denotes the set of clauses obtained by resolution between q and a clause of P ,

- for a clause c of a peer P , $S(c)$ (resp. $L(c)$) denotes the disjunction of literals of c whose variables are shared (resp. not shared) with any acquaintance of P . $S(c) = \square$ thus expresses that c does not contain any shared variable,

- $\odot$ is the distribution operator on sets of clauses: $S_1 \odot \dots \odot S_n = \{c_1 \vee \dots \vee c_n \mid c_1 \in S_1, \dots, c_n \in S_n\}$. If $L = \{l_1, \dots, l_p\}$, we use $\odot_{l \in L} S_l$ to denote $S_{l_1} \odot \dots \odot S_{l_p}$.

The main properties of the resulting distributed message passing algorithm, stated in [Adjiman *et al.*, 2004a], can be summarized as follows:

- 1) The algorithm is sound: the answers that are returned are all implicates of the literal having triggered the reasoning.
- 2) The algorithm terminates and the user is notified of the termination, which is crucial for an anytime algorithm.
- 3) The completeness of the algorithm is guaranteed if each local theory is saturated by resolution and if the acquaintance graph is such that if two local theories have a common variable, there exists in the acquaintance graph a path between those two theories, all the edges of which are labeled with that variable. If that property is not satisfied, the algorithm still applies but does not guarantee to return all the proper prime implicates when it terminates.

Note that $\square$ can be returned by our algorithm (lines 1 to 3 and 8 to 10 in $RECEIVEQUERYMESSAGE$). Therefore, our algorithm can be exploited for checking the satisfiability of the global theory at each join of a new peer. For the sake of simplicity, our algorithm is presented as applying to literals. However any clause can be handled by splitting it into literals and recombining the results obtained for each literal using the $\odot$ operator.

4 Experimental analysis

In order to study scalability issues of our P2P algorithm we have conducted a significant experimentation on networks composed of 1000 peers. To the best of our knowledge, this is the first experimental study on such large P2PIS. Our motivation was twofold. First, to study how deep and how wide reasoning spreads on the network. Second, to evaluate the time needed to obtain answers and to check to what extent the P2PIS is able to support the traffic load.

Since we want to use our infrastructure in real Web applications, we have chosen to generate acquaintance graphs having the so-called *small world* property, which is admitted [Newman, 2000] as being a general property of social networks (including the Web). Following [Watts and Strogatz, 1998], we start from a regular ring of 1000 nodes, 10 edges

Algorithm 1: Procedure handling queries. It takes care of the propagation of the literal q by $Self$.

```

RECEIVEQUERYMESSAGE( $m(Sender, Self, query, hist, q)$ )
(1) if  $(\bar{q}, -, -) \in hist$ 
(2) send  $m(Self, Sender, answer, [(q, Self, \square)|hist], \square)$ 
(3) send  $m(Self, Sender, final, [(q, Self, true)|hist], true)$ 
(4) else if  $q \in Self$  or  $(q, Self, -) \in hist$ 
(5) send  $m(Self, Sender, final, [(q, Self, true)|hist], true)$ 
(6) else
(7) LOCAL( $Self$ )  $\leftarrow \{q\} \cup Resolvent(q, Self)$ 
(8) if  $\square \in LOCAL(Self)$ 
(9) send  $m(Self, Sender, answer, [(q, Self, \square)|hist], \square)$ 
(10) send  $m(Self, Sender, final, [(q, Self, true)|hist], true)$ 
(11) else
(12) LOCAL( $Self$ )  $\leftarrow \{c \in LOCAL(Self) \mid L(c) \in Target(Self)\}$ 
(13) if for every  $c \in LOCAL(Self)$ ,  $S(c) = \square$ 
(14) foreach  $c \in LOCAL(Self)$ 
(15) send  $m(Self, Sender, answer, [(q, Self, c)|hist], c)$ 
(16) send  $m(Self, Sender, final, [(q, Self, true)|hist], true)$ 
(17) else
(18) foreach  $c \in LOCAL(Self)$ 
(19) if  $S(c) = \square$ 
(20) send  $m(Self, Sender, answer, [(q, Self, c)|hist], c)$ 
(21) else
(22) foreach literal  $l \in S(c)$ 
(23) if  $l \in Target(Self)$ 
(24) ANSWER( $l, [(q, Self, c)|hist]$ )  $\leftarrow \{l\}$ 
(25) else
(26) ANSWER( $l, [(q, Self, c)|hist]$ )  $\leftarrow \emptyset$ 
(27) FINAL( $l, [(q, Self, c)|hist]$ )  $\leftarrow false$ 
(28) foreach  $RP \in ACQ(l, Self)$ 
(29) send  $m(Self, RP, query, [(q, Self, c)|hist], l)$ 

```

Algorithm 2: Procedure handling answers. r is returned as a consequent of the last literal added to the history $hist$.

```

RECEIVEANSWERMESSAGE( $m(Sender, Self, answer, hist, r)$ )
(1)  $hist$  is of the form  $[(l', Sender, c'), (q, Self, c)|hist']$ 
(2) ANSWER( $l', hist$ )  $\leftarrow ANSWER(l', hist) \cup \{r\}$ 
(3) RESULT  $\leftarrow \odot_{l \in S(c) \setminus \{l'\}} ANSWER(l, hist) \odot \{L(c) \vee r\}$ 
(4) if  $hist' = \emptyset$ ,  $U \leftarrow User$  else  $U \leftarrow$  the first peer  $P'$  of  $hist'$ 
(5) foreach  $cs \in RESULT$ 
(6) send  $m(Self, U, answer, [(q, Self, c)|hist'], cs)$ 

```

Algorithm 3: Procedure handling notifications: answer computation for the last literal added to $hist$ is completed.

```

RECEIVEFINALMESSAGE( $m(Sender, Self, final, hist, true)$ )
(1)  $hist$  is of the form  $[(l', Sender, true), (q, Self, c)|hist']$ 
(2) FINAL( $l', hist$ )  $\leftarrow true$ 
(3) if for every  $l \in S(c)$ , FINAL( $l, hist$ ) = true
(4) if  $hist' = \emptyset$   $U \leftarrow User$  else  $U \leftarrow$  the first peer  $P'$  of  $hist'$ 
(5) send  $m(Self, U, final, [(q, Self, true)|hist'], true)$ 
(6) foreach  $l \in S(c)$ 
(7) ANSWER( $l, [(l, Sender, -), (q, Self, c)|hist']$ )  $\leftarrow \emptyset$ 

```

per node, and rewire each node with a given probability pr . It has been shown that between regular graphs ($pr = 0$) and uniform random graphs ($pr = 1$), the graphs generated with $pr = 0.1$ have "small world" properties.

In the following experiments, the number q of shared variables labelling each edge varies, and each of the 1000 local theories is a 2+p clausal theory composed of clauses of length 2 and a varying ratio p of clauses of length 3. Each local theory is generated in two steps. First, 70 clauses of length 2 are

uniformly generated over 70 variables that are proper to the local theory, among which 40 are chosen as target variables. Then, we add clauses, involving shared variables, of length 2 or 3. We denote $\%3cnf$ the percentage of clauses of length 3 to generate in the whole set of mapping clauses. Each mapping clause is randomly generated by picking a variable in each of the two peers and by negating it with probability 0.5 (if the clause is of length 3, the third variable is chosen at random between the variables of the two peers).

The experiments have been conducted on two different platforms. For the measurements concerning the behavior of query processing (number of messages, depth and width of each query processing) we have used a single computer running 1000 peers. Such measurements, consisting in building reports on all peer traces are easier to perform when all data are available on a single computer. In contrast, when time was part of the measurement, we deployed our algorithm on a cluster of 75 heterogeneous computers (Athlons with 1GB of RAM: 26 at 1.4GHZ, 9 at 1.8Ghz and 40 at 2 GHZ). In these last experiments, each computer was running around 14 peers, randomly selected.

As we will see, results often exhibit an *exponential* distribution: some queries may need a very long time to complete. It was thus not always possible to perform our experimental analysis without introducing a *timeout* parameter. Each query is labeled with its remaining time to live, which is decreased each time a query needs to traverse a peer to be processed. When necessary, the *timeout* has been set to 30 seconds.

We first report on the distributed behavior of query processing by measuring the number of peers that are involved in query processing. Then, we report on the time of query processing and on the number of answers.

4.1 Distributed behavior of query processing

We have measured the distribution of the depth of query processing as well as the potential width of a query. The *depth* of a query is the maximum length of the reasoning branches developed by the distributed algorithm for returning an answer. The width of the query estimates the number of neighbors that are solicited, on average, for processing a query.

Figure 2 shows the cumulative distribution function of the depth of 1000 queries, when we make vary the number q of shared variables per edge and the ratio $\%3cnf$ of mapping clauses of length 3. Each point on the figure reports a run, for a distinct query.

The four top curves show a relatively small query depth. For instance, with $q = 2$ and $\%3cnf = 0$ no query depth is greater than 7, and none of those four curves have a query depth greater than 36, which suggests that our algorithm behaves well on such networks.

As soon as the value of $\%3cnf$ increases queries have longer depths: with $q = 3$ and $\%3cnf = 20$, 22% of the queries have a depth greater than 100 (the maximum being 134). If we focus on the last three curves on the right, a sharp threshold clearly appears, showing three phases: a sharp growth, representing query processing with small depth, followed by a plateau, and then a slower growth. The small depth query processing and the 'plateau' are characteristics of an exponential distribution of values: most of the process-

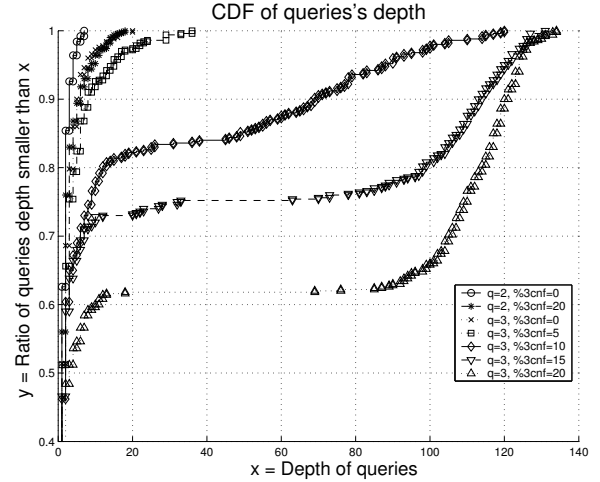


Figure 2: Cumulative Distribution Function of queries depth. The Y scale is re-centered on $[0.4 - 1]$.

ing is easy, but the little remaining is very hard. The slow growth observed is due to the timeout, a side-effect of which is to bound the query depth. Without such a timeout, previous experiments suggest that there would exist some queries requiring very long reasoning branches. This point is outlined on the curve corresponding to the hardest cases ($q = 3$ and $\%3cnf = 20$) where there is no query of depth between 20 and 60. This suggests that when hard processing appears, it is *very hard*. In experiments that are not reported here, we have seen that such an exponential distribution of values was not observed when the acquaintance graphs have a regular ring structure, but was observed on random graphs. We have also measured the *integration degree* of queries, which is the number of *distinct* peers involved in the query processing. We have observed the same kind of exponential distribution of values than for the depth, but with 20% smaller values: 1/5 of the peers are repeated in the histories. That phenomenon was not observed on random acquaintance graphs and seems closely related to the small-world topology.

Figure 3 shows the cumulative distribution function for the *width* of the queries. Each curve summarizes 20000 runs. With $q = 2$ and $\%3cnf = 0$, more than 75% of the queries are solved locally and 15% other are solved by asking just one neighbor. With $q = 5$ and $\%3cnf = 100$, 25% of the queries solicit 10 neighbors on average, each of them soliciting 10 peers, with 25% chance and so on. Such result explains the combinatorial explosion observed on hard instances.

Our experiments have pointed out a direct impact of the $\%3cnf$ value on query processing, which is not surprising considering the hardness of clauses of length 3 for prime implicate computation. Those experiments also suggest an exponential distribution of query depths, due to the short path length between two peers in the acquaintance graphs, and with an important repetition of solicited peers, due to the large clustering coefficient of small world acquaintance graphs.

4.2 Time and number of answers

We now report a time performance study of our algorithm when it is deployed on a real cluster of 75 heterogeneous

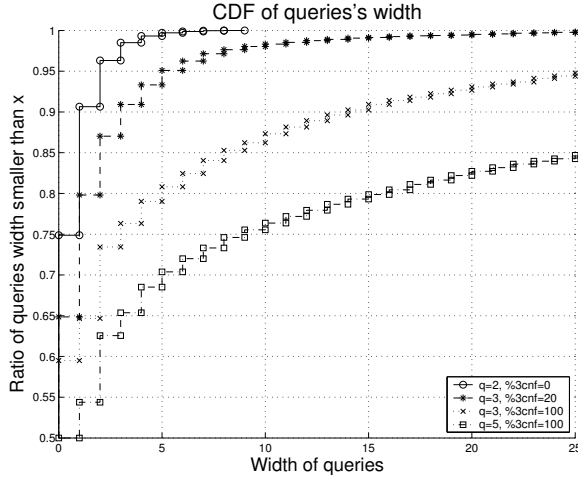


Figure 3: Cumulative Distribution Function of queries width. The Y scale is re-centered on $[0.5 - 1]$, the X axis on $[0 - 25]$.

computers. Based on the previous observations, we have chosen to focus on 4 different kinds of acquaintance graphs, denoted *Easy*, *Medium*, *Hard* and *Very Hard* (see Table 1).

	Network			
	Easy $q = 2$ $\%3cnf = 0$	Medium $q = 3$ $\%3cnf = 20$	Hard $q = 3$ $\%3cnf = 100$	Very Hard $q = 5$ $\%3cnf = 100$
1 st ans.	0.04s (100%)	1.26s (99.6%)	1.58s (95.6%)	1.39s (89.3%)
10 th ans.	0.06s (14.3%)	1.37s (25.6%)	0.99s (33.3%)	1.13s (12.0%)
100 th ans.	–	2.11s (12.7%)	0.84s (27.0%)	4.09s (10.7%)
1000 th ans.	–	4.17s (6.80%)	4.59s (21.2%)	11.35s (7.15%)
all	0.07s	5.56s	14.6s	21.23s
% timeout	0	13.96	37.5	66.9
#answers	5.17	364	1006	1004
%unsat	4.4	3.64	3.76	1.84

Table 1: Characteristics of the query processing on easy to very hard networks.

The values reported in the Table 1 are mean values over more than 300 different queries. For each column, we show the time needed to produce the 1st, 10th, 100th and 1000th answers of a query. Each mean time is followed by the percentage of initial queries that are taken into account in the averaging. For instance, for the Medium case, 12.7% of the queries have produced more than 100 answers, and the 100th answer was given on average after 2.11 seconds (the average does not take into account queries that did not produce at least 100 answers). The *all* row is the mean time needed to produce all answers, including queries that lead to timeout, the percentage of which is reported in the *%timeout* row. The last two rows report the mean number of answers and the ratio of proven unsatisfiable queries w.r.t. the network.

It is not surprising to find that there is no timeout for the *Easy* case. It is known [Marquis, 2000] that satisfiability checking and prime implicates computation are tractable for sets of clauses of length 2. Moreover, the high partitioning of the global theory induced by the low value of q (number of shared variables) is often a witness of “easy” cases for reasoning for centralized theories. The point to outline for this case

is that there are on average 5 answers, and they are produced very quickly.

Even on hard and very hard instances, our algorithm produces a lot of answers coming from a number of different peers. For instance, we measured on average 1006 answers for *Hard*, and 1004 answers for *Very Hard* problems, which already represents a large amount of data. In addition, on those *Very Hard* instances, 90% of runs produced at least one answer. It is noticeable that such hard instance may also be hard for checking the satisfiability of an equivalent, centralized, theory. The formula corresponding to the centralized version of all the distributed theories has $n=70\,000$ variables and $m=120\,000$ clauses 50 000 of which are of length 3. It has been shown in [Monasson *et al.*, 1999] that, for such 2+p formulas, if one does not restrict the locality of variables, the SAT/UNSAT transition is continuous for $p < p_0$, where $p_0 = 0.41$, and discontinuous for $p > p_0$, like in 3-SAT instances. Intuitively, for $p > p_0$, the random 2+p-SAT problem shares the characteristics of the random 3-SAT problems. Let us emphasize here that, with the characteristics of our *Very Hard* network, we have $p = 0.416$ for which the transition phase between SAT and UNSAT instance [Monasson *et al.*, 1999] occurs at $m/n=1.69$. Here, we have $m/n=1.71$, which is near the transition phase to confirm that this is where the hard instances would be in practice. Of course, such a comparison is only indicative, because there is no variable locality restriction in the standard 2+p model.

To summarize, when deployed on a real cluster of heterogeneous computers, our algorithm scales very well. Even on very hard instances that shares characteristics of a very large 2+p formula at the crossover between the 2-SAT/3-SAT and the SAT/UNSAT transitions, our algorithm was able to generate a large number of answers in a reasonable time, for a majority of runs.

5 Related work

The distributed message passing algorithm that we have described in Section 3 can be viewed as a distributed version of an Ordered Linear deduction [Chang and Lee, 1973] to produce new target clauses, which was extended by [Siegel, 1987] in order to produce all implicates of a given clause belonging to some target language, and further extended to the first order case in [Inoue, 1992]. The problem of computing new derived clauses (a.k.a. $\langle L, \Phi \rangle$ -prime implicates) corresponds exactly to the problem of computing proper prime implicates w.r.t. a theory. It has been extensively studied in the centralized case (see [Marquis, 2000] for a survey).

We have already pointed out the differences between our work and [Amir and McIlraith, 2000]. First, in a full peer-to-peer setting, tree decomposition of the acquaintance graph is not possible. Second, in contrast with partition-based algorithms as in [Amir and McIlraith, 2000], our algorithm is able to *combine* answers from different peers in order to compute implicates involving target variables of different peers.

The model-based diagnosis algorithm for distributed embedded systems [Provan, 2002] exploits the knowledge on the distribution of the system to diagnose for optimization purpose. In distributed ATMS [Mason and Johnson, 1989],

agents exchange nogood sets in order to converge to a globally consistent set of justifications. Such a distributed vision of ATMS relies on a global knowledge shared by all the agents and aims at getting a unique global solution. We think that our peer-to-peer inference system can be applied for handling fully distributed model-based diagnosis and fully distributed ATMS, in which no global knowledge is required.

Some recent work deals with distributed first order logic. A model-based and a proof-theoretic semantics has been defined in [Ghidini and Serafini, 1998] for a collection of first order theories communicating through bridge rules that define semantic mappings between their respective domains of interpretation. Based on that work, distributed description logics has been introduced in [Borgida and Serafini, 2003] and a distributed tableau method has been proposed for satisfiability checking.

6 Conclusion

We have presented a peer-to-peer inference system based on propositional logic and we have shown that it scales up to a thousand of peers. The peer-to-peer infrastructure that we have developed is used in a joint project with France Télécom, aiming at enriching web applications with semantics and reasoning services.

So far, we have restricted our algorithm to deal with a vocabulary-based target language. However, it can be adapted to more sophisticated target languages (implicates of a given language, e.g., based on literals and not only variables, of bounded size,...). This can be done by adding a simple tag over all messages to encode the desired target language.

We plan to extend our current work in two main directions. First, we want to tackle Semantic Web applications, in which the bottleneck is to deal with distributed resources shared at large scale. RDF(S) (see [Antoniou and van Harmelen, 2004]) is a standard for annotating web resources, which we think can be encoded in our propositional setting. Second, we want to handle more sophisticated reasoning in order to deal with a real multi-agent setting, in which possible inconsistencies between agents must be handled.

References

- [Adjiman *et al.*, 2004a] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting. Technical report, <http://www.lri.fr/~goasdoue/bib/ACGRS-TR-1385.pdf>, 2004.
- [Adjiman *et al.*, 2004b] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting, short paper. In *ECAI*, 2004.
- [Amir and McIlraith, 2000] E. Amir and S. McIlraith. Partition-based logical reasoning. In *KR*, 2000.
- [Antoniou and van Harmelen, 2004] G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. The MIT Press, 2004.
- [Borgida and Serafini, 2003] A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1, 2003.
- [Chang and Lee, 1973] C. L. Chang and R. C. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Computer Science Classics, Academic Press, 1973.
- [Dechter and Rish, 1994] R. Dechter and I. Rish. Directed resolution: the davis-putnam procedure revisited. In *KR*, 1994.
- [del Val, 1999] A. del Val. A new method for consequence finding and compilation in restricted languages. In *AAAI*, 1999.
- [Ghidini and Serafini, 1998] C. Ghidini and L. Serafini. *Distributed First Order Logics*. Research studies Press, 1998.
- [Halevy *et al.*, 2003a] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *ICDE*, 2003.
- [Halevy *et al.*, 2003b] A. Halevy, Z. Ives, I. Tatarinov, and Peter Mork. Piazza: data management infrastructure for semantic web applications. In *WWW*, 2003.
- [Inoue, 1992] K. Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, (56), 1992.
- [Marquis, 2000] P. Marquis. *Handbook on Defeasible Reasoning and Uncertainty Management Systems, Algorithms for Defeasible and Uncertain Reasoning*, volume 5, chapter Consequence Finding Algorithms, pages 41–145. Kluwer Academic Publisher, 2000.
- [Mason and Johnson, 1989] C.L. Mason and R.R. Johnson. *Distributed Artificial Intelligence II*, chapter DATMS: a framework for distributed assumption based reasoning. Pitman, 1989.
- [Monasson *et al.*, 1999] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. 2+p-sat: Relation of typical-case complexity to the nature of the phase transition. *Random Structure and Algorithms*, 15(414), 1999.
- [Nedjl *et al.*, 2002] W. Nedjl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. Edutella: a p2p networking infrastructure based on rdf. In *WWW*, 2002.
- [Newman, 2000] M. E. J. Newman. Models of the small world. *J. Stat. Phys.*, 101:819–841, 2000.
- [Provan, 2002] G. Provan. A model-based diagnosis framework for distributed embedded systems. In *KR*, 2002.
- [Siegel, 1987] P. Siegel. *Représentation et utilisation de la connaissance en calcul propositionnel*. PhD thesis, Université d’Aix-Marseille II, 1987.
- [Stoica *et al.*, 2001] I. Stoica, R. Morris, D. Karger, M.F. Kaasshoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for internet applications. In *Conference on applications, technologies, architecture and protocols for computer communications*, 2001.
- [Tatarinov and Halevy, 2004] I. Tatarinov and A. Halevy. Efficient query reformulation in peer data management systems. In *SIGMOD 04*, 2004.
- [Watts and Strogatz, 1998] D. J. Watts and S. H. Strogatz. Models of the small world. *Nature*, 393:440–442, June 1998.

Identifiability of Path-Specific Effects*

Chen Avin, Ilya Shpitser, Judea Pearl

Cognitive Systems Laboratory
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA. 90095
{avin, ilyas, judea}@cs.ucla.edu

Abstract

Counterfactual quantities representing path-specific effects arise in cases where we are interested in computing the effect of one variable on another only along certain causal paths in the graph (in other words by excluding a set of edges from consideration). A recent paper [Pearl, 2001] details a method by which such an exclusion can be specified formally by fixing the value of the parent node of each excluded edge. In this paper we derive simple, graphical conditions for experimental identifiability of path-specific effects, namely, conditions under which path-specific effects can be estimated consistently from data obtained from controlled experiments.

1 Introduction

Total, direct and indirect effects are important quantities in practical causal reasoning about legal, medical, and public policy domains, among others. The task of explicating, and computing these quantities has been successfully addressed in the framework of linear structural equation models (SEM), but encountered difficulties in non-linear as well as non-parametric models. See for instance [Robins and Greenland, 1992], [Galles and Pearl, 1995], [Pearl, 2001],

In the linear SEM framework, the *total effect* of Z on Y is the response of Y to a unit change in the setting of Z . On the other hand, the *direct effect* is the effect of Z on Y not mediated by any other variable in the model while the *indirect effect* is the effect of Z on Y excluding the direct effect.

In non-parametric models, we can define the *controlled direct effect* as the change in the measured response of Y to a change in Z , while all other variables in the model, henceforth called *context variables*, are held constant. Unfortunately, there is no way to construct an equivalent notion of controlled indirect effects, since it is not clear to what values other variables in the model need to be fixed in order to measure such an effect.

Recently, a novel formulation of *natural* [Pearl, 2001] or *pure* [Robins and Greenland, 1992] effects was proposed

*This research was partially supported by AFOSR grant #F49620-01-1-0055, NSF grant #IIS-0097082, and ONR (MURI) grant #N00014-00-1-0617.

which defined effects in a more refined way by holding variables constant not to predetermined values, but to values they would have attained in some situation. For example, the natural direct effect of Z on Y is the sensitivity of Y to changes in Z , while the context variables are held fixed to the values they would have attained had no change in Z taken place. Similarly, the natural indirect effect is the sensitivity of Y to changes the context variables would have undergone had Z been changed, while Z is actually being fixed.

Being complex counterfactual quantities, natural effects tend to have intricate verbal descriptions. It is often easier to explain such effects using the visual intuitions provided by graphical causal models. Graphical causal models represent causal assumptions as graphs, with vertices representing variables, and edges representing direct causal paths. In such models, natural direct effect can be interpreted as the effect along the edge $Z \rightarrow Y$, with the effect along all other edges 'turned off.' Similarly, the natural indirect effect can be interpreted as the effect along all edges except the one between Z and Y . Using this interpretation, the suggestive next step in the study of natural effects is to consider effects along a select subset of edges between Z and Y which are called *path-specific effects*.

1.1 A Motivating Example

Consider the following example, inspired by [Robins, 1997], A study is performed on the effects of the AZT drug on AIDS patients. AZT is a harsh drug known to cause a variety of complications. For the purposes of the model, we restrict our attention to two – pneumonia and severe headaches. In turn, pneumonia can be treated with antibiotics, and severe headache sufferers can take painkillers. Ultimately, all the above variables, except headache, are assumed to have a direct effect on the survival chances of the patient. The graphical causal model for this situation is shown in Fig. 1.

The original question considered in this model was the total effect of AZT and antibiotics treatment on survival. However, a variety of other questions of interest can be phrased in terms of natural effects. For instance, what is the direct effect of AZT on survival, if AZT produced no side effects in the patient, which is just the natural direct effect of AZT on survival. See Fig. 2 (a). Similarly, we might be interested in how just the side effects of AZT affect survival, independent of the effect of AZT itself. This corresponds to the natural

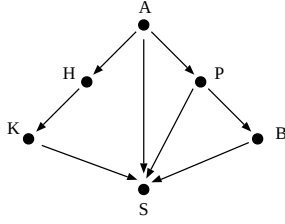


Figure 1: The AZT example. *A*: AZT, *P*: pneumonia, *H*: headaches, *B*: antibiotics, *K*: painkillers, *S*: survival

indirect effect of AZT on survival. See Fig. 2 (b).

Furthermore, certain interesting questions cannot be phrased in terms of either direct or indirect natural effects. For example we might be interested in the interactions between antibiotics and AZT that negatively affect survival. To study such interactions, we might consider the effect of administering AZT on survival in the idealized situation where the antibiotics variable behaved as if AZT was *not* administered, and compare this to the total effect of AZT on survival. Graphically, this amounts to 'blocking' the direct edge between antibiotics and survival or more precisely, keeping the edge functioning at the level it would have had no AZT been given, while letting the rest of the edges function as usual. This is shown graphically in Fig. 3 (a). The edges which we wish to block will be crossed out in the graph.

1.2 Outline and Discussion of Our Approach

Our goal is to study and characterize situations where path-specific effects like the one from the previous section can be computed uniquely from the data available to the investigator. Our main result is a simple, necessary, graphical condition for the identifiability of path-specific effects from experimental data. Furthermore, our condition becomes sufficient for models with no spurious correlations between observables, also known as Markovian models.

The condition can be easily described in terms of blocked and unblocked paths as follows. Let X, Y be variables in a causal model M inducing a graph G . Then given a set of blocked edges g , the corresponding path-specific effect of X on Y cannot be identified if and only if there exists a node W with an unblocked directed path from X to W , an unblocked directed path from W to Y , and a blocked directed path from W to Y . For instance, the effects of A on S are identifiable in Fig. 2 (a), (b), and Fig. 3 (b), but not in Fig. 3 (a). Therefore, in general we cannot study the interactions of AZT and antibiotics in the way described above, but we can study the interactions of AZT and painkillers. The latter case is made tractable by an absence of blocked and unblocked paths sharing edges.

Our condition also shows that all identifiable path-specific effects are 'equivalent', in a sense made precise later, to effects where only root-emanating edges are blocked. Thus identifiable path-specific effects are a generalization of both natural direct effects, where a single root-emanating edge is unblocked, and of natural indirect effects, where a single root-emanating edge is blocked.

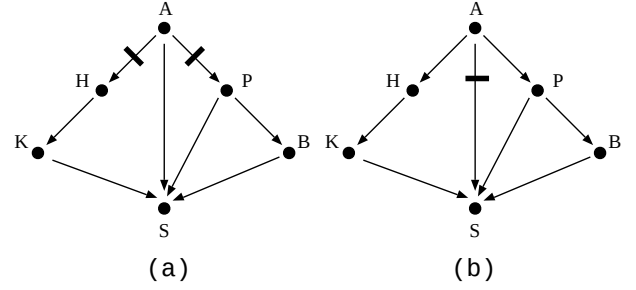


Figure 2: (a) Natural direct effect (b) Natural indirect effect

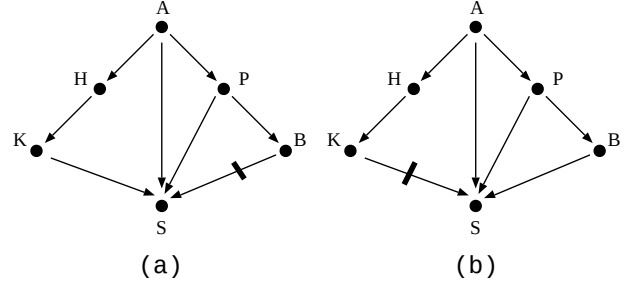


Figure 3: Path specific effects

To obtain this result formally, we treat effects as probabilities of statements in a certain counterfactual logic. However, rather than manipulating these probabilities directly, we convert them to subgraphs of the original causal model, and reason about and perform manipulations on the subgraphs. We then introduce simple counterfactual formulas whose probabilities are not identifiable, and prove that certain simple graphical conditions must be described by such formulas, and lack of such conditions leads to subgraphs corresponding to identifiable effects.

Due to space considerations, the proofs of some lemmas have been omitted, while the proofs included generally are missing some technical details. Our technical report contains the complete proofs.

2 Preliminaries

This paper deals extensively with causal models and counterfactuals. We reproduce their definitions here for completeness. A full discussion can be found in [Pearl, 2000]. For the remainder of the paper, variables will be denoted by capital letters, and their values by small letters. Similarly, sets of variables will be denoted by bold capital letters, sets of values by bold small letters. We will also make use of some graph theoretic abbreviations. We will write $Pa(A)_G$, $De(A)_G$, and $An(A)_G$, to mean the set of parents, descendants (inclusive), and ancestors (inclusive) of node A in graph G . G will be omitted from the subscript when assumed or obvious. If a variable is indexed, i.e. V^i , we will sometimes denote the above sets as Pa^i , De^i , and An^i , respectively.

2.1 Causal Models and Counterfactual Logic

Definition 1 A probabilistic causal model (PCM) is a tuple $M = \langle U, V, F, P(u) \rangle$, where

- (i) U is a set of background or exogenous variables, which cannot be observed or experimented on, but which can influence the rest of the mode
- (ii) V is a set $\{V^1, \dots, V^n\}$ of observable or endogenous variables. These variables are considered to be functionally dependent on some subset of $U \cup V$.
- (iii) F is a set of functions $\{f^1, \dots, f^n\}$ such that each f^i is a mapping from a subset of $U \cup V \setminus \{V^i\}$ to V^i , and such that $\bigcup F$ is a function from U to V .
- (iv) $P(u)$ is a joint probability distribution over the variables in U .

A causal model M induces a directed graph G , where each variable corresponds to a vertex in G and the directed edges are from the variables in the domain of f^i (i.e. Pa^i) to V^i for all the functions. For the remainder of this paper, we consider causal models which induce directed acyclic graphs.

A Markovian causal model M has the property that each exogenous variable U is in the domain of at most one function f . A causal model which does not obey this property is called semi-Markovian. By convention, nodes corresponding to variables in U are not shown in graphs corresponding to Markovian models.

For the purposes of this paper, we will represent counterfactual statements in a kind of propositional modal logic, similar to the one used in [Halpern, 2000]. Furthermore, the distribution $P(u)$ will induce an additional probabilistic interpretation on the statements in the logic.

Definition 2 (atomic counterfactual formula) Let M be a causal model, let X be a variable and Z be a (possibly empty) set of variables. Then for any value x of X , and values z of Z , x is a term, and $X_z(u)$ is a term, taken to mean 'the value X attains when Z is forced to take on values z , and U attain values u .'

For two terms t_1 and t_2 , an atomic counterfactual formula has the form $t_1 = t_2$. We will abbreviate formulas of the form $X_z(u) = x$ as $x_z(u)$.

The 'forcing' of the variables to z is called an intervention, and is denoted by $do(z)$ in [Pearl, 2000]. Counterfactual formulas are constructed from atomic formulas using conjunction and negation.

Definition 3 (counterfactual formula)

- (i) An atomic formula $\alpha(u)$ is a counterfactual formula.
- (ii) If $\alpha(u)$ is a counterfactual formula, then so is $(\neg\alpha)(u)$.
- (iii) If $\alpha(u)$ and $\beta(u)$ are counterfactual formulas, then so is $(\alpha \wedge \beta)(u)$.

The satisfaction of counterfactual formulas by causal models is defined in the standard way, which we reproduce from [Halpern, 2000].

Definition 4 (entailment) A causal model M satisfies a counterfactual formula $\alpha(u)$, written $M \models \alpha(u)$, if all variables appearing in α are in M and one of the following is true

- (i) $\alpha(u) \equiv t_1 = t_2$ and for the given setting of u , the terms t_1 and t_2 are equal in M .
- (ii) $\alpha(u) \equiv (\neg\beta)(u)$ and $M \not\models \beta(u)$.
- (iii) $\alpha(u) \equiv (\beta \wedge \gamma)(u)$ and $M \models \beta(u)$ and $M \models \gamma(u)$

Thus a formula $\alpha(u)$ has a definite truth value in M . If the values u are unknown, we cannot in general determine the truth of α . However, we can easily define a natural notion of probability of α in M as follows:

$$P(\alpha|M) = \sum_{\{u|M \models \alpha(u)\}} P(u) \quad (1)$$

We will omit the conditioning on M if the model in question is assumed or obvious.

If we consider each value assignment u as a *possible world*, then we can view $P(u)$ as describing our degree of belief that a particular world is true, and $P(\alpha)$ as our belief that a particular statement is true in our causal model if viewed as a *type 2 probability structure* [Halpern, 1990].

2.2 Submodels and Identifiability

Definition 5 (submodel) For a causal model $M = \langle U, V, F, P(u) \rangle$, an intervention $do(z)$ produces a new causal model $M_z = \langle U, V_z, F_z, P(u) \rangle$, where V_z is a set of distinct copies of variables in V , and F_z is obtained by taking distinct copies of functions in F , but replacing all copies of functions which determine the variables in Z by constant functions setting the variables to values z .

The joint distribution $P(V_z)$ over the endogenous variables in M_z is called an interventional distribution, and is sometimes denoted as P_z . For a given causal model M , define P_* as $\{P_z | Z \subseteq V, z \text{ a value assignment of } Z\}$. In other words, P_* is the set of all possible interventional (or experimental) distributions of M .

Intuitively, the submodel is the original causal model, minimally altered to render Z equal to z , while preserving the rest of its probabilistic structure.

Because there is no requirement that interventions in atomic counterfactuals in a formula α be consistent with each other, it is in general impossible to alter the original model using only interventions in such a way as to make the entire formula true. Thus, we introduce a causal model which encompasses the 'parallel worlds' described by the counterfactual formula.

Before doing so, we give a simple notion of union of submodels, as follows:

Definition 6 (causal model union) Let M_x , and M_z be submodels derived from M . Then $M_x \cup M_z$ is defined to be M_x if $z = x$, and $\langle U, V_x \cup V_z, F_x \cup F_z, P(u) \rangle$, otherwise.

Definition 7 (parallel worlds model) Let M be a causal model, α a counterfactual formula. Then the parallel worlds model M_α is the causal model union of the submodels corresponding to atomic counterfactuals of α .

We call the joint distribution $P(V_\alpha)$ over the endogenous variables in M_α a counterfactual distribution, and will sometimes denote it as P_α . In the language of the potential outcomes framework [Rubin, 1974], we can view P_α as the joint distribution over the unit-response variables mentioned in α .

The parallel worlds model is a generalization of the twin network model, first appearing in [Balke and Pearl, 1994], to more than two possible worlds. It displays independence assumptions between counterfactual quantities in the same way a regular causal model displays independence assumptions between observable quantities – by positing counterfactuals are independent of their non-descendants given their parents.

Given a causal model M and a formula α , we are interested in whether the corresponding counterfactual joint distribution P_α (or its marginal distributions) can be computed uniquely from the set of joint distributions available to the investigator. The formal statement of this question is as follows:

Definition 8 (identifiability) *Let M be a causal model from a set of models $\mathcal{M}$ inducing the same graph G , M_α a parallel worlds model, and Q be a marginal distribution of the counterfactual joint distribution P_α . Let K be a set of known probability distributions derived from M . Then Q is K -identifiable in $\mathcal{M}$ if it is unique and computable from K in any $M \in \mathcal{M}$.*

It follows from the definition that if we can construct two models in $\mathcal{M}$ with the same K but different Q , then Q is not identifiable. An important, well-studied special case of this problem – which we call evidential identifiability of interventions – assumes α is an atomic counterfactual, and K is the joint distribution over the endogenous variables in M , or $P(V)$. Being able to identify an interventional marginal in this way is being able to compute the effects of an intervention without having to actually perform the intervention, and instead relying on passive, observational data.

In this paper we are concerned with identifying probabilities of counterfactuals formulas using the set P_* of all interventional distributions of M as a given. In other words, we are interested in computing probabilities of counterfactuals from experimental and observational probabilities.

3 Path-Specific Effects

Our aim is to provide simple, graphical conditions for the P_* -identifiability of path-specific effects. To do so, we must formalize such effects as counterfactual formulas, and translate the identifiability conditions on the formula to conditions on the graph.

The following is the formalization of the notion of path-specific effect in terms of a modified causal model, as it appears in [Pearl, 2001]:

Definition 9 (path-specific effect) *Let G be the causal graph associated with model M , and let g be an edge-subgraph of G containing the paths selected for effect analysis (we will refer to g as the **effect subgraph**). The g -specific effect of z on Y (relative to reference z^*) is defined as the total effect of z on Y in a modified model M_g formed as follows. Let each parent set PA^i in G be partitioned into two parts $PA^i = \{PA^i(g), PA^i(\bar{g})\}$, where $PA^i(g)$ represents those members of PA^i that are linked to V^i in g , and $PA^i(\bar{g})$ represents the complementary set. We replace each function f^i in M with a new function f_g^i in M_g , defined as follows: for every set of instantiations $pa^i(g)$ of $PA^i(g)$, $f_g^i(pa^i(g), \mathbf{u}) = f^i(pa^i(g), pa^i(\bar{g})^*, \mathbf{u})$, where $pa^i(\bar{g})^*$ takes*

the value of $PA^i(\bar{g})_{z^}(\mathbf{u})$ in M . The collection of modified functions forms a new model M_g . The g -specific effect of z on Y , denoted $SE_g(z, z^*; Y, \mathbf{u})_M$ is defined as the total effect (abbreviated as TE) of z on Y in the modified model:*

$$SE_g(z, z^*; Y, \mathbf{u})_M = TE(z, z^*; Y, \mathbf{u})_{M_g} \quad (2)$$

where $TE(z, z^*; Y, \mathbf{u})_{M_g} = Y_z(\mathbf{u})_{M_g} - Y_{z^*}(\mathbf{u})_{M_g}$.

If we wish to summarize the path-specific effect over all settings of $\mathbf{u}$, we should resort to the expectation of the above difference, or the expected path-specific effect. To identify this effect, we need to identify $P(y_z)$ and $P(y_{z^*})$ in M_g . For our purposes we can restrict our attention to $P(y_z)$, as the second term corresponds to the quantity $P(y_{z^*})$ in the original model M , and so is trivially P_* -identifiable.

In this paper we assume, without loss of generality, edges in $\bar{g} = G \setminus g$ are all along directed paths between Z and Y . The next theorem states that any path specific effect, expressed as a total effect in the modified model M_g , can be expressed as a counterfactual formula in the original model M .

Theorem 1 *Every path specific effect $P(y_z)_{M_g}$ has a corresponding counterfactual formula α in M s.t for every $\mathbf{u}$,*

$$M_g \models y_z(\mathbf{u}) \iff M \models \alpha(\mathbf{u})$$

Proof outline: The proof is for causal models with finite domains. Fix M , $\mathbf{u}$, y , z and g . To prove the theorem, we need to 'unroll' y_z and remove any implicit references to modified functions in M_g , while preserving the truth value of the statement. Our proof will use the axiom of composition, known to hold true for causal models under consideration. In our language, the axiom states that for any three variables Z, Y, W , and any settings $\mathbf{u}, z, w, y$, $(W_z = w \Rightarrow Y_{z,w} = Y_z)(\mathbf{u})$.

Fix $\mathbf{u}_1$. Let $\mathcal{S} = An(Y) \cap De(Z)$ Then by axiom of composition, $y_z(\mathbf{u}_1)$ has the same truth value as a conjunction of atomic formulas of the form $v_{pa^i(g)}^i$, where $V^i \in \mathcal{S}$, $PA^i(g)$ is the set of parents of V^i in M_g , and $pa^i(g)$ and v^i are suitably chosen constants. Denote this conjunction α_1 .

For every term $v_{pa^i(g)}^i$ in α_1 corresponding to V^i with $PA^i(g) \subset PA^i$, replace it by $v_{pa^i(g), pa^i(\bar{g})^*}^i \wedge pa^i(\bar{g})_{z^*}^*$ in the conjunction, where $pa^i(\bar{g})^*$ takes the value of $PA^i(\bar{g})_{z^*}(\mathbf{u}_1)$ in M . Denote the result α_1^* . Note that α_1^* is in M and $M_g \models y_z(\mathbf{u}_1) \iff M \models \alpha_1^*(\mathbf{u}_1)$. We construct a similar conjunction α_j^* for every instantiation $\mathbf{u}_j$ in M . Let $\alpha = \bigvee_j \alpha_j^*$. It's easy to see the claim holds for α by construction. $\square$

An easy corollary of the theorem is, as before, that $P(y_z)_{M_g} = P(\alpha)_M$. Note that different α_i in the proof only differ in the values they assign to variables in $\mathcal{S}$. Since M is composed of functions, the values of variables in $\mathcal{S}$ are fixed given $\mathbf{u}$, and since $P(\alpha) = \sum_{\{u | M \models \bigvee_i \alpha_i(u)\}} P(u)$ by definition, we can express $P(\alpha)$ as a summation over the variables in $\mathcal{S} \setminus \{Y\}$.

For instance, the first term of the path-specific effect in Fig.

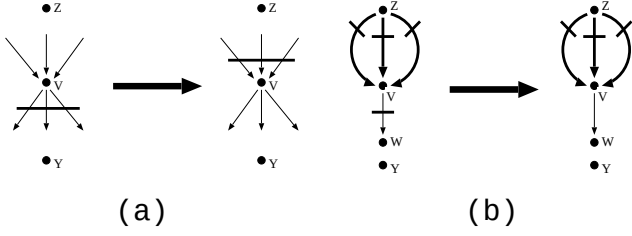


Figure 4: Bold edges represent directed paths (a) R_1 Rule (b) R_2 Rule

2 (a) can be expressed as

$$\begin{aligned} P(s_a)_{M_{g_{2a}}} &= \sum_{k,b,p,h} P(s_{k,b,p,a} \wedge k_h \wedge b_p \wedge p_{a*} \wedge h_{a*}) \\ &= \sum_{h,p} P(s_{a,h,p} \wedge h_{a*} \wedge p_{a*}) \end{aligned} \quad (3)$$

which is just the direct effect. The more general case of Fig. 3 (a) can be expressed as:¹

$$\begin{aligned} P(s_a)_{M_{g_{3a}}} &= \sum_{k,b,p,h} P(s_{k,b,p,a} \wedge k_h \wedge b_{a*} \wedge p_a \wedge h_a) \\ &= \sum_b P(s_{a,b} \wedge b_{a*}) \end{aligned} \quad (4)$$

It looks as if the expressions in Eq. (3) and (4) for the two effects are very similar, moreover we know that direct effects are always P_* -identifiable in Markovian models. Surprisingly, the path specific effect of Fig. 3 (a) and Eq. (4) is not P_* -identifiable as we will show later.

We will find it useful to modify the effect subgraph g while preserving the value of the path-specific effect. We do so by means of the following two rules. Let M be a causal model with the graph G , g an effect subgraph of G , and $\bar{g} = G \setminus g$. For a node V , let $in(V)$ denote the set of edges incoming into V , and $out(V)$ denote the set of edges outgoing from V , in G .

R_1 : If there is a node V in G such that $out(V) \subseteq \bar{g}$, then $R_1(g) = (g \setminus out(V)) \cup in(V)$. See Fig. 4 (a).

R_2 : If there is an edge $e \in \bar{g}$, such that for all directed paths from Z to Y which include e , there exists another edge $e' \in \bar{g}$, which occurs 'upstream' from e , then $R_2(g) = g \setminus \{e\}$. See Fig. 4 (b).

Theorem 2 (Effect-Invariant Rules) *If R_1 is applicable the $R_1(g)$ -specific effect is equal to the g -specific effect. If R_2 is applicable the $R_2(g)$ -specific effect is equal to the g -specific effect.*

Proof outline: The proof is by induction on graph structure, and is an easy consequence of the definition of g -specific effect, and the R_1 and R_2 rules. $\square$

Intuitively, R_1 'moves' the blocked edges closer to the manipulated variable Z , and R_2 removes redundant blocked

¹Note that Eq (4) is different from $\sum_{b_{a*}} P(s_{a,b} \wedge b_{a*})$ which is just a marginalization over the counterfactual variable b_{a*}

Table 1: The functions f_R^1 and f_R^2

Z	U_R	$R = f_R^1(z, u_R)$	$R = f_R^2(z, u_R)$
0	1	0	1
0	2	1	1
0	3	1	0
1	1	1	1
1	2	0	0
1	3	0	0

edges. Thus, it is not surprising these two identities cannot be applied forever in a dag.

Lemma 1 *Let M be a causal model, g an effect subgraph. Then any sequence of applications of R_1 and R_2 to g will reach a fixed point g^* .*

4 Problematic Counterfactual Formulas

Identification of a distribution must precede its estimation, as there is certainly no hope of estimating a quantity not uniquely determined by the modeling assumptions. Furthermore, uniqueness frequently cannot be guaranteed in causal models. For instance, when identifying interventions from observational data, a particular graph structure, the 'bow-arc', has proven to be troublesome. Whenever the graph of a causal model contains the bow-arc, certain experiments become unidentifiable [Pearl, 2000]. Our investigation revealed that a similarly problematic structure exists for experimental identifiability, which we call the 'kite graph', due to its shape. The kite graph arises when we try to identify counterfactual probabilities of the form $P(r_{z*} \wedge r'_z)$.

Lemma 2 *Let M be a causal model, let Z and R be variables such that Z is a parent of R . Then $P(r_{z*} \wedge r'_z)$ is not P_* -identifiable if $z^* \neq z$.*

Proof outline: The proof is by counter example. We let $\alpha = r_{z*} \wedge r'_z$, and construct two causal models M^1 and M^2 that agree on the interventional distribution set P_* , but disagree on $P(\alpha)$. In fact, we only need 2 variables. The two models agree on the following: Z is the parent of R , U_Z , Z and R are binary variables, U_R be a ternary variable, $f_Z = U_Z$, and $P(u_Z)$, and $P(u_R)$ are uniform. The two models only differ on the functions f_R , which are given by table 4. It's easy to verify our claim holds for the two models for any values $z^* \neq z$ of Z . $\square$

The next theorem shows how a particular path-specific effect leads to problematic counterfactuals from the previous lemma.

Theorem 3 *The g -specific effect of Z on Y as described in Fig. 5 (a) is not P_* -identifiable.*

Proof: We extend models M^1 and M^2 from the previous proof with additional variables V , Y , and U_Y . We assume $P(u_Y)$ is uniform, and both $P(V, Y|R)$ and the functions which determine V and Y are the same in both models.

Note that since all variables are discrete, the conditional probability distributions can be represented as tables. If we require $|R| = |V|$ and $|Y| = |V| * |R|$, then the conditional

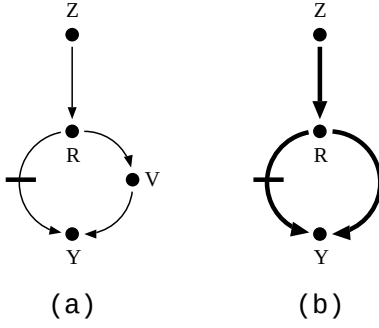


Figure 5: (a) Problematic effect (b) The kite graph

probabilities are representable as square matrices. We fix the functions f_V and f_Y , as well as the exogenous parents of V and Y such that the matrices corresponding to $P(V, Y|R)$ and $P(V|Y)$ are invertible.

Call the extended models M^3 and M^4 . Note that by construction, the two models are Markovian. Since M^1 and M^2 have the same P_* , and since the two extended models agree on all functions and distributions not in M^1 and M^2 , they must also have the same P_* .

Consider the g -specific effect shown in Fig. 5 (a). From Theorem 1 we can express the path-specific effect in M_g^3 in terms of M^3 . In particular:

$$\begin{aligned} P(y_z)_{M_g^3} &= \sum_{rv} P(y_{rv} \wedge r_{z^*} \wedge v_z)_{M^3} \\ &= \sum_{r,v,r'} P(y_{rv} \wedge r_{z^*} \wedge v_{r'} \wedge r'_z)_{M^3} \\ &= \sum_{r,v,r'} P(y_{rv})_{M^3} P(v_{r'})_{M^3} P(r_{z^*}, r'_z)_{M^3} \end{aligned}$$

The last step is licensed by the independence assumptions encoded in the parallel worlds model of $y_{rv} \wedge r_{z^*} \wedge v_{r'} \wedge r'_z$. The same expression can be derived for $P(y_z)_{M_g^4}$. Note that since P_* is the same for both models they have the same values for the interventional distributions $P(y_{rv})$ and $P(v_{r'})$. Note that since $P(Y|R, V)$ and $P(V|R)$ are square matrices, the summing out of $P(Y|R, V)$ and $P(V|R)$ can be viewed as a *linear transformation*. Since the matrices are invertible, the transformations are one to one, and so if their composition. Since $P(y_{rv}) = P(y|r, v)$ and $P(v_{r'}) = P(v|r')$, and since $P(r_{z^*} \wedge r'_z)$ is different in the two models, we obtain that $P(y_z)_{M_g^3} \neq P(y_z)_{M_g^4}$. Since adding directed or bidirected edges to a graph cannot help identifiability, the result also holds in semi-Markovian models. $\square$

5 Main Result

The main result of this section is that a simple sufficient and necessary (in Markovian models) graphical criterion exists. This condition is easily stated and can be derived from the effect subgraph g in linear time. By contrast, the only other methods known to us for obtaining identifiability results of probabilities of general counterfactual logic formulas are proof search procedures based on results in [Galles and

Pearl, 1998], [Halpern, 2000]. Such procedures are far less intuitive, do not have running time bounds, and cannot be used to obtain non-identifiability proofs.

First let's define this criterion:

Definition 10 (Recanting witness criterion) Let $R \neq Z$ be a node in G , such that there exists a directed path in g from Z to R , a directed path from R to Y in g , and a direct path from R to Y in G but not g . Then Z, Y , and g satisfy the recanting witness criterion with R as a witness

The recanting witness criterion is illustrated graphically as the 'kite pattern' in Fig. 5 (b). The name 'recanting witness' comes from the behavior of the variable R in the center of the 'kite.' This variable, in some sense, 'tries to have it both ways.' Along one path from R to Y , R behaves as if the variable Z was set to one value, but along another path, R behaves as if Z was set to another value. This 'changing of the story' of R is what causes the problem, and as we will show it essentially leads to the existence of a non P_* -identifiable expression of the type discussed in section 4.

To proceed, we must make use of the following helpful lemmas: Let g be an effect subgraph of G and g^* the fixed point of R_1 and R_2 . Let $\bar{g}^* = G \setminus g^*$.

Lemma 3 g^* satisfies the recanting witness criterion iff g does. Moreover, if g^* does satisfy the criterion, then there exists a witness R s.t $out(R) \cap \bar{g}^* \neq \emptyset$. If g^* does not, then $\bar{g}^* \subseteq out(Z)$.

Lemma 3 states that repeated applications of rules R_1 and R_2 preserves the satisfaction of the recanting witness criterion. Moreover, if the witness exists in the fixed point g^* , then some outgoing edge from it is blocked. If the witness does not exist in g^* , then only root-emerging edges are blocked.

Lemma 4 Assume the g^* -specific effect of Z on Y is P_* -identifiable. Let E be any set of edges in $\bar{g}^*$. Let $g' = E \cup g^*$. Then the g' -specific effect of Z on Y is P_* -identifiable.

Lemma 4 states that if a path specific effect is not identified, then adding blocked directed edges 'does not help,' in that the effect remains unidentified. Now we can state and prove the main results:

Theorem 4 If g satisfies the recanting witness criterion, then the g -specific effect of Z on Y is not P_* -identifiable.

Proof: Let M be our model and assume that g satisfies the recanting witness criterion. By Lemma 3 so does $\bar{g}^*$, let R be the witness from the lemma s.t $e = R \rightarrow V$ is in $\bar{g}^*$. Assume the g -specific effect is identifiable, By Theorem 2 so is the g^* -specific effect. Let g' be the path specific effect obtained by adding all edges to g^* , but e . By Lemma 4 the g' -specific effect is also P_* -identifiable. Now by composing the functions in g' we can obtain a new model M' which is exactly the model of Fig. 5 (a)² and $P(y_z)_{M_{g'}} = P(y_z)_{M'}$. From Theorem 3 we know that $P(y_z)_{M_{g'}}$ is not P_* -identifiable, hence, neither is $P(y_z)_{M_{g'}}$ and the g' -specific effect is not P_* -identifiable. Contradiction. $\square$ To illustrate the use of

² or a similar model where we "cut" the edge $R \rightarrow V$ and not the edge $R \rightarrow Y$

the theorem, consider the example in Eq. (4) from Section 3. The expression $\sum_b P(s_{a,b} \wedge b_{a^*}) =$

$$\begin{aligned} &= \sum_{b,p} P(s_{a,b} \wedge b_{p'} \wedge p'_{a^*}) \\ &= \sum_{b,p,p'} P(s_{a,b,p} \wedge b_{p'} \wedge p'_{a^*} \wedge p_a) \\ &= \sum_{b,p,p'} P(s_{a,b,p} \wedge b_{p'}) P(p'_{a^*} \wedge p_a) \end{aligned} \quad (5)$$

The first two steps are by definition, the last step is licensed by the parallel worlds model corresponding to the formula in Eq. 5. The theorem shows that, as in this example, non-identifiability arises because formulas of the form $p'_{a^*} \wedge p_a$ appear whenever the recanting witness criterion holds.

Theorem 5 *If g does not satisfy the recanting witness criterion, then the g -specific effect of Z on Y is P_* -identifiable in Markovian models.*

Proof: From theorem 2 we have that $P(y_z)_{M_{g^*}} = P(y_z)_{M_g}$. Since g does not satisfy the recanting witness criterion, by Lemma 3 all the edges in $\bar{g}^*$ emanate from Z . From Theorem 1 there is a formula $\alpha(g^*)$ corresponding to $P(y_z)_{M_{g^*}}$ that contains only atomic counterfactuals of the form $v_{pa^i}^i$. Since all blocked edges emanate from Z , it can be easily observed that for each two atomic counterfactuals in $\alpha(g^*)$, $v_{pa^i}^i, v_{pa^j}^j$, $i \neq j$. This follows, since we only introduce atomic counterfactuals with $do(z^*)$ where we cut edges. Now since in Markovian models any two different variables are independent if you set all their parents, all the atomic counterfactual in $\alpha(g^*)$ are independent of each other which makes the expression P_* -identifiable. $\square$

For example, we stated earlier that the g specific effect of Fig 3 (b) is identifiable, this is true since g does not satisfy the recanting witness criterion. In particular the expression for the path-specific effect is:

$$\begin{aligned} P(s_a)_{M_{g3b}} &= \sum_{k,b,p,h} P(s_{k,b,p,a} \wedge k_h \wedge b_a \wedge p_a \wedge h_{a^*}) \\ &= \sum_h P(s_{h,a} \wedge h_{a^*}) \\ &= \sum_h P(s_{h,a}) P(h_{a^*}) \end{aligned} \quad (6)$$

As before, the first two steps are by definition, and the last step is licensed by the parallel worlds model corresponding to the formula in Eq. 6. But now note that $P(s_{h,a}), P(h_{a^*}) \in P_*$, therefore the above expression can be computed from experiments.

6 Conclusions

Our paper presented a sufficient and necessary graphical conditions for the experimental identifiability of path-specific effects, using tools from probability theory, graph theory, and counterfactual logic. We related identifiable path-specific effects to direct and indirect effects by showing that all such effects only block root-emanating edges.

While it is possible to give a sufficient condition for identifiability of general counterfactual formulas in our language, using induction on formula structure, this does not give a single necessary and sufficient condition for semi-Markovian models. The search for such a condition is a good direction for future work.

Another interesting direction is to consider special cases of causal models where path-specific effects can be identified even in the presence of the 'kite' – this is true in linear models, for instance.

Finally, our result assumes causal models with finite domains, and 'small' graphs. An interesting generalization is to consider causal models with 'large' or infinite graphs and infinite domains. Such models may require adding first-order features to the language.

7 Acknowledgements

The authors would like to thank Brian Gaeke and Paul Twohey for proofreading earlier versions of this paper.

References

- [Balke and Pearl, 1994] Alexander Balke and Judea Pearl. Counterfactual probabilities: Computational methods, bounds and applications. In *Proceedings of UAI-94*, pages 46–54, 1994.
- [Galles and Pearl, 1995] David Galles and Judea Pearl. Testing identifiability of causal effects. In *Proceedings of UAI-95*, pages 185–195, 1995.
- [Galles and Pearl, 1998] David Galles and Judea Pearl. An axiomatic characterization of causal counterfactuals. *Foundation of Science*, 3:151–182, 1998.
- [Halpern, 1990] Joseph Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46(3):311–350, 1990.
- [Halpern, 2000] Joseph Halpern. Axiomatizing causal reasoning. *Journal of A.I. Research*, pages 317–337, 2000.
- [Pearl, 2000] Judea Pearl. *Causality: models, reasoning, and inference*. Cambridge University Press, 2000.
- [Pearl, 2001] Judea Pearl. Direct and indirect effects. In *Proceedings of UAI-01*, pages 411–420, 2001.
- [Robins and Greenland, 1992] James M. Robins and Sander Greenland. Identifiability and exchangeability of direct and indirect effects. *Epidemiology*, 3:143–155, 1992.
- [Robins, 1997] James M. Robins. Causal inference from complex longitudinal data. In *Latent Variable Modeling and Applications to Causality*, volume 120, pages 69–117, 1997.
- [Rubin, 1974] D. B. Rubin. Estimating causal effects of treatments in randomized and non-randomized studies. *Journal of Educational Psychology*, 66:688–701, 1974.

Pushing the $\mathcal{EL}$ Envelope

Franz Baader and Sebastian Brandt and Carsten Lutz

Institute for Theoretical Computer Science

TU Dresden, Germany

lastname@tcs.inf.tu-dresden.de

Abstract

Recently, it has been shown that the small description logic (DL) $\mathcal{EL}$, which allows for conjunction and *existential restrictions*, has better algorithmic properties than its counterpart $\mathcal{FL}_0$, which allows for conjunction and *value restrictions*. Whereas the subsumption problem in $\mathcal{FL}_0$ becomes already intractable in the presence of acyclic TBoxes, it remains tractable in $\mathcal{EL}$ even with general concept inclusion axioms (GCIs). On the one hand, we extend the positive result for $\mathcal{EL}$ by identifying a set of expressive means that can be added to $\mathcal{EL}$ without sacrificing tractability. On the other hand, we show that basically all other additions of typical DL constructors to $\mathcal{EL}$ with GCIs make subsumption intractable, and in most cases even EXPTIME-complete. In addition, we show that subsumption in $\mathcal{FL}_0$ with GCIs is EXPTIME-complete.

1 Introduction

The quest for tractable (i.e., polynomial-time decidable) description logics (DLs), which started in the 1980s after the first intractability results for DLs were shown [Brachman and Levesque, 1984; Nebel, 1988], was until recently restricted to DLs extending the basic language $\mathcal{FL}_0$, which allows for conjunction ($\sqcap$) and value restrictions ($\forall r.C$). The main reason was that, when clarifying the logical status of property arcs in semantic networks and slots in frames, the decision was taken that arcs/slots should be read as value restrictions rather than existential restrictions ($\exists r.C$).

For subsumption between concept descriptions, the tractability barrier was investigated in detail in the early 1990s [Donini *et al.*, 1991]. However, as soon as terminologies (TBoxes) were taken into consideration, tractability turned out to be unattainable: even with the simplest form of acyclic TBoxes, subsumption in $\mathcal{FL}_0$ (and thus in all languages extending it) is coNP-hard [Nebel, 1990]. Subsumption in $\mathcal{FL}_0$ is PSPACE-complete w.r.t. cyclic TBoxes [Baader, 1996; Kazakov and de Nivelle, 2003], and we show in this paper that it becomes even EXPTIME-complete in the presence of general concept inclusion axioms (GCIs), which are supported by all modern DL systems.

For these reasons, and also because of the need for expressive DLs supporting GCIs in applications, from the mid 1990s on the DL community has mainly given up on the quest of finding tractable DLs. Instead, it investigated more and more expressive DLs, for which reasoning is worst-case intractable. The goal was then to find practical subsumption algorithms, i.e., algorithms that are easy to implement and optimize, and which—though worst-case exponential or even worse—behave well in practice (see, e.g., [Horrocks *et al.*, 2000]). This line of research has resulted in the availability of highly optimized DL systems for expressive DLs [Horrocks, 1998; Haarslev and Möller, 2001], and successful applications: most notably the recommendation by the W3C of the DL-based language OWL [Horrocks *et al.*, 2003] as the ontology language for the Semantic Web.

Recently, the choice of value restrictions as a sine qua non of DLs has been reconsidered. On the one hand, it was shown that the DL $\mathcal{EL}$, which allows for conjunction and existential restrictions, has better algorithmic properties than $\mathcal{FL}_0$. Subsumption in $\mathcal{EL}$ stays tractable w.r.t. both acyclic and cyclic TBoxes [Baader, 2003b], and even in the presence of GCIs [Brandt, 2004]. On the other hand, there are applications where value restrictions are not needed, and where the expressive power of $\mathcal{EL}$ or small extensions thereof appear to be sufficient. In fact, SNOMED, the Systematized Nomenclature of Medicine, employs $\mathcal{EL}$ [Spackman, 2000] with an acyclic TBox. Large parts of the Galen medical knowledge base can also be expressed in $\mathcal{EL}$ with GCIs and transitive roles [Rector and Horrocks, 1997]. Finally, the Gene Ontology [Consortium, 2000] can be seen as an acyclic $\mathcal{EL}$ TBox with one transitive role.

Motivated by the positive complexity results cited above and the use of extensions of $\mathcal{EL}$ in applications, we start with the DL $\mathcal{EL}$ with GCIs, and investigate the effect on the complexity of the subsumption problem that is caused by the addition of standard DL constructors available in ontology languages like OWL. We prove that the subsumption problem remains tractable when adding the bottom concept (and thus disjointness statements), nominals (i.e., singleton concepts), a restricted form of concrete domains (e.g., references to numbers and strings), and a restricted form¹ of role-value maps

¹Adding arbitrary role-value maps to $\mathcal{EL}$ is known to cause undecidability [Baader, 2003a].

Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\perp$	$\emptyset$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
concrete domain	$p(f_1, \dots, f_k)$ for $p \in \mathcal{P}^{\mathcal{D}_j}$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y_1, \dots, y_k \in \Delta^{\mathcal{D}_j} : f_i^{\mathcal{I}}(x) = y_i \text{ for } 1 \leq i \leq k \wedge (y_1, \dots, y_k) \in p^{\mathcal{D}_j}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
RI	$r_1 \circ \dots \circ r_k \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \dots \circ r_k^{\mathcal{I}} \subseteq r^{\mathcal{I}}$

Table 1: Syntax and semantics of $\mathcal{EL}^{++}$.

(which can express transitivity and the right-identity rule required in medical applications [Spackman, 2000]). We then prove that, basically, all other additions of standard DL constructors lead to intractability of the subsumption problem, and in most cases even to EXPTIME-hardness. Proofs and further technical details can be found in the accompanying technical report [Baader et al., 2005].

2 The Description Logic $\mathcal{EL}^{++}$

In DLs, *concept descriptions* are inductively defined with the help of a set of *constructors*, starting with a set N_C of *concept names*, a set N_R of *role names*, and (possibly) a set N_I of *individual names*. In this section, we introduce the extension $\mathcal{EL}^{++}$ of $\mathcal{EL}$, whose concept descriptions are formed using the constructors shown in the upper part of Table 1. There and in general, we use a and b to denote individual names, r and s to denote role names, and C, D to denote concept descriptions.

The concrete domain constructor provides an interface to so-called concrete domains, which permits reference to, e.g., strings and integers. Formally, a *concrete domain* $\mathcal{D}$ is a pair $(\Delta^{\mathcal{D}}, \mathcal{P}^{\mathcal{D}})$ with $\Delta^{\mathcal{D}}$ a set and $\mathcal{P}^{\mathcal{D}}$ a set of *predicate names*. Each $p \in \mathcal{P}$ is associated with an arity $n > 0$ and an extension $p^{\mathcal{D}} \subseteq (\Delta^{\mathcal{D}})^n$. To provide a link between the DL and the concrete domain, we introduce a set of *feature names* N_F . In Table 1, p denotes a predicate of some concrete domain $\mathcal{D}$ and $f_1, \dots, f_k$ are feature names. The DL $\mathcal{EL}^{++}$ may be equipped with a number of concrete domains $\mathcal{D}_1, \dots, \mathcal{D}_n$ such that $\Delta^{\mathcal{D}_i} \cap \Delta^{\mathcal{D}_j} = \emptyset$ for $1 \leq i < j \leq n$. If we want to stress the use of particular concrete domains $\mathcal{D}_1, \dots, \mathcal{D}_n$, we write $\mathcal{EL}^{++}(\mathcal{D}_1, \dots, \mathcal{D}_n)$ instead of $\mathcal{EL}^{++}$.

The semantics of $\mathcal{EL}^{++}(\mathcal{D}_1, \dots, \mathcal{D}_n)$ -concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The *domain* $\Delta^{\mathcal{I}}$ is a non-empty set of individuals and the *interpretation function* $\cdot^{\mathcal{I}}$ maps each concept name $A \in N_C$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$, each individual name $a \in N_I$ to an individual $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, and each feature name $f \in N_F$ to a partial function $f^{\mathcal{I}}$ from $\Delta^{\mathcal{I}}$ to $\bigcup_{1 \leq i \leq n} \Delta^{\mathcal{D}_i}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined as shown in the third column of Table 1.

An $\mathcal{EL}^{++}$ *constraint box* (CBox) is a finite set of *general concept inclusions* (GCIs) and *role inclusions* (RIs), whose

syntax can be found in the lower part of Table 1. Note that a finite set of GCIs would commonly be called a *general TBox*. We use the term CBox due to the presence of RIs. An interpretation $\mathcal{I}$ is a *model* of a CBox $\mathcal{C}$ if, for each GCI and RI in $\mathcal{C}$, the conditions given in the third column of Table 1 are satisfied. In the definition of the semantics of RIs, the symbol “ $\circ$ ” denotes composition of binary relations.

The main inference problem considered in this paper is subsumption. Given two $\mathcal{EL}^{++}$ -concept descriptions C, D we say that C is *subsumed by* D w.r.t. the CBox $\mathcal{C}$ ($C \sqsubseteq_{\mathcal{C}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{C}$.

Some remarks regarding the expressivity of $\mathcal{EL}^{++}$ are in order. First, though we restrict the attention to subsumption, $\mathcal{EL}^{++}$ is expressive enough to reduce all other standard reasoning tasks (concept satisfiability, ABox consistency, instance problem) to the subsumption problem and vice versa [Baader et al., 2005]. Second, our RIs generalize three means of expressivity important in ontology applications: *role hierarchies* $r \sqsubseteq s$; *transitive roles*, which can be expressed by writing $r \circ r \sqsubseteq r$; and so-called *right-identity rules* $r \circ s \sqsubseteq s$, which are important in medical applications [Spackman, 2000; Horrocks and Sattler, 2003]. Third, the bottom concept in combination with GCIs can be used to express *disjointness* of complex concept descriptions: $C \sqcap D \sqsubseteq \perp$ says that C, D are disjoint. Finally, the *unique name assumption* for individual names can be enforced by writing $\{a\} \sqcap \{b\} \sqsubseteq \perp$ for all relevant individual names a and b .

3 Tractability of $\mathcal{EL}^{++}$

Before we can describe a polynomial-time subsumption algorithm for $\mathcal{EL}^{++}$, we must introduce an appropriate normal form for CBoxes. Given a CBox $\mathcal{C}$, we use $\text{BC}_{\mathcal{C}}$ to denote the smallest set of concept descriptions that contains the top concept $\top$, all concept names used in $\mathcal{C}$, and all concept descriptions of the form $\{a\}$ or $p(f_1, \dots, f_k)$ appearing in $\mathcal{C}$. Then, $\mathcal{C}$ is in *normal form* if

1. all GCIs have one of the following forms, where $C_1, C_2 \in \text{BC}_{\mathcal{C}}$ and $D \in \text{BC}_{\mathcal{C}} \cup \{\perp\}$:

$$\begin{array}{ll} C_1 \sqsubseteq D, & C_1 \sqsubseteq \exists r.C_2, \\ C_1 \sqcap C_2 \sqsubseteq D, & \exists r.C_1 \sqsubseteq D. \end{array}$$

2. all role inclusions are of the form $r \sqsubseteq s$ or $r_1 \circ r_2 \sqsubseteq s$.

By introducing new concept and role names, any CBox $\mathcal{C}$ can be turned into a normalized CBox $\mathcal{C}'$ that is a *conservative extension* of $\mathcal{C}$, i.e., every model of $\mathcal{C}'$ is also a model of $\mathcal{C}$, and every model of $\mathcal{C}$ can be extended to a model of $\mathcal{C}'$ by appropriately choosing the interpretations of the additional concept and role names. In [Baader et al., 2005] it is shown that this transformation can actually be done in linear time, yielding a normalized CBox $\mathcal{C}'$ whose size is *linear* in the size of $\mathcal{C}$.

Lemma 1 *Subsumption w.r.t. CBoxes in $\mathcal{EL}^{++}$ can be reduced in linear time to subsumption w.r.t. normalized CBoxes in $\mathcal{EL}^{++}$.*

In the following, all CBoxes are assumed to be normalized. When developing the subsumption algorithm for normalized $\mathcal{EL}^{++}$ CBoxes, we can restrict our attention to subsumption

- CR1** If $C' \in S(C)$, $C' \sqsubseteq D \in \mathcal{C}$, and $D \notin S(C)$
then $S(C) := S(C) \cup \{D\}$
- CR2** If $C_1, C_2 \in S(C)$, $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{C}$, and $D \notin S(C)$
then $S(C) := S(C) \cup \{D\}$
- CR3** If $C' \in S(C)$, $C' \sqsubseteq \exists r.D \in \mathcal{C}$, and $(C, D) \notin R(r)$
then $R(r) := R(r) \cup \{(C, D)\}$
- CR4** If $(C, D) \in R(r)$, $D' \in S(D)$, $\exists r.D' \sqsubseteq E \in \mathcal{C}$,
and $E \notin S(C)$
then $S(C) := S(C) \cup \{E\}$
- CR5** If $(C, D) \in R(r)$, $\perp \in S(D)$, and $\perp \notin S(C)$,
then $S(C) := S(C) \cup \{\perp\}$
- CR6** If $\{a\} \in S(C) \cap S(D)$, $C \rightsquigarrow_R D$, and $S(D) \not\sqsubseteq S(C)$
then $S(C) := S(C) \cup S(D)$
- CR7** If $\text{con}_j(S(C))$ is unsatisfiable in $\mathcal{D}_j$ and $\perp \notin S(C)$,
then $S(C) := S(C) \cup \{\perp\}$
- CR8** If $\text{con}_j(S(C))$ implies $p(f_1, \dots, f_k) \in \text{BC}_{\mathcal{C}}$ in $\mathcal{D}_j$
and $p(f_1, \dots, f_k) \notin S(C)$,
then $S(C) := S(C) \cup \{p(f_1, \dots, f_k)\}$
- CR9** If $p(f_1, \dots, f_k), p'(f'_1, \dots, f'_k) \in S(C)$, $p \in \mathcal{P}^{\mathcal{D}_j}$,
 $p' \in \mathcal{P}^{\mathcal{D}_\ell}$, $j \neq \ell$, $f_s = f'_t$ for some s, t , and $\perp \notin S(C)$,
then $S(C) := S(C) \cup \{\perp\}$
- CR10** If $(C, D) \in R(r)$, $r \sqsubseteq s \in \mathcal{C}$, and $(C, D) \notin R(s)$
then $R(s) := R(s) \cup \{(C, D)\}$
- CR11** If $(C, D) \in R(r_1)$, $(D, E) \in R(r_2)$, $r_1 \circ r_2 \sqsubseteq r_3 \in \mathcal{C}$,
and $(C, E) \notin R(r_3)$
then $R(r_3) := R(r_3) \cup \{(C, E)\}$

Table 2: Completion Rules

between concept *names*. In fact, $C \sqsubseteq_{\mathcal{C}} D$ iff $A \sqsubseteq_{\mathcal{C}'} B$, where $\mathcal{C}' = \mathcal{C} \cup \{A \sqsubseteq C, D \sqsubseteq B\}$ with A and B new concept names. Our subsumption algorithm not only computes subsumption between two given concept names w.r.t. the normalized input CBox $\mathcal{C}$; it rather *classifies* $\mathcal{C}$, i.e., it simultaneously computes the subsumption relationships between *all* pairs of concept names occurring in $\mathcal{C}$.

Now, let $\mathcal{C}$ be a CBox in normal form that is to be classified. We use $R_{\mathcal{C}}$ to denote the set of all role names used in $\mathcal{C}$. The algorithm computes

- a mapping S from $\text{BC}_{\mathcal{C}}$ to a subset of $\text{BC}_{\mathcal{C}} \cup \{\top, \perp\}$, and
- a mapping R from $R_{\mathcal{C}}$ to a binary relation on $\text{BC}_{\mathcal{C}}$.

The intuition is that these mappings make implicit subsumption relationships explicit in the following sense:

- (I1) $D \in S(C)$ implies that $C \sqsubseteq_{\mathcal{C}} D$,
- (I2) $(C, D) \in R(r)$ implies that $C \sqsubseteq_{\mathcal{C}} \exists r.D$.

In the algorithm, these mappings are initialized as follows:

- $S(C) := \{C, \top\}$ for each $C \in \text{BC}_{\mathcal{C}}$,
- $R(r) := \emptyset$ for each $r \in R_{\mathcal{C}}$.

Then the sets $S(C)$ and $R(r)$ are extended by applying the completion rules shown in Table 2 until no more rule applies.

Some of the rules use abbreviations that still need to be introduced. First, **CR6** uses the relation $\rightsquigarrow_R \subseteq \text{BC}_{\mathcal{C}} \times$

$\text{BC}_{\mathcal{C}}$, which is defined as follows: $C \rightsquigarrow_R D$ iff there are $C_1, \dots, C_k \in \text{BC}_{\mathcal{C}}$ such that

- $C_1 = C$ or $C_1 = \{b\}$ for some individual name b ,
- $(C_j, C_{j+1}) \in R(r_j)$ for some $r_j \in R_{\mathcal{C}}$ ($1 \leq j < k$),
- $C_k = D$.

Second, rules **CR7** and **CR8** use the notion $\text{con}_j(S_i(C))$, and satisfiability and implication in a concrete domain. If p is a predicate of the concrete domain $\mathcal{D}_j$, then the $\mathcal{EL}^{++}$ -concept description $p(f_1, \dots, f_n)$ can be viewed as an atomic first-order formula with variables $f_1, \dots, f_n$. Thus, it makes sense to consider Boolean combinations of such atomic formulae, and to talk about whether such a formula is satisfiable in (the first-order interpretation) $\mathcal{D}_j$, or whether in $\mathcal{D}_j$ one such formula implies another one. For a set Γ of $\mathcal{EL}^{++}(\mathcal{D}_1, \dots, \mathcal{D}_n)$ -concept descriptions and $1 \leq j \leq n$, we define

$$\text{con}_j(\Gamma) := \bigwedge_{p(f_1, \dots, f_k) \in \Gamma \text{ with } p \in \mathcal{P}^{\mathcal{D}_j}} p(f_1, \dots, f_k).$$

For the rules **CR7** and **CR8** to be executable in polynomial time, satisfiability and implication in the concrete domains $\mathcal{D}_1, \dots, \mathcal{D}_n$ must be decidable in polynomial time. However, for our algorithm to be complete, we must impose an additional condition on the concrete domains. The concrete domain $\mathcal{D}$ is *p-admissible* iff

1. satisfiability and implication in $\mathcal{D}$ are decidable in polynomial time;
2. $\mathcal{D}$ is *convex*: if a conjunction of atoms of the form $p(f_1, \dots, f_k)$ implies a disjunction of such atoms, then it also implies one of its disjuncts.

Let us now show that the rules of Table 2 indeed yield a polynomial algorithm for subsumption in $\mathcal{EL}^{++}(\mathcal{D}_1, \dots, \mathcal{D}_n)$ provided that the concrete domains $\mathcal{D}_1, \dots, \mathcal{D}_n$ are p-admissible.

The following lemma is an easy consequence of the facts that (i) each rule application adds an element to one of the sets $S(C) \subseteq \text{BC}_{\mathcal{C}} \cup \{\top, \perp\}$ or $R(C, D) \subseteq \text{BC}_{\mathcal{C}} \times \text{BC}_{\mathcal{C}}$, (ii) the cardinality of $\text{BC}_{\mathcal{C}}$ is polynomial in the size of $\mathcal{C}$, and (iii) the relation $\rightsquigarrow_R$ can be computed using (polytime) graph reachability, and (iv) the concrete domains are p-admissible.

Lemma 2 *For a normalized CBox $\mathcal{C}$, the rules of Table 2 can only be applied a polynomial number of times, and each rule application is polynomial.*

The next lemma shows how all subsumption relationships between concept names occurring in $\mathcal{C}$ can be determined once the completion algorithm has terminated.

Lemma 3 *Let S be the mapping obtained after the application of the rules of Table 2 for the normalized CBox $\mathcal{C}$ has terminated, and let A, B be concept names occurring in $\mathcal{C}$. Then $A \sqsubseteq_{\mathcal{C}} B$ iff one of the following two conditions holds:*

- $S(A) \cap \{B, \perp\} \neq \emptyset$,
- there is an $\{a\} \in \text{BC}_{\mathcal{C}}$ such that $\perp \in S(\{a\})$.

The if-direction of this lemma (soundness) immediately follows from the fact that (I1) and (I2) are satisfied for the initial

definition of S, R , and that application of the rules preserves (I1) and (I2). This is trivial for most of the rules. We consider **CR6** in more detail. If $\{a\} \in S(C) \cap S(D)$, then $C, D \sqsubseteq_C \{a\}$. Now, $C \rightsquigarrow_R D$ implies that $C \sqsubseteq_C \exists r_1 \dots \exists r_{k-1}. D$ or $\{b\} \sqsubseteq_C \exists r_1 \dots \exists r_{k-1}. D$ for some individual name b . In the second case, this implies that D cannot be empty in any model of $\mathcal{C}$, and in the first case it implies that D is non-empty in any model of $\mathcal{C}$ for which C is non-empty. Together with $C, D \sqsubseteq_C \{a\}$, this implies that $C \sqsubseteq_C D$, which shows that the rule **CR6** is sound since it preserves (I1).

To show the only-if-direction of the lemma, we assume that the two conditions do not hold, and then use the computed mappings S, R to construct a model $\mathcal{I}$ of $\mathcal{C}$ such that $A^\mathcal{I} \not\sqsubseteq B^\mathcal{I}$. Basically, the domain $\Delta^\mathcal{I}$ of this model consists of all elements $C \in \text{BC}_C$ such that $A \rightsquigarrow_R C$. However, we identify elements C, D of BC_C if there is an individual name a such that $\{a\} \in S(C) \cap S(D)$. The interpretation of concept and role names is determined by S, R : if $D \in \text{BC}_C$ is a concept name, then $D^\mathcal{I} = \{C \in \text{BC}_C \mid D \in S(C)\}$, and if r is a role name, then $(C, D) \in r^\mathcal{I}$ iff $(C, D) \in R(r)$. Finally, the interpretation of the feature names is determined by the assignments satisfying the conjunctions $\text{con}_j(S(C))$ (see [Baader et al., 2005] for a more detailed description of this construction, and a proof that it indeed yields a countermodel to the subsumption relationship $A \sqsubseteq_C B$).

To sum up, we have shown the following tractability result:

Theorem 4 *Let $\mathcal{D}_1, \dots, \mathcal{D}_n$ be p -admissible concrete domains. Then subsumption in $\mathcal{EL}^{++}(\mathcal{D}_1, \dots, \mathcal{D}_n)$ w.r.t. CBoxes can be decided in polynomial time.*

P-admissible and non-admissible concrete domains

In order to obtain concrete DLs of the form $\mathcal{EL}^{++}(\mathcal{D}_1, \dots, \mathcal{D}_n)$ for $n > 0$ to which Theorem 4 applies, we need concrete domains that are p -admissible. In the following, we introduce two concrete domains that are p -admissible, and show that small extensions of them are also p -admissible.

The concrete domain $\mathbb{Q} = (\mathbb{Q}, \mathcal{P}^\mathbb{Q})$ has as its domain the set $\mathbb{Q}$ of rational numbers, and its set of predicates $\mathcal{P}^\mathbb{Q}$ consists of the following predicates:

- a unary predicate $\top_\mathbb{Q}$ with $(\top_\mathbb{Q})^\mathbb{Q} = \mathbb{Q}$;
- unary predicates $=_q$ and $>_q$ for each $q \in \mathbb{Q}$;
- a binary predicate $=$;
- a binary predicate $+_q$, for each $q \in \mathbb{Q}$, with $(+_q)^\mathbb{Q} = \{(q', q'') \in \mathbb{Q}^2 \mid q' + q = q''\}$.

The concrete domain $\mathbb{S}$ is defined as $(\Sigma^*, \mathcal{P}^\mathbb{S})$, where Σ is the ISO 8859-1 (Latin-1) character set and $\mathcal{P}^\mathbb{S}$ consists of the following predicates:

- a unary predicate $\top_\mathbb{S}$ with $(\top_\mathbb{S})^\mathbb{S} = \Sigma^*$;
- a unary predicate $=_w$, for each $w \in \Sigma^*$;
- a binary predicate $=$;
- a binary predicate conc_w , for each $w \in \Sigma^*$, with $\text{conc}_w^\mathbb{S} = \{(w', w'') \mid w'' = w'w\}$.

Polynomiality of reasoning in $\mathbb{Q}$ can be shown by a reduction to linear programming, and polynomiality of reasoning in $\mathbb{S}$

Name	Syntax	Semantics
negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
disjunction	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
value restriction	$\forall r. C$	$\{x \mid \forall y : (x, y) \in r^\mathcal{I} \rightarrow y \in C^\mathcal{I}\}$
at-least restriction	$(\geq n \ r)$	$\{x \mid \#\{y \in \Delta^\mathcal{I} \mid (x, y) \in r^\mathcal{I}\} \geq n\}$
at-most restriction	$(\leq n \ r)$	$\{x \mid \#\{y \in \Delta^\mathcal{I} \mid (x, y) \in r^\mathcal{I}\} \leq n\}$
inverse roles	$\exists r^-. C$	$\{x \mid \exists y : (y, x) \in r^\mathcal{I} \wedge y \in C^\mathcal{I}\}$

Table 3: The additional constructors.

has been proved in [Lutz, 2003]. Moreover, we can show [Baader et al., 2005] that both $\mathbb{Q}$ and $\mathbb{S}$ are convex.

Proposition 5 *$\mathbb{Q}$ and $\mathbb{S}$ are p -admissible.*

Both $\mathbb{Q}$ and $\mathbb{S}$ are interesting concrete domains since they allow us to refer to concrete numbers and strings in concepts, and use the properties of the concrete predicates when reasoning. However, the predicates available in these concrete domains are rather restricted. Unfortunately, p -admissibility is a fragile property, i.e., we cannot extend $\mathbb{Q}$ and $\mathbb{S}$ by other interesting predicates without losing p -admissibility. As an illustration, we consider one extension of $\mathbb{Q}$ and one of $\mathbb{S}$.

The concrete domain $\mathbb{Q}^{\leq q, > q}$ extends $\mathbb{Q}$ by the additional unary predicates $(\leq q)_{q \in \mathbb{Q}}$. Then the $\mathbb{Q}^{\leq q, > q}$ -conjunction $>_0(f')$ implies the disjunction $\leq_0(f) \vee >_0(f)$ without implying one of its disjuncts. Thus, $\mathbb{Q}^{\leq q, > q}$ is not convex.

Next, consider any concrete domain $\mathbb{S}^*$ with domain Σ^* for some finite alphabet Σ and, for every $s \in \Sigma^*$, the unary predicates pref_s and suff_s with the semantics

$$\begin{aligned} \text{pref}_s^{\mathbb{S}^*} &:= \{s' \in \Sigma^* \mid s \text{ is a prefix of } s'\} \\ \text{suff}_s^{\mathbb{S}^*} &:= \{s' \in \Sigma^* \mid s \text{ is a suffix of } s'\}. \end{aligned}$$

Let $\Sigma = \{a_1, \dots, a_n\}$. Then the $\mathbb{S}^*$ -conjunction $\text{suff}_{a_1}(f)$ implies the disjunction $\text{pref}_{a_1}(f) \vee \dots \vee \text{pref}_{a_n}(f)$ without implying any of its disjuncts.

4 Intractable extensions of $\mathcal{EL}$ with GCIs

In this section we consider the sublanguage $\mathcal{EL}$ of $\mathcal{EL}^{++}$ and restrict the attention to general TBoxes, i.e., finite sets of GCIs. Recall that $\mathcal{EL}$ is obtained from $\mathcal{EL}^{++}$ by dropping all concept constructors except conjunction, existential restriction, and top. We will show that the extension of $\mathcal{EL}$ with basically any typical DL constructor not present in $\mathcal{EL}^{++}$ results in intractability of subsumption w.r.t. general TBoxes. Syntax and semantics of the additional constructors used in this section can be found in Table 3, where $\#S$ denotes the cardinality of a set S .

In addition to the subsumption problem, we will sometimes also consider the *satisfiability problem*: the concept description C is satisfiable w.r.t. the general TBox $\mathcal{T}$ iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ with $C^\mathcal{I} \neq \emptyset$. As in the previous section, we can restrict the attention to satisfiability/subsumption of concept *names* w.r.t. general TBoxes.

Atomic negation

Let $\mathcal{EL}^\neg$ be the extension of $\mathcal{EL}$ with negation, and let $\mathcal{EL}^{(\neg)}$ be obtained from $\mathcal{EL}^\neg$ by restricting the applicability of negation to concept names (*atomic negation*). Since $\mathcal{EL}^\neg$ is a notational variant of the DL $\mathcal{ALC}$, EXPTIME-completeness of

satisfiability and subsumption in $\mathcal{ALC}$ w.r.t. general TBoxes [Schild, 1991] carries over to $\mathcal{EL}^\neg$. EXPTIME-completeness even carries over to $\mathcal{EL}^{(\neg)}$ since $\neg C$ with C complex can be replaced with $\neg A$ for a new concept name A if we add the two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$.

Theorem 6 In $\mathcal{EL}^{(\neg)}$, satisfiability and subsumption w.r.t. general TBoxes is EXPTIME-complete.

Disjunction

Let $\mathcal{ELU}$ be the extension of $\mathcal{EL}$ with disjunction. Subsumption in $\mathcal{ELU}$ w.r.t. general TBoxes is in EXPTIME since $\mathcal{ELU}$ is a fragment of $\mathcal{ALC}$. To obtain a matching EXPTIME lower bound, we reduce satisfiability in $\mathcal{EL}^{(\neg)}$ w.r.t. general TBoxes to subsumption in $\mathcal{ELU}$ w.r.t. general TBoxes. To this end, let A_0 be a concept name and $\mathcal{T}$ a general $\mathcal{EL}^{(\neg)}$ TBox. For each concept name A occurring in $\mathcal{T}$, we take a new concept name A' (i.e., one not occurring in $\mathcal{T}$). Also fix an additional new concept name L . Then the general TBox $\mathcal{T}^*$ is obtained from $\mathcal{T}$ by replacing each subconcept $\neg A$ with A' , and then adding the following GCIs:

- $\top \sqsubseteq A \sqcup A'$ and $A \sqcap A' \sqsubseteq L$ for each $A \in N_C$ in $\mathcal{T}$;
- $\exists r.L \sqsubseteq L$.

Note that $\exists r.L \sqsubseteq L$ is equivalent to $\neg L \sqsubseteq \forall r.\neg L$. It thus ensures that L acts as the bottom concept in connected countermodels of $A_0 \sqsubseteq_{\mathcal{T}^*} L$. Using this observation, it is not hard to verify that A_0 is satisfiable w.r.t. $\mathcal{T}$ iff $A_0 \not\sqsubseteq_{\mathcal{T}^*} L$.

Theorem 7 In $\mathcal{ELU}$, subsumption w.r.t. general TBoxes is EXPTIME-complete.

At-least restrictions

Let $\mathcal{EL}^{\geq 2}$ be the extension of $\mathcal{EL}$ with at-least restrictions of the form $(\geq 2 r)$. Subsumption in $\mathcal{EL}^{\geq 2}$ w.r.t. general TBoxes is in EXPTIME since $\mathcal{EL}^{\geq 2}$ is a fragment of $\mathcal{ALC}$ extended with number restrictions [De Giacomo and Lenzenini, 1994]. We establish a matching lower bound by reducing subsumption in $\mathcal{ELU}$ w.r.t. general TBoxes. Let A_0 and B_0 be concept names and $\mathcal{T}$ a general $\mathcal{ELU}$ TBox. Without loss of generality, we may assume that all concept inclusions in $\mathcal{T}$ have one of the following forms:

$$\begin{aligned} C \sqsubseteq D, \quad \exists r.C \sqsubseteq D, \quad C \sqsubseteq \exists r.D, \\ C_1 \sqcap C_2 \sqsubseteq C, \quad C \sqsubseteq C_1 \sqcup C_2, \end{aligned}$$

where C, D, C_1 , and C_2 are concept names or $\top$. Call the resulting TBox $\mathcal{T}^*$. To convert $\mathcal{T}$ into an $\mathcal{EL}^{\geq 2}$ CBox, we simulate each GCI $C \sqsubseteq C_1 \sqcup C_2$ by introducing two new concept names A and B and a new role name r , and putting

$$\begin{aligned} C \sqsubseteq \exists r.A \sqcap \exists r.B, \\ C \sqcap \exists r.(A \sqcap B) \sqsubseteq C_1, \quad C \sqcap (\geq 2 r) \sqsubseteq C_2. \end{aligned}$$

It is easy to see that $A_0 \sqsubseteq_{\mathcal{T}} B_0$ iff $A_0 \sqsubseteq_{\mathcal{T}^*} B_0$.

Theorem 8 In $\mathcal{EL}^{\geq 2}$, subsumption w.r.t. general TBoxes is EXPTIME-complete.

The interested reader may note that similar reductions can be used to show EXPTIME-completeness for $\mathcal{EL}$ extended with one of the role constructors negation, union, and transitive closure.

Non-p-admissible concrete domains

P-admissibility of the concrete domains is not only a sufficient condition for polynomiality of reasoning in $\mathcal{EL}^{++}$, but also a necessary one: if $\mathcal{D}$ is a non-convex concrete domain, then subsumption in $\mathcal{EL}(\mathcal{D})$ is EXPTIME-hard, where $\mathcal{EL}(\mathcal{D})$ is the extension of $\mathcal{EL}$ with the concrete domain $\mathcal{D}$. The proof uses a stronger version of Theorem 7: we can show that subsumption of concept names w.r.t. *restricted* $\mathcal{ELU}$ TBoxes is EXPTIME-complete, where a restricted $\mathcal{ELU}$ TBox is a general $\mathcal{EL}$ TBox extended with a *single* GCI of the form $A \sqsubseteq B_1 \sqcup B_2$ [Baader et al., 2005].

The subsumption problem for such restricted $\mathcal{ELU}$ TBoxes can be reduced to subsumption in $\mathcal{EL}^{++}(\mathcal{D})$ as follows. Let A_0 and B_0 be concept names and $\mathcal{T}$ a restricted $\mathcal{ELU}$ TBox. Since $\mathcal{D}$ is not convex, there is a satisfiable conjunction c of atoms of the form $p(f_1, \dots, f_k)$ that implies a disjunction $a_1 \vee \dots \vee a_m$ of such atoms, but none of its disjuncts. If we assume that this is a minimal such counterexample (i.e., m is minimal), then we also know that c does not imply $a_2 \vee \dots \vee a_m$, and that each of the a_i is satisfiable. Then we have (i) each assignment of values from $\mathcal{D}$ that satisfies c satisfies a_1 or $a_2 \vee \dots \vee a_m$; (ii) there is an assignment satisfying c and a_1 , but not $a_2 \vee \dots \vee a_m$; (iii) there is an assignment satisfying c and $a_2 \vee \dots \vee a_m$ but not a_1 . Now, let $\mathcal{T}^*$ be obtained from $\mathcal{T}$ by replacing the single GCI $A \sqsubseteq B \sqcup B'$ by $A \sqsubseteq c$, $a_1 \sqsubseteq B$, and $a_i \sqsubseteq B'$ for $i = 2, \dots, m$. It is easy to see that $A_0 \sqsubseteq_{\mathcal{T}} B_0$ iff $A_0 \sqsubseteq_{\mathcal{T}^*} B_0$.

Theorem 9 For any non-convex concrete domain $\mathcal{D}$, subsumption in $\mathcal{EL}(\mathcal{D})$ w.r.t. general TBoxes is EXPTIME-hard.

For example, this theorem applies to the non-convex concrete domains introduced in Section 3.

Inverse roles

Let $\mathcal{ELI}$ be the extension of $\mathcal{EL}$ with inverse roles. We show that subsumption in $\mathcal{ELI}$ w.r.t. general TBoxes is PSPACE-hard by reducing satisfiability in $\mathcal{ALC}$ w.r.t. *primitive* TBoxes: $\mathcal{ALC}$ extends $\mathcal{EL}$ with value restrictions and atomic negation; primitive TBoxes are finite sets of GCIs whose left-hand side is a concept *name*. This satisfiability problem is known to be PSPACE-complete [Calvanese, 1996].

Let A_0 be a concept name and $\mathcal{T}$ a primitive $\mathcal{ALC}$ TBox. We assume without loss of generality that $\mathcal{T}$ contains only concept inclusions of the following forms:

$$A \sqsubseteq B, \quad A \sqsubseteq \neg B, \quad A \sqsubseteq \exists r.B, \quad A \sqsubseteq \forall r.B,$$

where A, B , and B' are concept names. We take a new concept name L and define the general $\mathcal{ELI}$ TBox $\mathcal{T}^*$ to consist of the following GCIs:

- $A \sqsubseteq D$ for all $A \sqsubseteq D \in \mathcal{T}$
if D is a concept name or of the form $\exists r.B$;
- $\exists r^-.A \sqsubseteq B$ for all $A \sqsubseteq \forall r.B \in \mathcal{T}$;
- $A \sqcap B \sqsubseteq L$ for all $A \sqsubseteq \neg B \in \mathcal{T}$;
- $\exists r.L \sqsubseteq L$.

As in the case of $\mathcal{ELU}$, the concept inclusion $\exists r.L \sqsubseteq L$ is equivalent to $\neg L \sqsubseteq \forall r.\neg L$ and ensures that L acts as the bottom concept in connected countermodels of $A_0 \sqsubseteq_{\mathcal{T}^*} L$. Additionally, $\exists r^-.A \sqsubseteq B$ is clearly equivalent to $A \sqsubseteq \forall r.B$.

Thus, it is not hard to verify that A_0 is satisfiable w.r.t. $\mathcal{T}$ iff $A_0 \sqsubseteq_{\mathcal{T}^*} L$.

Theorem 10 *Subsumption in $\mathcal{ELI}$ w.r.t. general TBoxes is PSPACE-hard.*

The exact complexity of this problem is still open (the best upper bound we know of is EXPTIME, stemming from results for the DL $\mathcal{ALCT}$ [De Giacomo and Lenzerini, 1994]).

At-most restrictions/functional roles/ $\mathcal{FL}_0$

Let $\mathcal{EL}^{\leq 1}$ be the extension of $\mathcal{EL}$ with at-most restrictions of the form $(\leq 1 \ r)$. As in the case of $\mathcal{EL}^{\geq 2}$, subsumption in $\mathcal{EL}^{\leq 1}$ w.r.t. general TBoxes is in EXPTIME since $\mathcal{EL}^{\leq 1}$ is a fragment of $\mathcal{ALC}$ with number restrictions. We prove a matching lower bound by reducing subsumption in the DL $\mathcal{FL}_0^{tf}$ w.r.t. general TBoxes. $\mathcal{FL}_0^{tf}$ offers only the concept constructors conjunction and value restriction and requires all roles to be interpreted as total functions. Subsumption in this DL w.r.t. general TBoxes was proved EXPTIME-complete in [Toman and Weddell, 2005].

Let A_0 and B_0 be concept names and $\mathcal{T}$ a general $\mathcal{FL}_0^{tf}$ TBox. We convert $\mathcal{T}$ into a general $\mathcal{EL}^{\leq 1}$ TBox $\mathcal{T}^*$ by replacing each subconcept $\forall r.C$ with $\exists r.C$ in GCI left-hand sides, and with $(\leq 1 \ r) \sqcap \exists r.C$ in GCI right-hand sides. Then $A_0 \sqsubseteq_{\mathcal{T}} B_0$ in $\mathcal{FL}_0^{tf}$ iff $A_0 \sqsubseteq_{\mathcal{T}^*} B_0$ in $\mathcal{EL}^{\leq 1}$.

Theorem 11 *Subsumption in $\mathcal{EL}^{\leq 1}$ w.r.t. general TBoxes is EXPTIME-complete.*

Interestingly, the result of Toman and Weddell can also be used to prove EXPTIME-completeness of subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes. Recall that $\mathcal{FL}_0$ is $\mathcal{FL}_0^{tf}$ without the restriction that roles be total functions. In [Baader et al., 2005] we show that $A_0 \sqsubseteq_{\mathcal{T}} B_0$ in $\mathcal{FL}_0$ iff $A_0 \sqsubseteq_{\mathcal{T}} B_0$ in $\mathcal{FL}_0^{tf}$, i.e., subsumption in $\mathcal{FL}_0$ and $\mathcal{FL}_0^{tf}$ coincides.

Theorem 12 *Subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes is EXPTIME-complete.*

5 Conclusion

We believe that the results of this paper show that—in contrast to the negative conclusions drawn from early complexity results in the area—the quest for tractable DLs that are expressive enough to be useful in practice can be successful. Our DL $\mathcal{EL}^{++}$ is tractable even w.r.t. GCIs, and it offers many constructors that are important in ontology applications.

This is in strong contrast to its counterpart with value restrictions: $\mathcal{FL}_0$ is tractable without TBoxes [Brachman and Levesque, 1984], co-NP-complete for acyclic TBoxes [Nebel, 1990], PSPACE-complete for cyclic TBoxes [Baader, 1996; Kazakov and de Nivelle, 2003], and EXPTIME-complete for general TBoxes (as shown above, and, independently, in [Hofmann, 2005]).

References

[Baader et al., 2005] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. LTCS-Report LTCS-05-01, Inst. for Theoretical Computer Science, TU Dresden, 2005. See <http://lat.inf.tu-dresden.de/research/reports.html>.
[Baader, 1996] F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *AMAI*, 18, 1996.

[Baader, 2003a] F. Baader. Restricted role-value-maps in a description logic with existential restrictions and terminological cycles. In *Proc. DL'03*, <http://CEUR-WS.org/Vol-81/>, 2003.
[Baader, 2003b] F. Baader. Terminological cycles in a description logic with existential restrictions. In *Proc. IJCAI'03*, 2003.
[Brachman and Levesque, 1984] R.J. Brachman and H.J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. AAAI'84*, 1984.
[Brandt, 2004] S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. ECAI 2004*, 2004.
[Calvanese, 1996] D. Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In *Proc. ECAI'96*, 1996.
[Consortium, 2000] The GO Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25, 2000.
[De Giacomo and Lenzerini, 1994] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. AAAI'94*, 1994.
[Donini et al., 1991] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Complexity of concept languages. In *Proc. KR'91*, 1991.
[Haarslev and Möller, 2001] V. Haarslev and R. Möller. RACER system description. In *Proc. IJCAR 01*, 2001.
[Hofmann, 2005] M. Hofmann. Proof-theoretic approach to description logic. In *Proc. of LICS-05*. 2005
[Horrocks and Sattler, 2003] I. Horrocks and U. Sattler. Decidability of SHIQ with complex role inclusion axioms. In *Proc. IJCAI-03*, 2003.
[Horrocks et al., 2000] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *J. of the IGPL*, 8, 2000.
[Horrocks et al., 2003] I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1, 2003.
[Horrocks, 1998] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. KR'98*, 1998.
[Kazakov and de Nivelle, 2003] Y. Kazakov and H. de Nivelle. Subsumption of concepts in $\mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete. In *Proc. DL 03*. <http://CEUR-WS.org/Vol-81/>, 2003.
[Lutz, 2003] C. Lutz. NExpTime-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 5(4):669-705, 2004.
[Nebel, 1988] B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34, 1988.
[Nebel, 1990] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43, 1990.
[Rector and Horrocks, 1997] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proc. Workshop on Ontological Engineering*, 1997.
[Schild, 1991] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. IJCAI'91*, 1991.
[Spackman, 2000] K.A. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *J. of the American Medical Informatics Association*, 2000. Fall Symposium Special Issue.
[Toman and Weddell, 2005] D. Toman and G. Weddell. On reasoning about structural equality in XML: A description logic approach. *TCS*, 2005. To appear.

Analysis and Verification of Qualitative Models of Genetic Regulatory Networks: A Model-Checking Approach

Grégory Batt¹, Delphine Ropers¹, Hidde de Jong¹, Johannes Geiselmann²,
Radu Mateescu¹, Michel Page^{1,3}, Dominique Schneider²

¹INRIA Rhône-Alpes, Montbonnot, France, ²Laboratoire Adaptation et Pathogénie des
Microorganismes, Université Joseph Fourier, Grenoble, France, ³Université Pierre
Mendès France, Grenoble, France

Abstract

Methods developed for the qualitative simulation of dynamical systems have turned out to be powerful tools for studying genetic regulatory networks. A bottleneck in the application of these methods is the analysis of the simulation results. In this paper, we propose a combination of qualitative simulation and model-checking techniques to perform this task systematically and efficiently. We apply our approach to the analysis of the complex network controlling the nutritional stress response in the bacterium *Escherichia coli*.

1 Introduction

Qualitative simulation aims at making predictions of the behavior of dynamical systems in the absence of detailed, quantitative information on parameter values and functional relations [Kuipers, 1994]. The idea of qualitative simulation has attracted much interest in the context of a biological application, the analysis of *genetic regulatory networks*, that is, the networks of regulatory interactions between genes, proteins, and small molecules underlying the development and functioning of all living organisms. On the one hand, mathematical methods supported by computer tools are indispensable for the analysis of genetic regulatory networks: most networks of interest involve many genes connected through complex feedback loops, thus making an intuitive understanding of their dynamics difficult to obtain. On the other hand, numerical simulation methods are difficult to apply, because only a few networks are well-understood on the molecular level, and quantitative information on the interactions is seldom available.

We have previously developed a method for the qualitative simulation of genetic regulatory networks [de Jong *et al.*, 2004b]. This method differs from traditional approaches towards qualitative simulation in that it has been tailored to a class of piecewise-linear (PL) differential equations with favorable mathematical properties [Glass and Kauffman, 1973]. This allows it to deal with large and complex networks of regulatory interactions. The qualitative simulation method has been implemented in a publicly-available computer tool, called Genetic Network Analyzer (GNA) [de Jong *et al.*, 2003].

GNA has been used to analyze several genetic regulatory networks of biological interest (*e.g.*, [de Jong *et al.*, 2004a]).

Given a model of a genetic regulatory network, the above qualitative simulation method produces a state transition graph, consisting of qualitative states and transitions between qualitative states. The graph describes the possible qualitative behaviors of the network. When simulating large and complex genetic regulatory networks, the state transition graph usually consists of hundreds or even thousands of states, as a consequence of which the analysis of the graph by visual inspection alone becomes error-prone or even practically infeasible. In order to analyze large state transition graphs, previous work in qualitative simulation has proposed the use of *model-checking* techniques [Brajnik and Clancy, 1998; Shults and Kuipers, 1997]. These techniques allow for the efficient verification of properties of discrete transition systems [Clarke *et al.*, 1999].

The aim of this paper, extending preliminary ideas in [Batt *et al.*, 2003], is to demonstrate the effectiveness and practical applicability of the combined use of qualitative simulation and model checking for the analysis of large state transition graphs produced by the qualitative simulation of genetic regulatory networks. In particular, we show how state transition graphs can be reformulated as so-called Kripke structures on which properties of the behavior of the network, expressed in temporal logic, can be verified. In addition, we present a new implementation of GNA that connects the qualitative simulator to state-of-the-art model checkers.

We have applied the above approach to the analysis of the complex regulatory network controlling the nutritional stress response in the bacterium *Escherichia coli*. Among other results, the use of model-checking tools has allowed the characterization of cycles in a large state transition graph obtained through qualitative simulation of the network. The cycles correspond to biologically-important phenomena that are currently being experimentally tested in our laboratory.

In the next section of this paper, we briefly review the qualitative modeling and simulation of genetic regulatory networks. This will set the stage for a discussion of the combined use of qualitative simulation and model-checking techniques in the third section. The application of this approach to the analysis of the nutritional stress response network is the subject of the next section. We finish with a discussion of the approach in the context of related work.

2 Qualitative modeling and simulation of genetic regulatory networks

The dynamics of genetic regulatory networks can be modeled by a class of piecewise-linear (PL) differential equations of the following general form [Glass and Kauffman, 1973]:

$$\dot{\mathbf{x}} = \mathbf{h}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) - \mathbf{g}(\mathbf{x})\mathbf{x}, \quad (1)$$

where $\mathbf{x} = (x_1, \dots, x_n)' \in \Omega$ is a vector of cellular protein concentrations, $\mathbf{f} = (f_1, \dots, f_n)'$, $\mathbf{g} = \text{diag}(g_1, \dots, g_n)$, and $\Omega \subset \mathbb{R}_{\geq 0}^n$ is a bounded n -dimensional phase space box. The rate of change of each protein concentration x_i , $i \in \{1..n\}$, is thus defined as the difference of the rate of synthesis $f_i(\mathbf{x})$ and the rate of degradation $g_i(\mathbf{x})x_i$ of the protein.

The function $f_i : \Omega \rightarrow \mathbb{R}_{\geq 0}$ expresses how the rate of synthesis of the protein encoded by gene i depends on the concentrations $\mathbf{x}$ of the proteins in the cell. More specifically, the function f_i is defined as a sum of terms having the general form $\kappa_i^l b_i^l(\mathbf{x})$, where $\kappa_i^l > 0$ is a rate parameter, and $b_i^l : \Omega \rightarrow \{0, 1\}$ a piecewise-continuous *regulation function* defined in terms of step functions s^+ and s^- :

$$s^+(x_i, \theta_i) = \begin{cases} 1, & \text{if } x_i > \theta_i, \\ 0, & \text{if } x_i < \theta_i, \end{cases} \text{ and } s^-(x_i, \theta_i) = 1 - s^+(x_i, \theta_i),$$

where x_i is an element of the state vector $\mathbf{x}$ and θ_i a constant denoting a threshold concentration for x_i .

The function g_i expressing the regulation of protein degradation is defined analogously, except that we demand that g_i is strictly positive. Notice that with the above definitions, $\mathbf{h}$ is a *piecewise-linear (PL)* vector-valued function. An example of a simple genetic regulatory network and its PL model is shown in Figure 1(a)-(b).

The use of step functions $s^\pm(x_i, \theta_i)$ in (1) gives rise to complications, because the step functions are discontinuous at $x_i = \theta_i$, and therefore $\mathbf{h}$ is discontinuous on $\Theta = \bigcup_{i \in \{1..n\}, l_i \in \{1..p_i\}} \{\mathbf{x} \in \Omega \mid x_i = \theta_i^{l_i}\}$, the union of the threshold hyperplanes. In order to deal with this problem, following an approach widely used in control theory, we extend the differential equation (1), defined on $\Omega \setminus \Theta$, to the differential inclusion

$$\dot{\mathbf{x}} \in H(\mathbf{x}), \quad (2)$$

defined on Ω , where $H(\mathbf{x})$ is the smallest closed *hyperrectangular* set containing the set of all limit values of $\mathbf{h}(\mathbf{y})$, for $\mathbf{y} \notin \Theta$ and $\mathbf{y} \rightarrow \mathbf{x}$ [Gouzé and Sari, 2002; de Jong *et al.*, 2004b; Batt *et al.*, 2005]. A *solution* of the differential inclusion (2) on a time interval I is an absolutely-continuous vector-valued function $\xi(t)$ such that $\dot{\xi}(t) \in H(\xi(t))$ almost everywhere on I .

Formally, we define the *PL system* Σ as the triple (Ω, Θ, H) , that is, the set-valued function H given by (2), defined on the n -dimensional phase space box Ω , with Θ the union of the threshold hyperplanes. Also, we denote by Ξ_Σ the set of all solutions $\xi(t)$, on a finite or infinite time interval, of the differential inclusion (2), that reach and leave a threshold hyperplane finitely-many times.

The mathematical analysis of PL systems has shown that mere knowledge of the relative order of threshold parameters and ratios of synthesis and degradation parameters, describing the local strength of gene expression, is sufficient

for the definition of a finite partition $\mathcal{D}$ of the phase space Ω into hyperrectangular regions, called *domains*. This partition is such that the derivatives of the solutions in each domain have a unique *sign pattern* (Figure 1(b)-(d)). The definition of the sign pattern $S(\mathbf{x})$ for a point $\mathbf{x}$ in a domain D is complicated by the fact that the solution of a differential inclusion is in general not unique. In particular, if $\mathbf{x}$ lies on a threshold hyperplane, $H(\mathbf{x})$ is multi-valued, and several solutions $\xi(t)$ may pass through $\mathbf{x}$ with different derivative signs. We therefore define $S(\mathbf{x}) = \{\text{sign}(\dot{\xi}(t_x)) \mid \xi \in \Xi_\Sigma \text{ in } D, \xi(t_x) = \mathbf{x}, \text{ and } \dot{\xi}(t_x) \in H(\xi(t_x))\}$. We have proven that $S(\mathbf{x})$ is the same for every $\mathbf{x} \in D$ [Batt *et al.*, 2005].¹

Using the domain partition $\mathcal{D}$ of the phase space Ω , together with the above qualitative characterization of the dynamics in each of the domains, we can discretize the continuous dynamics. In the resulting abstract description, the state of the system is represented by a domain, and there exists a transition from a domain D to a domain D' , if and only if there exists a solution reaching D' from D , without leaving $D \cup D'$. This leads to the introduction of the so-called *qualitative transition system*, $\Sigma\text{-QTS} = (\mathcal{D}, \rightarrow, \Pi, \models_\Pi)$, where $\mathcal{D}$ is the set of all domains, $\rightarrow \subseteq \mathcal{D} \times \mathcal{D}$ is a transition relation describing all transitions between the domains, $\Pi = \{D\text{sign} = S \mid S \in 2^{\{-1,0,1\}^n}\}$ is a set of propositions describing the derivative sign patterns, and $\models_\Pi \subseteq \mathcal{D} \times \Pi$ is a satisfaction relation that associates to each domain a qualitative description of the dynamics of the system in the domain, defined by $D \models_\Pi D\text{sign} = S$ iff for all $\mathbf{x} \in D$: $S(\mathbf{x}) = S$.

The graph representation of the qualitative transition system, $G = (\mathcal{D}, \rightarrow)$, is called a *state transition graph*. The domains are also called *qualitative states*. A qualitative state D is called *instantaneous*, if no solution remains in D longer than a time instant, and *persistent*, otherwise. Moreover, it is a *qualitative equilibrium state*, if $D \models_\Pi D\text{sign} = S$, with $\mathbf{0} \in S$, since then for every $\mathbf{x} \in D$, $\xi(t) = \mathbf{x}$ is a solution. The state transition graph for the example network is shown in Figure 1(e). A path in the graph describes a possible qualitative behavior of the system (Figure 1(f)).

In [Batt *et al.*, 2005], the qualitative transition system is defined as the *discrete quotient* of a continuous transition system having the same reachability properties as the original PL system Σ , given the equivalence relation induced by the partition $\mathcal{D}$. Using standard results from hybrid systems theory [Alur *et al.*, 2000], it directly follows that the qualitative transition system $\Sigma\text{-QTS}$ is a *conservative approximation* of Σ , in the sense that if a solution ξ of the PL system Σ , defined on some time interval I , passes through the time-ordered sequence of domains $(D^0, \dots, D^m)$, then $(D^0, \dots, D^m)$ is a path in $\Sigma\text{-QTS}$. On the other hand, it may happen that no solution of Σ passes through a sequence of domains corresponding to a path in $\Sigma\text{-QTS}$.²

Simple rules exploiting the favorable mathematical properties of the class of PL models have been formulated such that the qualitative transition system can be symbolically com-

¹In previous work [de Jong *et al.*, 2004b], a coarser-grained partition of Ω is used, for which this property does not hold.

²In the terminology of qualitative simulation, the method is sound but incomplete.

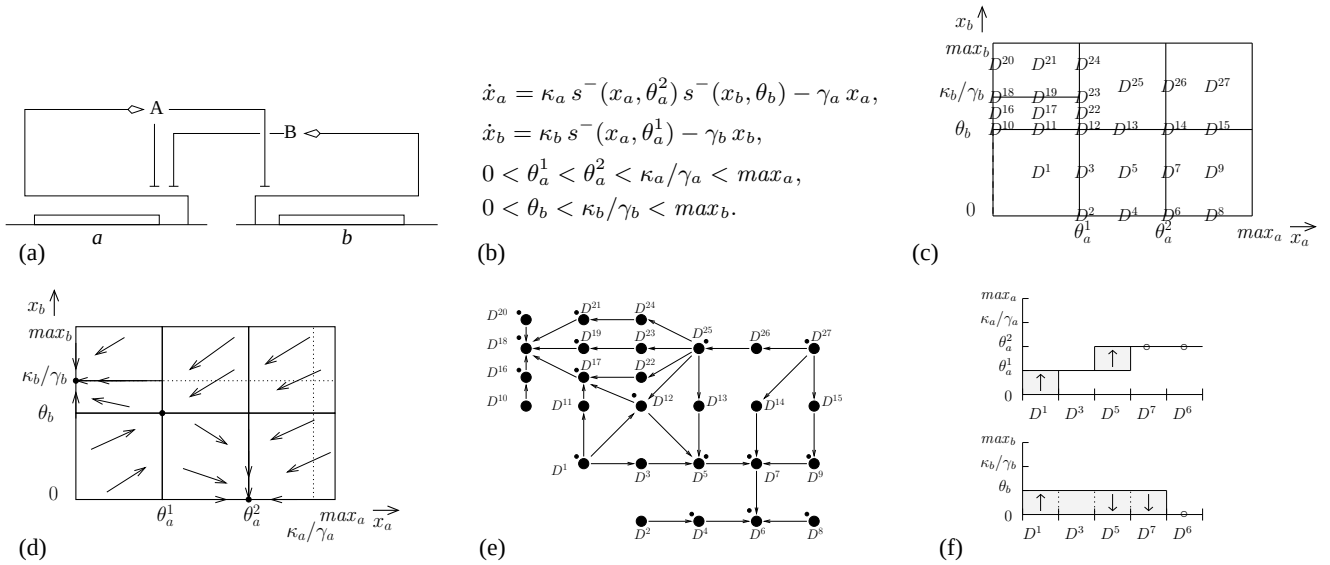


Figure 1: Qualitative simulation of a two-gene network. (a) Graphical representation (legend in Figure 3(a)). Gene b encodes protein B, which inhibits gene a expression, that is, protein A synthesis. (b) Model consisting of PL differential equations and parameter inequalities. x_a and x_b are the protein concentrations, κ_a and κ_b (γ_a and γ_b) the synthesis (degradation) rates of the proteins, and θ_a^1 , θ_a^2 , and θ_b threshold concentrations. The second differential equation states that the synthesis rate of protein B is κ_b if x_a is below θ_a^1 , and 0 if x_a is above θ_a^1 , and that its degradation rate is $\gamma_b x_b$. The parameter inequalities for protein B state that gene b may inhibit gene a expression, because when synthesized, protein B concentration tends towards the level κ_b / γ_b , for which $s^-(x_b, \theta_b) = 0$. (c) Domain partition of the phase space Ω : $\mathcal{D} = \{D^1, \dots, D^{27}\}$. (d) Sketch of the flow in Ω showing the unicity of the derivative sign pattern in each domain. Dots denote equilibrium points. (e) Graph representation of the qualitative transition system, Σ -QTS. Dots denote self-transitions. (f) Temporal evolution of the protein concentrations in the path $(D^1, \dots, D^6)$. Arrows indicate the derivative signs in persistent states ($\uparrow$ for $S = \{1\}$; $\downarrow$ for $S = \{-1\}$; $\circ$ for $S = \{0\}$).

puted from a PL model of the network and a set of parameter inequalities defining the relative order of the parameters [Batt *et al.*, 2005]. The implementation of these rules has resulted in a new version of the computer tool *Genetic Network Analyzer (GNA)*, available at <http://www-helix.inrialpes.fr/gna>.

3 Analysis of genetic regulatory networks by model checking

Models of genetic regulatory networks of biological interest may produce large state transition graphs, thus making manual analysis of dynamical properties error-prone or even practically infeasible. To address this problem, we propose to combine the qualitative modeling and simulation method outlined above with techniques for *model checking* [Clarke *et al.*, 1999]. These techniques allow for the efficient verification of properties of the behavior of discrete transition systems, expressed as formulae in some temporal logic.

Various model-checking frameworks exist, differing by their expressiveness, user-friendliness, and computational efficiency. For the sake of clarity of exposition, we focus on one particular framework in this paper, in which the discrete transition system takes the form of a *Kripke structure*, and the behavioral properties are expressed in *Computation Tree Logic (CTL)* [Clarke *et al.*, 1999]. However, we emphasize that our approach is not restricted to CTL model-checking, and allows other, more expressive temporal logics to be used as well (Section 3.3).

3.1 Translate qualitative transition system into Kripke structure

As a preliminary step, we introduce a set of *atomic propositions*, AP , to describe the state of the system. More precisely, the set of atomic propositions we use consists of simple expressions describing a protein concentration (e.g., $value\ x_i = \theta_i$, $value\ x_i < \kappa_i / \gamma_i$), the sign of its derivative (e.g., $sign\ x_i = \{1\}$), or the type of a state ($type = pers$, $type = inst$, $type = eq$, for persistent, instantaneous, and equilibrium state, respectively). The set of atomic propositions for the two-gene example is shown in Figure 2(a).

We define the *Kripke structure* over AP corresponding to Σ -QTS as Σ -KS = $(\mathcal{D}, \rightarrow, \models_{AP})$, where $\models_{AP} \subseteq \mathcal{D} \times 2^{AP}$ is the satisfaction relation associating to every domain D the atomic propositions that hold in D . The existence of at least one solution ξ on some time interval $[0, \tau]$, $\tau > 0$, with initial condition $\xi(0) = x_0$, for every $x_0 \in \Omega$, guarantees that $\rightarrow$ is total (i.e., every state has a successor), which is required for Kripke structures. The Kripke structure for the two-gene example is shown in Figure 2(b).

3.2 Express dynamical properties in temporal logic

CTL formulae are interpreted with respect to a Kripke structure. A CTL formula is built upon atomic propositions. The usual operators from propositional logic, such as negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$), and implication ($\rightarrow$), can also be used. In addition, CTL provides two types of op-

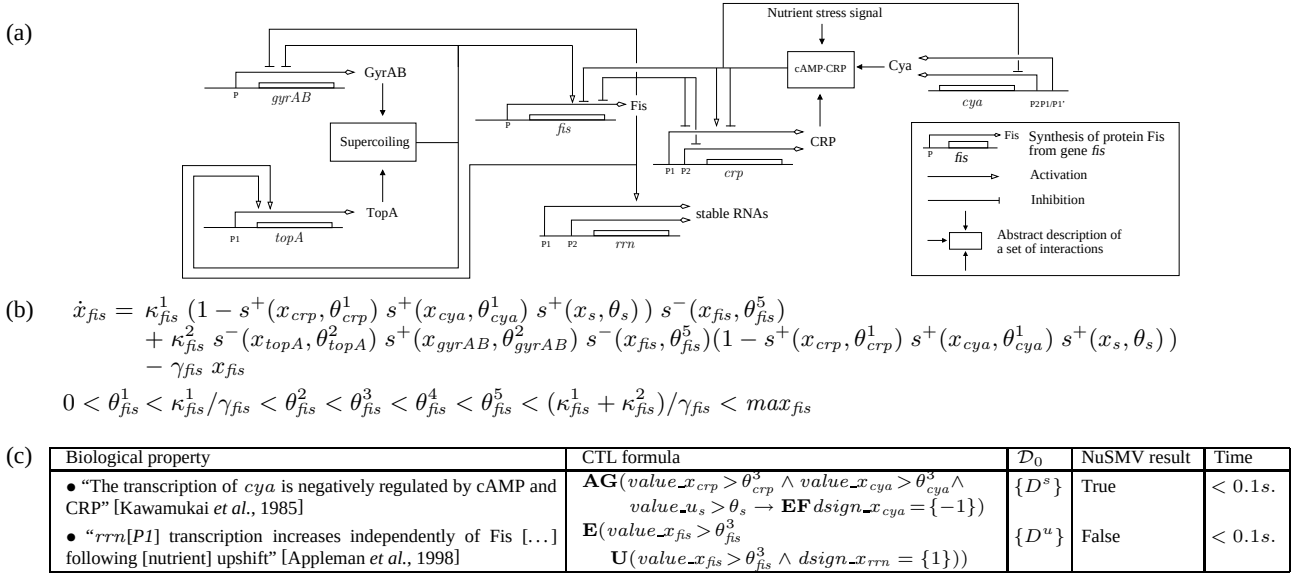


Figure 3: Network of key genes, proteins, and regulatory interactions involved in the nutritional stress response in *E. coli* [Ropers *et al.*, 2004]. (b) PL differential equation and parameter inequalities for the protein Fis. (c) Some expected properties of the network. Their translation into CTL and interpretation are not detailed here. $\mathbf{U}$ is the standard CTL operator Until [Clarke *et al.*, 1999]. D^s and D^u correspond to the initial states for nutrient starvation and nutrient upshift, respectively.

transition from exponential to stationary phase, and nutrient upshift, leading to the reentry into exponential phase. The simulation results have been analyzed by means of the model-checker NuSMV, using the export functionality of GNA. Below, we give some examples.

The simulation of a nutrient starvation has given rise to a state transition graph of 66 states (27 of which are persistent), computed in less than 1 s. on a PC (800 MHz, 256 Mb). The graph contains a single equilibrium state corresponding to stationary-phase conditions. Are the predictions obtained from the model consistent with experimental data? In [Azam *et al.*, 1999], the measured concentration of the global regulator Fis is shown to decrease and become steady in stationary phase, which is characterized by a low concentration x_{rrn} of stable RNAs:

$$\mathbf{EF}(dsign_{x_{fis}} = \{-1\} \wedge \mathbf{EF}(dsign_{x_{fis}} = \{0\} \wedge value_{x_{rrn}} < \theta_{rrn})) \quad (3)$$

The verification of (3) takes a fraction of a second to complete and shows that the observed temporal evolution of the Fis concentration is reproduced by the model.

We have also studied the reentry into exponential phase after a nutrient upshift. Using the same model as above, but starting the simulation from the qualitative state characterizing stationary-phase conditions and with the nutritional stress signal switched off, qualitative simulation results in a state transition graph of 1143 states (202 of which are persistent), generated in 1.7 s. The graph contains several strongly connected components (SCCs). One of these SCC, containing 222 states, can be reached from all states in the graph. The qualitative transition system satisfies the property

$$\mathbf{AG}(statesInSCC \rightarrow \mathbf{AG} statesInSCC), \quad (4)$$

where the predicate *statesInSCC* is satisfied by all and only states in the SCC. That is, if the system has reached this SCC,

it always remains in it. Checking this property by visual inspection is tedious and error-prone, while it takes NuSMV only 3.3 s. Further mathematical analysis has revealed that the cyclic paths correspond to solutions spiraling inwards to an equilibrium point [Ropers *et al.*, 2004]. In other words, during the reentry into stationary phase, the concentrations of some of the proteins oscillate towards a new equilibrium level. This is a surprising result, which is currently subject to experimental verification in our laboratory. We also checked a dozen of other properties, two of which are shown in Figure 3(c).

5 Discussion

We propose a combination of qualitative simulation and model-checking techniques for the analysis and verification of qualitative models of genetic regulatory networks. Model checkers help in dealing with the problem that the state transition graphs generated by qualitative simulation may become prohibitively large. They permit complex dynamical properties, whose verification defeats manual analysis capabilities, to be efficiently and reliably checked. The analysis of the genetic regulatory network composed of key genes involved in the nutritional stress response in *E. coli* illustrates the applicability of the approach.

Model-checking techniques have been used before for analyzing biological networks. Most approaches start from discrete models, such as concurrent transition systems [Chabrier-Rivier *et al.*, 2004] and Boolean networks [Bernot *et al.*, 2004]. In this paper we show that model-checking techniques can also be used for more conventional continuous models, in particular differential equation models, when using qualitative abstractions to discretize the dynamics of the system. In comparison with ideas along the same line

[Brajnik and Clancy, 1998; Shults and Kuipers, 1997], our approach is adapted to a particular class of PL differential equations with favorable mathematical properties, allowing the development of tailored algorithms that scale up well to models of large and complex genetic regulatory networks.

The model-validation approach of this paper has been illustrated in the context of CTL model checking. While CTL allows a variety of biologically-meaningful properties to be expressed, some properties fall outside its scope. For instance, in Section 4 we would have liked to be able to express the occurrence of oscillations in some of the protein concentrations after a nutrient upshift. The formula $\mathbf{EG}(p \rightarrow \mathbf{F}\neg p \wedge \neg p \rightarrow \mathbf{F}p)$ expresses this property, where p means that the concentration of some protein is above a certain threshold. Unfortunately it is not a CTL formula and it does not admit any CTL equivalent [Clarke and Draghicescu, 1988]. However, the property can be expressed in μ -calculus and evaluated using XTL, a component of the CADP toolbox [Mateescu and Garavel, 1998]. The capability of GNA to generate export files for different model checkers, allows one to take advantage from the specific strengths of each of these.

References

- [Alur *et al.*, 2000] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proc. IEEE*, 88(7):971–84, 2000.
- [Appleman *et al.*, 1998] J.A. Appleman, W. Ross, J. Salomon, and R.L. Gourse. Activation of *E. coli* rRNA transcription by Fis during a growth cycle. *J. Bacteriol.*, 180(6):1525–32, 1998.
- [Azam *et al.*, 1999] T. Ali Azam, A. Iwata., A. Nishimura, S. Ueda, and A. Ishihama. Growth phase-dependent variation in protein composition of the *E. coli* nucleoid. *J. Bacteriol.*, 181(20):6361–70, 1999.
- [Batt *et al.*, 2003] G. Batt, H. de Jong, J. Geiselmann, and M. Page. Analysis of genetic regulatory networks: A model-checking approach. *QR’03*, 31–38, 2003.
- [Batt *et al.*, 2005] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, M. Page, and D. Schneider. Qualitative analysis and verification of hybrid models of genetic regulatory networks. *HSCC’05, LNCS 3414*, 134–50. Springer, 2005.
- [Bernot *et al.*, 2004] G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. A fruitful application of formal methods to biological regulatory networks: Extending Thomas’ asynchronous logical approach with temporal logic. *J. Theor. Biol.*, 229(3):339–47, 2004.
- [Brajnik and Clancy, 1998] G. Brajnik and D.J. Clancy. Focusing qualitative simulation using temporal logic. *Ann. Math. Artif. Intell.*, 22(1-2):59–86, 1998.
- [Chabrier-Rivier *et al.*, 2004] N. Chabrier-Rivier, M. Chiverini, V. Danos, F. Fages, and V. Schächter. Modeling and querying biomolecular interaction networks. *Theor. Comput. Sci.*, 325(1):25–44, 2004.
- [Cimatti *et al.*, 2002] A. Cimatti *et al.* NuSMV2: An open-source tool for symbolic model checking. *CAV’02, LNCS 2404*, 359–64. Springer, 2002.
- [Clarke and Draghicescu, 1988] E.M. Clarke and I.A. Draghicescu. Expressibility results for linear-time and branching-time logics. *REX Workshop, LNCS 354*, 428–37. Springer, 1988.
- [Clarke *et al.*, 1999] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, 1999.
- [de Jong *et al.*, 2003] H. de Jong, J. Geiselmann, C. Hernandez, and M. Page. Genetic Network Analyzer: Qualitative simulation of genetic regulatory networks. *Bioinformatics*, 19(3):336–44, 2003.
- [de Jong *et al.*, 2004a] H. de Jong, J. Geiselmann, G. Batt, C. Hernandez, and M. Page. Qualitative simulation of the initiation of sporulation in *B. subtilis*. *Bull. Math. Biol.*, 66(2):261–99, 2004.
- [de Jong *et al.*, 2004b] H. de Jong, J.-L. Gouzé, C. Hernandez, M. Page, T. Sari, and J. Geiselmann. Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bull. Math. Biol.*, 66(2):301–40, 2004.
- [Glass and Kauffman, 1973] L. Glass and S.A. Kauffman. The logical analysis of continuous non-linear biochemical control networks. *J. Theor. Biol.*, 39(1):103–29, 1973.
- [Gouzé and Sari, 2002] J.-L. Gouzé and T. Sari. A class of piecewise-linear differential equations arising in biological models. *Dyn. Syst.*, 17(4):299–316, 2002.
- [Grumberg and Long, 1994] O. Grumberg and D.E. Long. Model checking and modular verification. *ACM Trans. Program. Lang. and Syst.*, 16(3):1512–42, 1994.
- [Huisman *et al.*, 1996] G.W. Huisman, D.A. Siegele, M.M. Zambrano, and R. Kolter. Morphological and physiological changes during stationary phase. In F.C. Neidhardt *et al.*, eds, *E. coli and Salmonella: Cellular and Molecular Biology*, 1672–82. ASM Press, 1996.
- [Kawamukai *et al.*, 1985] M. Kawamukai, J. Kishimoto, R. Utsumi, M. Himeno, T. Komano, and H. Aiba. Negative regulation of adenylate cyclase gene (*cya*) expression by cyclic AMP-cyclic AMP receptor protein in *E. coli*. *J. Bacteriol.*, 164(2):872–77, 1985.
- [Kuipers, 1994] B.J. Kuipers. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press, 1994.
- [Mateescu and Garavel, 1998] R. Mateescu and H. Garavel. XTL: A meta-language and tool for temporal logic model-checking. *STTT’98*, 33–42, 1998.
- [Mateescu and Sighireanu, 2003] R. Mateescu and M. Sighireanu. Efficient on-the-fly model-checking for regular alternation-free μ -calculus. *Sci. Comput. Program.*, 46(3):255–81, 2003.
- [Ropers *et al.*, 2004] D. Ropers, H. de Jong, M. Page, D. Schneider, and J. Geiselmann. Qualitative simulation of nutritional stress response in *E. coli*. Technical Report RR-5412, INRIA, 2004.
- [Shults and Kuipers, 1997] B. Shults and B.J. Kuipers. Proving properties of continuous systems: Qualitative simulation and temporal logic. *Artif. Intell.*, 92:91–130, 1997.

Revision of Partially Ordered Information: Axiomatization, Semantics and Iteration

Salem Benferhat and Sylvain Lagrue
CRIL-CNRS, Université d'Artois
rue Jean Souvraz
62307 Lens Cedex. France.
{benferhat,lagrue}@cril.univ-artois.fr

Odile Papini
LSIS-CNRS, Université du Sud Toulon-Var.
BP 132.
83957 La Garde Cedex France.
papini@univ-tln.fr

Abstract

This paper deals with iterated revision of partially ordered information. The first part of this paper concerns the Katsuno-Mendelzon's postulates: we first point out that these postulates are not fully satisfactory since only a class of partially ordered information can be revised. We then propose a suitable definition of faithful assignment, followed by a new set of postulates and a representation theorem. The second part of this paper investigates additional postulates dedicated to iterated revision operators of partially ordered information. Three extensions of well-known iterated belief revision operations for dealing with partially ordered information are briefly presented.

1 Introduction

Belief revision is a central topic in databases, philosophy and artificial intelligence. The revision of a database consists of the insertion of some new information while preserving consistency.

This process has been discussed in different frameworks (probability theory, Spohn's ordinal conditional functions, Grove's systems of spheres, possibility theory, etc.) and from different points of view (axiomatization [Alchourrón *et al.*, 1985; Katsuno and Mendelzon, 1991], computational issue, etc.). The proposal of Alchourrón, Gärdenfors and Makinson (AGM) [Alchourrón *et al.*, 1985] represents one of the most influential work. It concerns revision operations that revise a theory (representing agent's current beliefs) with a formula (representing new pieces of information). The AGM framework consists in a set of natural postulates that any revision operation should satisfy.

An important issue is that any revision operator satisfying AGM's postulates is implicitly based on a priority ordering. This result suggests that an agent's epistemic state, denoted by Ψ , is something more complex than a simple representation of agent's current beliefs (called belief set and denoted by $Bel(\Psi)$). It also contains strategies to revise his beliefs. The AGM's postulates have been largely studied and/or adapted in the literature. For instance, additional postulates have been proposed to manage the concept of iterated belief change in [Darwiche and Pearl, 1997] (see also [Rott, 2001; Lehmann, 1995; Nayak, 1994; Konieczny and Pino Pérez,

2000]). More generally, iterated revision is applied to an epistemic state, instead of a belief set, and the result is also an epistemic state.

Epistemic states are often represented by total pre-orders on possible worlds (or interpretations), denoted by $\leq_\Psi$. The set of models of $Bel(\Psi)$ are identified to be minimal (i.e., preferred) elements with respect to $\leq_\Psi$.

There is another issue in belief revision and which has been less investigated in the literature: the iterated revision of epistemic states represented by partial pre-orders (instead of total pre-orders) [Katsuno and Mendelzon, 1991]. Partial pre-orders offer more flexibility than total pre-orders to represent incomplete knowledge.

Katsuno and Mendelzon proposed in section 5 of [Katsuno and Mendelzon, 1991], a set of postulates and a representation theorem that characterize revision operations based on partial pre-orders. Unfortunately, the proposed approach is not satisfactory since only one class of partial pre-orders can be revised. For instance, assume that we have a partial pre-order $\preceq_\Psi$ on four possible interpretations $\{\omega_0, \omega_1, \omega_2, \omega_3\}$, such that ω_0 is "strictly preferred" to ω_1 and ω_1 is "strictly preferred" to ω_2 but ω_3 is incomparable with ω_0, ω_1 and ω_2 (cf. figure 1, for sake of clarity, reflexivity and transitivity are not represented). Clearly, ω_0 and ω_3 characterize (are models of) agent's current beliefs $Bel(\Psi)$ (no interpretation is strictly preferred to ω_0 or to ω_3), but $\preceq_\Psi$ is not a faithful assignment in the sense of [Katsuno and Mendelzon, 1991].

More generally, there is no consistent propositional formula that admits $\preceq_\Psi$ of Figure 1 as a faithful assignment. In

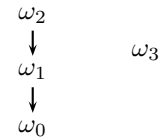


Figure 1: Example of $\preceq_\Psi$ which is not a faithful assignment.

fact, faithful assignment requires that each model of agent's current beliefs (namely $Bel(\Psi)$) should be strictly preferred to each counter-model of agent's current beliefs. This requirement is appropriate for total pre-orders, but is too strong for partial pre-orders.

This paper investigates iterated belief revision of partially ordered information. Its main contributions can be summa-

ized as follows:

- We propose an alternative definition of faithful assignment and an alternative set of natural postulates for revising partially ordered information.
- We provide a representation theorem for iterated revision operators satisfying this new set of postulates.
- We investigate additional postulates dedicated to iterated belief revision. In particular, we discuss the extension of Darwiche and Pearl's postulates, Boutilier's postulate and we propose two alternative postulates for capturing other forms of iterated belief revision.
- Lastly, we illustrate our results with three well-known operators (revision with memory, possibilistic revision and natural belief revision) for revising partially ordered information.

The rest of this paper is organized as follows. Section 2 presents the Katsuno and Mendelzon's postulates for belief revision. Section 3 contains a modification of these postulates. Section 4 characterizes Darwiche and Pearl's postulates and Boutilier's postulate for iterated belief revision with partially ordered information. Lastly, Section 5 proposes two alternative additional postulates and provides three examples of operators that satisfy the proposed postulates.

2 Notations and KM postulates

2.1 Notations

In this paper we use propositional calculus, denoted by $\mathcal{L}$, as knowledge representation language with usual connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\equiv$ (logical equivalence). The lower case letters a , b , c , $\dots$, are used to denote propositional variables and lower case Greek letters φ , ψ , $\dots$, are used to denote formulas. We denote by Ω the set of interpretations of $\mathcal{L}$ and by $Mod(\psi)$ the set of models of a formula ψ , that is $Mod(\psi) = \{\omega \in \Omega, \omega \models \psi\}$. Upper case Greek letters Ψ , Φ , $\dots$, are used to denote epistemic states.

A partial pre-order, denoted by $\preceq$, on a set A is a reflexive and transitive binary relation. Let $x, y \in A$, we define the equality $x = y$ iff $x \preceq y$ and $y \preceq x$. The corresponding strict partial pre-order associated with $\preceq$, denoted by $\prec$, is defined as usual: $x \prec y$ iff $x \preceq y$ holds but $x \preceq y$ does not hold. We denote by $\sim$ the incomparability relation, namely $x \sim y$ iff neither $x \preceq y$ nor $y \preceq x$ holds.

Given $\preceq$ on a set A , the set of the minimal elements of A , denoted by $Min(A, \preceq)$, is defined by $Min(A, \preceq) = \{x : x \in A, \nexists y \in A, y \prec x\}$.

2.2 The KM postulates for belief revision

Katsuno and Mendelzon proposed in [Katsuno and Mendelzon, 1991] a reformalization of the well-known AGM postulates [Alchourrón *et al.*, 1985]. In this framework, the epistemic state of an agent is represented by a formula ψ . A revision operator $\circ$ is a function from a formula ψ and produces a new formula denoted by $\psi \circ \mu$. The operator $\circ$ should satisfy the following postulates.

- (R_1) $\psi \circ \mu \models \mu$,
- (R_2) if $\psi \wedge \mu$ is satisfiable, then $\psi \circ \mu \equiv \psi \wedge \mu$,

(R_3) if μ is satisfiable, then $\psi \circ \mu$ is satisfiable,

(R_4) if $\psi_1 \equiv \psi_2$ and $\mu_1 \equiv \mu_2$ then $\psi_1 \circ \mu_1 \equiv \psi_2 \circ \mu_2$,

(R_5) $(\psi \circ \mu) \wedge \varphi \models \psi \circ (\mu \wedge \varphi)$,

(R_6) if $(\psi \circ \mu) \wedge \varphi$ is satisfiable, then $\psi \circ (\mu \wedge \varphi) \models (\psi \circ \mu) \wedge \varphi$.

An important result is that any revision operator that satisfies (R_1) – (R_6) is implicitly based a total pre-order $\leq_\psi$ on Ω .

In order to deal with partial pre-orders, Katsuno and Mendelzon proposed in section 5 of [Katsuno and Mendelzon, 1991] to replace (R_6) by the two following postulates:

(R_7) if $\psi \circ \mu_1 \models \mu_2$ and $\psi \circ \mu_2 \models \mu_1$ then $\psi \circ \mu_1 \equiv \psi \circ \mu_2$,

(R_8) $(\psi \circ \mu_1) \wedge (\psi \circ \mu_2) \models \psi \circ (\mu_1 \vee \mu_2)$.

The following representation theorem has been proposed with these postulates for partial pre-orders.

Theorem 1 ([Katsuno and Mendelzon, 1991]) A revision operator $\circ$ satisfies the postulates (R_1) to (R_5), (R_7) and (R_8) iff there exists a faithful assignment that maps ψ to a partial pre-order $\preceq_\psi$ such that: $Mod(\psi \circ \mu) = Min(Mod(\mu), \preceq_\psi)$.

Where the faithful assignment is defined as follows.

Definition 1 (faithful assignment) Let ψ be a propositional formula. A partial pre-order on Ω , associated with ψ , denoted by $\preceq_\psi$, is said to be a faithful assignment if the three following conditions hold :

- (1) if $\omega, \omega' \in Mod(\psi)$ then $\omega <_\psi \omega'$ does not hold,
- (2) if $\omega \in Mod(\psi)$ and $\omega' \notin Mod(\psi)$, then $\omega <_\psi \omega'$ holds,
- (3) if $\psi \equiv \varphi$, then $\preceq_\psi = \preceq_\varphi$.

In the following, by KM postulates, we refer to (R_1) – (R_5), (R_7) and (R_8).

3 KM postulates revisited

In the introduction, we pointed out that KM postulates are not fully satisfactory for iterated belief revision of partially ordered information. This section justifies this statement and proposes a revisited set of KM postulates.

3.1 Faithful assignment revisited

This subsection proposes a modified definition of faithful assignment. The one given by definition 1 is not appropriate. Indeed, if $\preceq_\psi$ is a partial pre-order on the set of interpretations Ω representing agent's epistemic state, and if we define $Mod(Bel(\Psi)) = Min(\Omega, \preceq_\Psi)$ to be the current agent's beliefs, then there is no guarantee that $\preceq_\Psi$ will be a faithful assignment.

For instance, the partial pre-order given by figure 1 is not a faithful assignment, since condition (2) of Definition 1 is not satisfied. Indeed $Mod(Bel(\Psi)) = \{\omega_0, \omega_3\}$ and $\omega_3 \prec_\Psi \omega_2$ does not hold. Of course, one may simply suggest to enforce minimal elements of $\preceq_\Psi$ to be equal. In our example, we enforce ω_0 to be equal to ω_3 (as it is illustrated by Figure 2).

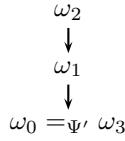


Figure 2: Example of $\preceq_{\Psi'}$.

But this is not satisfactory. Indeed, if $\preceq_{\Psi'}$ denotes the partial pre-order obtained from $\preceq_{\Psi}$ by enforcing minimal elements of $\preceq_{\Psi}$ to be equal then, in general, $Bel(\Psi' \circ \mu) \neq Bel(\Psi \circ \mu)$. In our example, if we consider a new piece of information μ such that $Mod(\mu) = \{\omega_1, \omega_3\}$, then, it can easily be checked that $Bel(\Psi \circ \mu) \neq Bel(\Psi' \circ \mu)$. Indeed $Mod(Bel(\Psi \circ \mu)) = Min(\{\omega_1, \omega_3\}, \preceq_{\Psi}) = \{\omega_1, \omega_3\}$ and $Mod(Bel(\Psi' \circ \mu)) = Min(\{\omega_1, \omega_3\}, \preceq_{\Psi'}) = \{\omega_3\}$, where $\preceq_{\Psi}$ and $\preceq_{\Psi'}$ are respectively represented by Figure 1 and Figure 2.

The problem with Definition 1 is the condition (2), which requires that each model of agent's current beliefs is preferred to all counter-models of agent's current beliefs. We propose to weaken this condition by only requiring that each counter-model of agent's current beliefs is strictly less preferred than at least one model of agent's current beliefs.

In order to deal with iterated belief revision, which applies to epistemic states instead of formulas, we also modify condition (3) exactly as it is proposed in [Darwiche and Pearl, 1997]. These two changes lead to the following modified definition of faithful assignment.

Definition 2 (P-faithful assignment) *Let Ψ be an epistemic state. A partial pre-order on Ω , associated to Ψ and denoted by $\preceq_{\Psi}$ is said to be a P-faithful assignment if the three following conditions hold :*

- (1) if $\omega, \omega' \models Bel(\Psi)$ then $\omega \prec_{\Psi} \omega'$ does not hold,
- (2) if $\omega' \not\models Bel(\Psi)$, then there exists ω such that $\omega \models Bel(\Psi)$ and $\omega \prec_{\Psi} \omega'$,
- (3) if $\Psi = \Phi$, then $\preceq_{\Psi} = \preceq_{\Phi}$.

Clearly, if $\preceq_{\Psi}$ is a faithful assignment, then it is also a P-faithful assignment. Note that if $\preceq_{\Psi}$ is a total pre-order, then this definition is equivalent to the definition of a faithful assignment of Darwiche and Pearl [Darwiche and Pearl, 1997].

3.2 Revisited postulates for partially ordered information

Let us now present the "revision" of KM postulates. In the whole paper, we assume that the new information μ is consistent and the initial beliefs $Mod(Bel(\Psi))$ is not empty. Basically, there is a need of two changes. The first change is the same as the one proposed by Darwiche and Pearl in [Darwiche and Pearl, 1997] (see also [Friedman and Halpern, 1999]). It consists in a simple adaptation of KM postulates in order to make belief revision operation a function on epistemic states instead on set of beliefs. More precisely, we propose to replace postulate (R_4) by the one proposed in [Darwiche and Pearl, 1997].

(P_4) if $\Psi_1 = \Psi_2$ and $\mu_1 \equiv \mu_2$ then $Bel(\Psi_1 \circ \mu_1) \equiv Bel(\Psi_2 \circ \mu_2)$.

Postulate (P_4) requires the equality between epistemic states while R_4 requires the equivalence between beliefs sets.

The second change concerns postulate (R_2) . We argue that (R_2) is too strong when dealing with partial pre-orders.

Let us consider again the example represented by Figure 1. Let μ be such that $Mod(\mu) = \{\omega_1, \omega_3\}$. Since $Min(Mod(\mu), \preceq_{\Psi}) = \{\omega_1, \omega_3\}$, we expect to have $Bel(\Psi \circ \mu) \equiv \{\omega_1, \omega_3\}$. But R_2 leads to have $Bel(\Psi \circ \mu) \equiv \{\omega_3\}$. Indeed, (R_2) says that if $Bel(\Psi) \wedge \mu$ is consistent, then $Bel(\Psi) \circ \mu \equiv Bel(\Psi) \wedge \mu$. This equivalence makes sense if for each ω, ω' models of $Bel(\Psi)$, $\omega =_{\Psi} \omega'$ holds. However, when models of $Bel(\Psi)$ may be incomparable, then (R_2) becomes questionable.

We suggest to replace R_2 by the following postulate. It concerns the situation where the new piece of information is a tautology. In this case, agent's current beliefs should not change. Namely:

(P_2) $Bel(\Psi \circ \top) \equiv Bel(\Psi)$.

3.3 Representation theorem

The new set of postulates that we propose for characterizing the revision of partially ordered beliefs are the following ones.

- (P_1) $Bel(\Psi \circ \mu) \models \mu$,
- (P_2) $Bel(\Psi \circ \top) \equiv Bel(\Psi)$,
- (P_3) if μ is satisfiable, then $Bel(\Psi \circ \mu)$ is satisfiable,
- (P_4) if $\Psi_1 = \Psi_2$ and $\mu_1 \equiv \mu_2$ then $Bel(\Psi_1 \circ \mu_1) \equiv Bel(\Psi_2 \circ \mu_2)$,
- (P_5) $Bel(\Psi \circ \mu) \wedge \varphi \models Bel(\Psi \circ (\mu \wedge \varphi))$,
- (P_6) if $Bel(\Psi \circ \mu_1) \models \mu_2$ and $Bel(\Psi \circ \mu_2) \models \mu_1$ then $Bel(\Psi \circ \mu_1) \equiv Bel(\Psi \circ \mu_2)$,
- (P_7) $Bel(\Psi \circ \mu_1) \wedge Bel(\Psi \circ \mu_2) \models Bel(\Psi \circ (\mu_1 \vee \mu_2))$.

The postulates (P_1) , (P_3) , (P_4) , (P_5) , correspond respectively to the postulates (R_1) , (R_3) , (R_4) , (R_5) . The postulates (P_6) and (P_7) are respectively a reformalization of the postulates (R_7) and (R_8) proposed in [Katsuno and Mendelzon, 1991] for epistemic states.

Two weakenings of (R_2) can be derived from this set of postulates:

(P'_2) $Bel(\Psi) \wedge \mu \models Bel(\Psi \circ \mu)$,

(P_{2w}) if $Bel(\Psi) \models \mu$ then $Bel(\Psi \circ \mu) \equiv Bel(\Psi) \wedge \mu$.

The first one means that joint of initial beliefs $Bel(\Psi)$ and the new information are contained in the set of new belief's models. The second one means that if we revise with some pieces of information that can be inferred from agent's current beliefs, then the agent's beliefs should not change.

Given this set of postulates and the definition of P-faithful assignment, we provide the following representation theorem:

Theorem 2 A revision operator $\circ$ satisfies the postulates (P_1) to (P_7) iff there exists a P-faithful assignment that maps each Ψ to a partial pre-order $\preceq_\Psi$ such that $\text{Mod}(\text{Bel}(\Psi \circ \mu)) = \text{Min}(\text{Mod}(\mu), \preceq_\Psi)$.

4 Characterization of iterated revision postulates

Several authors (e.g. [Darwiche and Pearl, 1997]) argue that, in order to have a genuine iterated belief revision operator, we should augment basic AGM postulates with some additional postulates. Indeed, these basic postulates constrain the relationships between $\text{Bel}(\Psi)$ and $\text{Bel}(\Psi \circ \mu)$. The aim of these additional postulates is to provide more constraints between Ψ and $\Psi \circ \mu$ by, for instance, expliciting the relationships between $\preceq_\Psi$ and $\preceq_{\Psi \circ \mu}$. Given a formula μ , there are three questions that need to be addressed:

- How the order between models of μ does evolve?
- How the order between counter-models of μ does evolve?
- How the order between models and counter-models of μ does evolve?

Darwiche and Pearl [Darwiche and Pearl, 1997] gave four natural postulates for iterated belief revision that provide answers for these questions.

- (C_1) if $\alpha \models \mu$, then $\text{Bel}((\Psi \circ \mu) \circ \alpha) \equiv \text{Bel}(\Psi \circ \alpha)$,
- (C_2) if $\alpha \models \neg\mu$, then $\text{Bel}((\Psi \circ \mu) \circ \alpha) \equiv \text{Bel}(\Psi \circ \alpha)$,
- (C_3) if $\text{Bel}(\Psi \circ \alpha) \models \mu$, then $\text{Bel}((\Psi \circ \mu) \circ \alpha) \models \mu$,
- (C_4) if $\text{Bel}(\Psi \circ \alpha) \not\models \neg\mu$, then $\text{Bel}((\Psi \circ \mu) \circ \alpha) \not\models \neg\mu$.

Another postulate, which has also been discussed in the literature, is the one which characterizes Boutilier's natural belief revision [Boutilier, 1993] (and hinted in [Spohn, 1988]) for totally ordered beliefs:

- (C_B) If $\text{Bel}(\Psi \circ \mu) \models \neg\alpha$ then $\text{Bel}((\Psi \circ \mu) \circ \alpha) \equiv \text{Bel}(\Psi \circ \alpha)$.

This section provides a characterization of the Darwiche and Pearl's postulates and the Boutilier's postulate, when epistemic states are represented by partial pre-orders.

In the following, we restrict ourselves to iterated belief revision operators that satisfy $(P_1) - (P_7)$. Let $\circ$ be a such revision operator and $\preceq_\Psi$ be its associated P-faithful assignment. In order to characterize $(C_1) - (C_4)$ and (C_B) , we define five properties on relationships between $\preceq_\Psi$ and $\preceq_{\Psi \circ \mu}$.

- (Ip_1) if $\omega \models \mu$ and $\omega' \models \mu$ then $\omega \prec_\Psi \omega'$ iff $\omega \prec_{\Psi \circ \mu} \omega'$,
- (Ip_2) if $\omega \not\models \mu$ and $\omega' \not\models \mu$ then $\omega \prec_\Psi \omega'$ iff $\omega \prec_{\Psi \circ \mu} \omega'$,
- (Ip_3) if $\omega \models \mu$ and $\omega' \not\models \mu$ then $\omega \prec_\Psi \omega'$ implies $\omega \prec_{\Psi \circ \mu} \omega'$,
- (Ip_4) if $\omega \models \mu$ and $\omega' \not\models \mu$ then $\omega \preceq_\Psi \omega'$ implies $\omega \preceq_{\Psi \circ \mu} \omega'$,
- (Ip_5) if $\omega, \omega' \not\models \text{Bel}(\Psi \circ \mu)$ then $\omega \prec_\Psi \omega'$ implies $\omega \prec_{\Psi \circ \mu} \omega'$.

(Ip_1) (resp. (Ip_2)) means that the strict order between models (resp. counter-models) of the new piece of information μ is preserved. (Ip_3) means that if a model of μ is strictly preferred to a counter-models then this strict preference is preserved. (Ip_4) is similar with (Ip_3) for non-strict preference. Finally, (Ip_5) means that the strict order between the counter-models of the agent's current beliefs is preserved.

The following representation theorem provides a characterization of $(C_1) - (C_4)$.

Theorem 3 Let $\circ$ be a revision operator satisfying the postulates $(P_1) - (P_7)$. Then:

- (1) $\circ$ satisfies (C_1) iff its associated faithful assignment satisfies (Ip_1) ,
- (2) $\circ$ satisfies (C_2) iff its associated faithful assignment satisfies (Ip_2) ,
- (3) $\circ$ satisfies (C_3) iff its associated faithful assignment satisfies (Ip_3) ,

Note that only the characterization of (C_3) is identical to the one provided by Darwiche and Pearl [Darwiche and Pearl, 1997]. The others are different.

More precisely, there are two main differences with the DP representation theorem. First, we have only

- (4) if the faithful assignment associated with $\circ$ satisfies (Ip_4) then $\circ$ satisfies (C_4) .

The converse does not hold.

The second main difference concerns (C_1) and (C_2) . Indeed, Darwiche and Pearl have shown that when $\preceq_\Psi$ is a total pre-order, then (C_1) (resp. (C_2)) guarantees that $\preceq_\Psi$ and $\preceq_{\Psi \circ \mu}$ to be identical on their subdomain $\text{Mod}(\mu) \times \text{Mod}(\mu)$ (resp. $\text{Mod}(\neg\mu) \times \text{Mod}(\neg\mu)$). However, when $\preceq_\Psi$ is a partial pre-order, then only the strict relations $\prec_\Psi$ and $\prec_{\Psi \circ \mu}$ are identical. Namely, it may happen that $\omega =_\Psi \omega'$ and $\omega \sim_{\Psi \circ \mu} \omega'$ both hold for revision operator satisfying $(P_1) - (P_7)$ and (C_1) (resp. (C_2)).

A similar remark also holds for Boutilier's postulate (C_B) .

Theorem 4 Let $\circ$ be a revision operator satisfying the postulates $(P_1) - (P_7)$. Then:

- $\circ$ satisfies (C_B) iff $\circ$ satisfies (Ip_5) .

When $\preceq_\Psi$ is a total pre-order, then $\prec_{\Psi \circ \mu}$ is uniquely defined for satisfying (C_B) . Theorem 4 shows that the unicity of the result is no longer valid and hence $\preceq_{\Psi \circ \mu}$ is not uniquely defined. Again, only the strict relation associated with $\preceq_{\Psi \circ \mu}$ is uniquely defined.

An example of revision operator satisfying (C_B) is an extension to "natural revision" for partial pre-orders:

Definition 3 Let Ψ be an epistemic state and μ be a propositional formula, the revised epistemic state $\Psi \circ_n \mu$ corresponds to the following partial pre-order:

- if $\omega, \omega' \in \text{Min}(\text{Mod}(\mu), \preceq_\Psi)$ then $\omega \preceq_{\Psi \circ_n \mu} \omega'$ iff $\omega \preceq_\Psi \omega'$,
- if $\omega, \omega' \notin \text{Min}(\text{Mod}(\mu), \preceq_\Psi)$ then $\omega \preceq_{\Psi \circ_n \mu} \omega'$ iff $\omega \preceq_\Psi \omega'$,

- if $\omega \in \text{Min}(\text{Mod}(\mu), \preceq_\Psi)$ and $\omega' \notin \text{Min}(\text{Mod}(\mu), \preceq_\Psi)$ then $\omega \prec_{\Psi \circ_n \mu} \omega'$.

The first condition imposes that the order between the preferred models of μ is preserved. The second condition preserves the ordering between non-preferred models of μ . Finally, the third condition imposes that any minimal models of μ is preferred to any other model.

Example 1 Let us consider the epistemic state Ψ represented by Figure 3, where $\omega_0 = \{\neg a, \neg b\}$, $\omega_1 = \{a, \neg b\}$, $\omega_2 = \{\neg a, b\}$, $\omega_3 = \{a, b\}$. Let μ be a new piece of information such that $\text{Mod}(\mu) = \{\omega_1, \omega_3\}$. Then the revision of Ψ by the new piece of information μ is represented by the partial pre-order $\preceq_{\Psi \circ_n \mu}$ described by Figure 4.

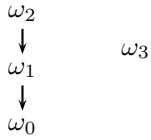


Figure 3: $\preceq_\Psi$

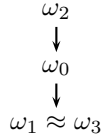


Figure 4: $\preceq_{\Psi \circ_n \mu}$

According to this definition it can be checked that the revision operator $\circ_n$ provided by definition 3 satisfies the postulates $(P_1) - (P_7)$ and (C_B) . Moreover, it can be noticed that $\circ_n$ does not satisfy (R_2) since $\text{Mod}(\text{Bel}(\Psi)) = \{\omega_0, \omega_3\}$, $\text{Mod}(\text{Bel}(\Psi) \circ_n \mu) = \{\omega_1, \omega_3\}$ and $\text{Mod}(\text{Bel}(\Psi) \wedge \mu) = \{\omega_3\}$.

5 Two alternative postulates

Among additional postulates $(C_1) - (C_4)$ proposed in [Darwiche and Pearl, 1997] for iterated belief revision, (C_1) seems to be very acceptable. However it is less obvious with (C_2) (for more detailed discussion, see, for instance, [Lehmann, 1995; Friedman and Halpern, 1999]). Moreover, (C_3) may appear to be too cautious. We now propose to analyze two alternative postulates which capture two other forms of iterated belief revision.

The first one departs from the idea of (C_2) , and it is close to the idea of conditioning with a completely sure information used in uncertainty theories. In probability theory (and similarly in possibility theory), if we revise a probability distribution by some completely sure formula, then all counter-models of these formulas are considered as impossible. This properties is captured by the following postulate.

$$(C_{\text{cond}}) \text{ Bel}((\Psi \circ \mu) \circ \neg \mu) \equiv \neg \mu.$$

It expresses that if Ψ is revised by μ , and again by its contrary $\neg \mu$, then piece of initial beliefs are ignored, and only last pieces of information are retained. The characterization of this postulate is as follows.

Theorem 5 A revision operator $\circ$ satisfies the postulates $(P_1) - (P_7)$ and (C_{cond}) iff its corresponding P -faithful assignment leads to $\preceq_{\Psi \circ \mu}$, which is such that if $\omega \models \neg \mu$ and $\omega' \models \neg \mu$ then neither $\omega \prec_{\Psi \circ \mu} \omega'$, nor $\omega' \prec_{\Psi \circ \mu} \omega$ holds.

We illustrate this postulate by the following definition of the extension of qualitative possibilistic revision [Dubois and Prade, 1992].

Definition 4 Let Ψ be an epistemic state such that its corresponding P -faithful assignment leads to $\preceq_\Psi$. Let μ be a propositional formula, The revised epistemic state $\Psi \circ_\pi \mu$ is defined by:

- if $\omega, \omega' \in \text{Mod}(\mu)$ then $\omega \preceq_{\Psi \circ_\pi \mu} \omega'$ iff $\omega \preceq_\Psi \omega'$,
- if $\omega, \omega' \notin \text{Mod}(\mu)$ then $\omega =_{\Psi \circ_\pi \mu} \omega'$,
- if $\omega \in \text{Mod}(\mu)$ and $\omega' \notin \text{Mod}(\mu)$ then $\omega \prec_{\Psi \circ_\pi \mu} \omega'$.

Namely, $\preceq_\Psi$ and $\preceq_{\Psi \circ_\pi \mu}$ are identical on subdomain $\text{Mod}(\mu) \times \text{Mod}(\mu)$. Each models of μ are preferred to each counter-models of μ . And lastly, the counter-models of μ are considered equally preferred.

Example 2 Let us again consider the epistemic state Ψ represented by Figure 3. Let μ be a new information such that $\text{Mod}(\mu) = \{\omega_1, \omega_3\}$. Then, the revision of Ψ by the new information μ is represented by the partial pre-order $\preceq_{\Psi \circ_\pi \mu}$ described by figure 5

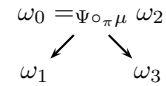


Figure 5: $\preceq_{\Psi \circ_\pi \mu}$

According to this definition it can be checked that the revision operator $\circ_\pi$ provided by definition 4 satisfies the postulates $(P_1) - (P_7)$, (C_1) , $(C_3) - (C_4)$ and (C_{cond}) .

The second postulate concerns relationships between models and counter-models of the new piece of information μ . The postulate (C_3) only requires if some strict preference in $\preceq_\Psi$ holds, then this preference should also hold in $\preceq_{\Psi \circ \mu}$. We hence propose a stronger version of (C_3) , called (C_{lex}) , and characterized by:

$$(C_{\text{lex}}) \text{ If } \alpha \text{ is not contradictory, } \alpha \models \mu \text{ and } \beta \models \neg \mu \text{ then } \text{Bel}((\Psi \circ \mu) \circ (\alpha \vee \beta)) \equiv \text{Bel}(\Psi \circ \alpha).$$

Note also that (C_{lex}) implies (C_1) , by replacing β by the contradiction $\perp$.

Theorem 6 A revision operator $\circ$ satisfies the postulates $(P_1) - (P_7)$ and (C_{lex}) iff its associated P -faithful assignments leading to $\preceq_\Psi$ and $\preceq_{\Psi \circ \mu}$ are such that

- if $\omega, \omega' \models \mu$ then $\omega \prec_\Psi \omega'$ iff $\omega \prec_{\Psi \circ \mu} \omega'$,
- if $\omega \models \mu$ and $\omega' \not\models \mu$ then $\omega \prec_{\Psi \circ \mu} \omega'$.

Revision operator $\circ$ satisfying (C_{lex}) constraints each models to be strictly preferred to all counter-models of the

new piece of information μ . Moreover, it preserves the order between the models of μ . An example of revision operator satisfying (C_{lex}) is the revision with memory proposed in [Papini, 2001] and hinted by Spohn [Spohn, 1988] (see also [Nayak, 1994; Konieczny and Pino Pérez, 2000]).

Definition 5 Let Ψ be an epistemic state and μ be a propositional formula, the revised epistemic state $\Psi \circ_{\triangleright} \mu$ corresponds to the following partial pre-order:

- if $\omega, \omega' \in \text{Mod}(\mu)$ then $\omega \preceq_{\Psi \circ_{\triangleright} \mu} \omega'$ iff $\omega \preceq_{\Psi} \omega'$,
- if $\omega, \omega' \notin \text{Mod}(\mu)$ then $\omega \preceq_{\Psi \circ_{\triangleright} \mu} \omega'$ iff $\omega \preceq_{\Psi} \omega'$,
- if $\omega \in \text{Mod}(\mu)$ and $\omega' \notin \text{Mod}(\mu)$ then $\omega \prec_{\Psi \circ_{\triangleright} \mu} \omega'$.

Example 3 Let us again consider the epistemic state Ψ represented by Figure 3. Let μ be a new information such that $\mu = a$. We have $\text{Mod}(\mu) = \{\omega_1, \omega_3\}$. Then, the revision of Ψ by the new information μ is represented by the partial pre-order $\preceq_{\Psi \circ_{\triangleright} \mu}$ described by figure 6

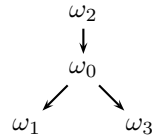


Figure 6: $\preceq_{\Psi \circ_{\triangleright} \mu}$

6 Conclusion

This paper has first proposed a new definition of faithful assignment suitable for revising partially ordered information. We then propose a revised version of Katsuno and Mendelzon's postulates. A representation theorem has been proposed for any revision operator satisfying this new set of postulates.

The paper also proposed a characterization of Darwiche and Pearl's postulates and a characterization of Boutilier's postulate when information is partially ordered. Two new postulates with their representation theorem have been proposed for capturing two other forms of belief revision.

When dealing with partial pre-orders, we have shown that revision operators satisfying iterated revision postulates such C_1 does not allow to guarantee that $\preceq_{\Psi}$ and $\preceq_{\Psi \circ_{\triangleright} \mu}$ to be identical on some subdomains (for instance on $\text{Mod}(\mu) \times \text{Mod}(\mu)$ if C_1 is satisfied). A future work will be to investigate additional postulates that preserve equality on some subdomains. For instance, it would be interesting to consider a stronger version of P_2 , by requiring equality between epistemic states instead of logical equivalence between belief sets.

Even if results of this paper are oriented belief revision, it can be interesting to discuss this work in the framework of updating [Katsuno and Mendelzon, 1992] (since updating has also questioned (R_2), even if representation theorems are clearly different).

Another future work is to extend results to this paper to the revision of a partial pre-order by a partial pre-order and extends, for instance, postulate proposed in [Benferhat et al., 2000].

References

- [Alchourrón et al., 1985] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic*, 50:510–530, 1985.
- [Benferhat et al., 2000] Salem Benferhat, Sébastien Konieczny, Odile Papini, and Ramón A. Pino Pérez. Iterated revision by epistemic states : Axioms, semantics and syntax. In *Proceedings of European Conference on Artificial Intelligence (ECAI'2000)*, pages 13–17, 2000.
- [Boutilier, 1993] Craig Boutilier. Revision sequences and nested conditionals. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 519–531, 1993.
- [Darwiche and Pearl, 1997] Adnan Darwiche and Judea Pearl. On the logic of iterated belief revision. *Artificial intelligence*, 89:1–29, 1997.
- [Dubois and Prade, 1992] Didier Dubois and Henri Prade. Belief change and possibility theory. In Peter Gärdenfors, editor, *Belief Revision*, pages 142–182. Cambridge University Press, 1992.
- [Friedman and Halpern, 1999] Nir Friedman and Joseph Y. Halpern. Belief revision: a critique. *Journal of Logic, Language, and Information*, 8:401–420, 1999.
- [Katsuno and Mendelzon, 1991] Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.
- [Katsuno and Mendelzon, 1992] Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Belief Revision*, pages 183–203. Cambridge University Press, 1992.
- [Konieczny and Pino Pérez, 2000] Sébastien Konieczny and Ramón Pino Pérez. A framework for iterated revision. *Journal of Applied Non-Classical Logics*, 10(3-4):339–367, 2000.
- [Lehmann, 1995] Daniel Lehmann. Belief revision, revised. In *Proceedings of 14th Int. Joint Conf. on Artificial Intelligence*, pages 1534–1539, 1995.
- [Nayak, 1994] Abhaya C. Nayak. Iterated belief change based on epistemic entrenchment. *Erkenntnis*, 41(1994):353–390, 1994.
- [Papini, 2001] Odile Papini. Iterated revision operations stemming from the history of an agent's observations. In Hans Rott and Mary-Anne Williams, editors, *Frontiers of Belief Revision*, pages 279–301. Kluwer Academic, 2001.
- [Rott, 2001] Hans Rott. *Change, Choice and Inference: A Study of Belief Revision and Nonmonotonic Reasoning*. Oxford University Press, 2001.
- [Spohn, 1988] Wolfgang Spohn. Ordinal conditional functions: a dynamic theory of epistemic state. *Causation in Decision, Belief Change and Statistics*, pages 105–134, 1988.

Computational ontologies of parthood, componenthood, and containment*

Thomas Bittner^{1,2,3} and Maureen Donnelly^{1,3}

¹Department of Philosophy, ²Department of Geography

³New York State Center of Excellence in Bioinformatics and Life Sciences
University at Buffalo, 135 Park Hall, Buffalo, NY 14260

Abstract

Parthood, componenthood, and containment relations are commonly assumed in biomedical ontologies and terminology systems, but are not usually clearly distinguished from another. This paper contributes towards a unified theory of parthood, componenthood, and containment relations. Our goal in this is to clarify distinctions between these relations as well as principles governing their interrelations. We first develop a theory of these relations in first order predicate logic and then discuss how description logics can be used to capture some important aspects of the first order theory.

1 Introduction

My car has components, for example, its engine, its oil pump, its wheels, etc. (See Figure 1.) Roughly, a *component* of an object is a proper part of that object which has a complete bona fide boundary (i.e., boundary that correspond discontinuities in reality) and a distinct function. Thus all components of my car are parts of my car, but my car has also parts that are not components. For example, the left side my car has neither a complete bona fide boundary nor a distinct function. My car is also a *container*. It contains the driver in the seat area and a tool box and a spare-tire in its trunk. Containment is here understood as a relation which holds between disjoint material objects when one object (the containee) is located within a space partly or wholly enclosed by the container. In this paper, we study formal properties of proper parthood, componenthood, and containment relations and demonstrate how they can be represented and distinguished from one other in formal ontologies expressed in languages of different expressive power.

At first sight, these three relations seem to have quite similar properties. All three are transitive and asymmetric. The screw-driver is contained in my tool box and the tool box is contained in the trunk of my car, therefore the screw-driver is contained in the trunk of my car. And if an object (e.g., a

tool box) is contained in the trunk of my car, then the trunk of my car is *not* contained in that object. It is easy to see that the componenthood (See Figure 1) and proper parthood relations are also asymmetric and transitive. Due to their similarities these relations are not always clearly distinguished in ontologies such as, e.g., GALEN [Rogers and Rector, 2000] or SNOMED [Spackman, 2001].

However, there are important differences between these relations. There can be a container with a single containee (e.g., the screw-driver is the only tool in my tool box) but no object can have single proper part. Also the components of complex artifacts form tree-structures. Thus, two components share a component only when one is a sub-component of the other. (It is because components form tree structures that tree graphs of component structures can be given in assembly manuals.) The parthood relation does not have this property: The left half of my car and the bottom half of my car share the bottom left part of my car but they are not proper parts of each other.

Ontologies are tools for making explicit the semantics of terminology systems [Guarino, 1998]. In this paper we develop ontologies which explicate the distinct properties of proper parthood, componenthood and containment relations. These ontologies can be used to specify the meaning of terms such as ‘proper-part-of’, ‘component-of’, and ‘contained-in’. We start by characterising important properties of binary relations and then study how these properties can be expressed both in ontological theories formulated in first order logic and in ontologies formulated in a description logic.

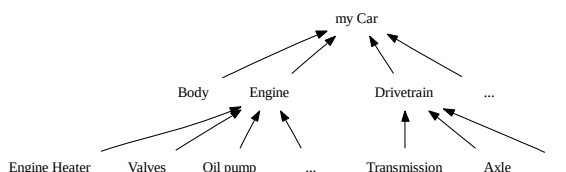


Figure 1: Car components

2 Binary relations

In this section, we define properties of binary relation structures that will be useful for distinguishing proper parthood, component-of, and containment relations.

*This work has been supported by the Wolfgang Paul Program of the Alexander von Humboldt Foundation, the EU Network of Excellence in Semantic Datamining, and the Volkswagen Foundation Project "Forms of Life".

2.1 R -structures

A R -structure is a pair, (Δ, R) , that consists of a non-empty domain Δ and a binary relation $\emptyset \neq R \subseteq \Delta \times \Delta$. We write $R(x, y)$ to say that the binary relation R holds between the individuals $x, y \in \Delta$, i.e., $(x, y) \in R$. We can define the following relations on Δ in terms of R :

$$\begin{aligned} D_{R=} \quad R_=(x, y) &=_{df} R(x, y) \text{ or } x = y \\ D_{R_O} \quad R_O(x, y) &=_{df} \exists z \in \Delta : R_=(z, x) \ \& \ R_=(z, y) \\ D_{R_i} \quad R_i(x, y) &=_{df} R(x, y) \ \& \ (\neg \exists z \in \Delta : R(x, z) \ \& \ R(z, y)) \end{aligned}$$

For a given R -structure, the defined relations $R_=$, R_O , or R_i may be empty or identical to R . For example, if R is the identity relation on Δ , i.e., $R = \{(x, x) \mid x \in \Delta\}$, then $R_= = R = R_O$ and $R_i = \emptyset$.

2.2 Properties of binary relations

An R -structure (Δ, R) may have or lack the properties listed in Table 1. For example, for any Δ the identity relation on Δ is reflexive, symmetric and transitive. Moreover, for any (Δ, R) , R_O is symmetric, R_i is intransitive, and $R_=$ is reflexive. As pointed out above, on their respective domains proper parthood, componenthood, and containment are asymmetric and transitive.

property	description
reflexive	$\forall x \in \Delta : R(x, x)$
irreflexive	$\forall x \in \Delta : \text{not } R(x, x)$
symmetric	$\forall x, y \in \Delta : \text{if } R(x, y) \text{ then } R(y, x)$
asymmetric	$\forall x, y \in \Delta : \text{if } R(x, y) \text{ then not } R(y, x)$
transitive	$\forall x, y, z \in \Delta : \text{if } R(x, y) \text{ and } R(y, z) \text{ then } R(x, z)$
intransitive	$\forall x, y \in \Delta : \text{if } R(x, y) \text{ and } R(y, z) \text{ then not } R(x, z)$
up-discrete	$\forall x, y \in \Delta : \text{if } R(x, y) \text{ then } R_i(x, y) \text{ or } \exists z \in \Delta : R(x, z) \text{ and } R_i(z, y)$
dn-discrete	$\forall x, y \in \Delta : \text{if } R(x, y) \text{ then } R_i(x, y) \text{ or } \exists z \in \Delta : R_i(x, z) \text{ and } R(z, y)$
discrete	up-discrete & dn-discrete
dense	$\forall x, y \in \Delta : \text{if } R(x, y) \text{ then } \exists z \in \Delta : R(x, z) \text{ and } R(z, y)$
WSP	$\forall x, y \in \Delta : \text{if } R(x, y) \text{ then } \exists z \in \Delta : R(z, y) \ \& \ \text{not } R_O(z, x)$
NPO	$\forall x, y \in \Delta : \text{if } R_O(x, y) \text{ then } x = y \text{ or } R(x, y) \text{ or } R(y, x)$
NSIP	$\forall x, y \in \Delta : \text{if } R_i(x, y) \text{ then } \exists z \in \Delta : R_i(z, y) \ \& \ \text{not } x = z$
SIS	$\forall x, y, z \in \Delta : \text{if } R_i(x, y) \text{ and } R_i(x, z) \text{ then } y = z$

Table 1: Properties of binary relations

We say (Δ, R) has the *weak supplementation property* (WSP) if and only if for all $x, y \in \Delta$ if $R(x, y)$ then there is a $z \in \Delta$ such that $R(z, y)$ but NOT $R_O(z, x)$. As an example of a relation that has the weak supplementation property, consider the proper parthood relation on the domain Δ_S of spatial objects, $(\Delta_S, \text{proper-part-of})$. In this structure proper-part-of_O is the overlap relation. WSP tells us that if x is a proper part of y then there exists a proper part z of y that does not overlap x . For example, since the left side of my car is a proper part of my car there is some proper part of my car (e.g., the right side of my car) which is discrete from the left side of my car.

Another example of a structure that has the weak supplementation property is the componenthood relation on the domain of artifacts, $(\Delta_A, \text{component-of})$. Here component-of_O is the relation of sharing a component. WSP tells us that if x is a component of y then there exists a component z of y such that z and x do not have a common component. For example, since the engine of my car is a component of my car there is some component of my car (e.g., the body of my car) which does not have a component in common with the engine. (See Figure 1.)

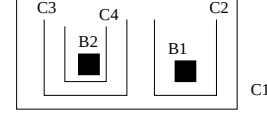


Figure 2: Nested containers

Consider the structure $(\Delta_C, \text{contained-in})$ with $\Delta_C = \{C_1, C_2, C_3, C_4, B_1, B_2\}$ as depicted in Figure 2. The block B_1 is immediately contained in the container C_2 which in turn is immediately contained in the container C_1 . B_1 is contained, but not immediately contained, in C_1 . Note that *contained-in* does NOT have the weak-supplementation property: B_1 is the only entity contained in C_2 . Thus, every entity contained in C_2 stands in the *contained-in*_O relation to B_1 .

We say (R, Δ) has the *no-partial-overlap* property (NPO) if and only if for all $x, y \in \Delta$: if $R_O(x, y)$ then $x = y$ or $R(x, y)$ or $R(y, x)$. The structure $(\Delta_A, \text{component-of})$ has the NPO property. As a representative example consider the substructure of $(\Delta_A, \text{component-of})$ depicted in Figure 1: Two distinct car components share a component only if one is a subcomponent of the other.

The structure $(\Delta_S, \text{proper-part-of})$, on the other hand, does not have the no-partial-overlap property. As pointed out earlier, the left half of my car and the lower half of my car overlap partially. Note also that containment structures (domains with a containment relation) often do not have the NPO property: Consider the tool box in the trunk of my car. It is also contained in my car. My car and the trunk of my car share a containee (the tool box), i.e., *contained-in*_O holds, but my car is not contained in the trunk of my car nor is the trunk contained in the car.

Containment structures are *discrete*. For example $(\Delta_C, \text{contained-in})$ is up- and dn-discrete: if x is contained in y then either x is an immediately contained in y or (a) there exists a z such that x is an immediately contained in z and z is contained in y , and (b) there exists a z such that x is contained in z and z is immediately contained in y . Similarly, the structure $(\Delta_A, \text{component-of})$ is discrete. If x is a component of y then either x is an immediate component of y or (a) there exists a z such that x is an immediate component of z and z is a component of y , and (b) there exists a z such that x is a component z and z is an immediate component of y . Again, Figure 1 is a representative example.

The structure $(\Delta_S, \text{proper-part-of})$, is *dense* due to the existence of *fiat* parts (parts which lack a complete bona fide

boundary) [Smith, 2001]. Consider my car and its proper parts. My car does not have an immediate proper part – Whatever proper part x we chose, there exists another slightly bigger proper part of my car that has x as a proper part.

(R, Δ) has the *single-immediate-successor* property (SIS) if and only if no $x \in \Delta$ can stand in the R_i relation to two distinct members of Δ . Again, a representative example is the component-of structure depicted in Figure 1. In the structure $(\Delta_S, \text{proper-part-of})$ SIS trivially holds since this structure has the density property and no immediate proper parts exist. But note that containment structures often do not have the SIS property: Consider again the tool box in the trunk of my car. It is also contained in my car. My car and the trunk of my car are distinct immediate containers for my tool box.

(R, Δ) has the *no-single-immediate-predecessor* property (NSIP) if and only if for all $x, y \in \Delta$: if $R_i(x, y)$ then there exists a $z \in \Delta$ such that $R_i(z, y)$ and not $x = z$. Again, the componenthood structure depicted in Figure 1 is a representative example for a structure that has the NSIP property. Again, in the structure $(\Delta_S, \text{proper-part-of})$ NSIP is trivially true since no immediate proper parts exist. But containment structures like $(\Delta_C, \text{contained-in})$ lack the NSIP property.

Given the properties in Table 1 we can classify R -structures according to the properties of the relation R . In Table 2 we list classes of R -structures that will be useful for modelling proper parthood, componenthood, and containment relations.

R -structure	properties
partial ordering (PO)	asymmetric, transitive
discrete PO	PO + discrete
parthood structure	PO + WSP + dense
component-of structure	PO + WSP, NPO, discrete

Table 2: Classes of R -structures

Finally, note the following facts about R structures: (F1) If (Δ, R) has the no-partial-overlap property then it has the single-immediate-successor property; (F2) If (Δ, R) is finite and has the single-immediate-successor property then it has the no-partial-overlap property; (F3) If (Δ, R) is up-discrete and has also the no-partial-overlap property, then (Δ, R) has the weak-supplementation property if and only if it has the no-single-immediate-predecessor property; (F4) If (Δ, R) is reflexive, then $R_i = \emptyset$.

2.3 Parthood-containment-component structures

The relations that we are interested in do not exist in separation but form complex structures involving more than one relation. The structure $(\Delta, \mathbf{PP}, \mathbf{CntIn}, \mathbf{CmpOf})$ is a *parthood-containment-component structure* if and only if: (i) the substructure $(\Delta, \mathbf{PP})$ is a parthood structure; (ii) $(\Delta, \mathbf{CntIn})$ is a discrete partial ordering; (iii) $(\Delta, \mathbf{CmpOf})$ is a component-of structure; and addition the following conditions hold:

- (iv) If $\mathbf{CntIn}(x, y)$ and $\mathbf{PP}(y, z)$ then $\mathbf{CntIn}(x, z)$;
- (v) If $\mathbf{PP}(x, y)$ and $\mathbf{CntIn}(y, z)$ then $\mathbf{CntIn}(x, z)$;
- (vi) If $\mathbf{CmpOf}(x, y)$ then $\mathbf{PP}(x, y)$;

As an example of a parthood-containment-component structure consider the set Δ formed by all parts of my car and everything that is contained in my car. The substructure $(\Delta, \mathbf{CmpOf})$ is depicted partly in Figure 1.

(iv) ensures that parts are contained in the container of the whole, e.g., my head is part of my body and my body is contained in my car, so my head must also be contained in my car. (v) ensures that if a part of some whole contains something then so does the whole, e.g., since my tool box is contained in the trunk of my car and the trunk is part of my car, my tool box is also contained in my car. (vi) tells us that componenthood is a special case of parthood, e.g., since the engine is a component of my car, it is also a proper part of my car.

3 A formal ontology of parthood, containment, and componenthood

The formal theory developed in this section is presented in standard first-order predicate logic with identity. We use x, y , and z for variables. Leading universal quantifiers are generally omitted. Names of axioms begin with the capital letter ‘A’, names of definitions begin with the capital letter ‘D’, and names of theorems begin with the capital letter ‘T’.

We include the primitive relation symbols PP , $\mathbf{CntIn}$, and $\mathbf{CmpOf}$ in the language of our theory. The intended interpretations are the relations $\mathbf{PP}$, $\mathbf{CntIn}$, and $\mathbf{CmpOf}$ respectively of parthood-containment-component structures.

3.1 Axioms for PP

We introduce the symbols $PP_=_$, PP_O , and define that $PP_=_ xy$ holds if and only if either $PP xy$ or x and y are identical ($D_{PP_=_}$); $PP_O xy$ holds iff x and y share a common part or are identical (D_{PP_O}).

$$\begin{aligned} D_{PP_=_} \quad PP_=_ xy &\equiv PP xy \vee x = y \\ D_{PP_O} \quad PP_O xy &\equiv (\exists z)(PP_=_ zx \wedge PP_=_ zy) \end{aligned}$$

We then include the axioms of asymmetry and transitivity (APP1-APP2) as well as an axiom (APP3) that ensures that interpretations of PP have the weak supplementation property (WSP).

$$\begin{aligned} APP1 \quad PP xy &\rightarrow \neg PP yx \\ APP2 \quad (PP xy \wedge PP yz) &\rightarrow PP xz \\ APP3 \quad PP xy &\rightarrow (\exists z)(PP zy \wedge \neg PP_O zx) \quad (WSP) \end{aligned}$$

The theory that includes APP1-3 as axioms is known as basic mereology [Simons, 1987]. Finally we add a density axiom to include fiat parts into our domain (APP4).

$$APP4 \quad PP xy \rightarrow (\exists z)(PP xz \wedge PP zy)$$

Models the the theory that includes APP1-4 as axioms are parthood structures as defined in Table 2.

3.2 Axioms for $\mathbf{CmpOf}$

We introduce the symbols $\mathbf{CmpOf}_=_$ and $\mathbf{CmpOf}_O$ and add the respective definitions ($D_{\mathbf{CmpOf}_=_}$ and $D_{\mathbf{CmpOf}_O}$).

$$\begin{aligned} D_{\mathbf{CmpOf}_=_} \quad \mathbf{CmpOf}_=_ xy &\equiv \mathbf{CmpOf} xy \vee x = y \\ D_{\mathbf{CmpOf}_O} \quad \mathbf{CmpOf}_O xy &\equiv (\exists z)(\mathbf{CmpOf}_=_ zx \wedge \mathbf{CmpOf}_=_ zy) \end{aligned}$$

We then include an axiom of transitivity (ACP1).

$$ACP1 \quad (CmpOf \, xy \wedge CmpOf \, yz) \rightarrow CmpOf \, xz$$

Corresponding to (vi) we add an axiom that ensures that $CmpOf \, xy$ implies $PP \, xy$ (ACP2) and can then prove that $CmpOf$ is asymmetric (TCP1).

$$ACP2 \quad CmpOf \, xy \rightarrow PP \, xy$$

$$TCP1 \quad CmpOf \, xy \rightarrow \neg CmpOf \, yx$$

We introduce the symbol $CmpOf_i$ and define $CmpOf_i \, xy$ to hold iff $CmpOf_i \, xy$ and there is no z such that $CmpOf_i \, xz$ and $CmpOf_i \, zy$ (D_{CmpOf_i}). We then add an axiom that enforces that interpretations of $CmpOf$ have the discreteness property (ACP3).

$$D_{CmpOf_i} \quad CmpOf_i \, xy \equiv CmpOf \, xy \wedge \neg(\exists z)(CmpOf \, xz \wedge CmpOf \, zy)$$

$$ACP3 \quad CmpOf \, xy \rightarrow (CmpOf_i \, xy \vee ((\exists z)(CmpOf_i \, xz \wedge CmpOf \, zy) \wedge (\exists z)(CmpOf \, xz \wedge CmpOf_i \, zy)))$$

From D_{CmpOf_i} we can prove immediately that $CmpOf_i$ is intransitive (TCP2).

$$TCP2 \quad CmpOf_i \, xy \wedge CmpOf_i \, yz \rightarrow \neg CmpOf_i \, xz$$

We then add axioms that require that $CmpOf$ has the no-partial-overlap property (ACP4) and that $CmpOf$ has the no-single-immediate-predecessor property (ACP5).

$$ACP4 \quad CmpOf_O \, xy \rightarrow (CmpOf_{=} \, xy \vee CmpOf \, xz)$$

$$ACP5 \quad CmpOf_i \, xy \rightarrow (\exists z)(CmpOf_i \, zy \wedge \neg z = x)$$

We now can prove that the weak-supplementation principle holds (TCP3) and that nothing has two distinct immediate successors (TCP4).

$$TCP3 \quad CmpOf \, xy \rightarrow (\exists z)(CmpOf \, zy \wedge \neg CmpOf_O \, zx)$$

$$TCP4 \quad CmpOf_i \, xz_1 \wedge CmpOf_i \, xz_2 \rightarrow z_1 = z_2$$

3.3 Axioms for $CntIn$

We introduce the symbols $CntIn_{=}$, $CntIn_O$, and $CntIn_i$ and add the respective definitions ($D_{CntIn_{=}}$, D_{CntIn_O} , and D_{CntIn_i}).

$$D_{CntIn_{=}} \quad CntIn_{=} \, xy \equiv CntIn \, xy \vee x = y$$

$$D_{CntIn_O} \quad CntIn_O \, xy \equiv (\exists z)(CntIn_{=} \, xz \wedge CntIn_{=} \, zy)$$

$$D_{CntIn_i} \quad CntIn_i \, xy \equiv CntIn \, xy \wedge \neg(\exists z)(CntIn \, xz \wedge CntIn \, zy)$$

We then include axioms of asymmetry, transitivity, and discreteness (ACT1-3).

$$ACT1 \quad CntIn \, xy \rightarrow \neg CntIn \, yx$$

$$ACT2 \quad (CntIn \, xy \wedge CntIn \, yz) \rightarrow CntIn \, xz$$

$$ACT3 \quad CntIn \, xy \rightarrow (CntIn_i \, xy \vee ((\exists z)(CntIn_i \, xz \wedge CntIn \, zy) \wedge (\exists z)(CntIn \, xz \wedge CntIn_i \, zy)))$$

We add axioms, corresponding to (iv) and (v), parts are contained in the container of the whole (ACT4) and that if a part contains something then so does the whole (ACT5).

$$ACT4 \quad PP \, xy \wedge CntIn \, yz \rightarrow CntIn \, xz$$

$$ACT5 \quad CntIn \, xy \wedge PP \, yz \rightarrow CntIn \, xz$$

We call the theory consisting of the axioms APP1-4, ACP1-5 and ACT1-5 *FO-PCC*. Parthood-composition-containment structures are models of this theory.

4 Representation in a description logic

Description Logics (DLs) are a family of logical formalisms which are significantly less powerful than first order logic but which are (relatively) easily implemented on the computer [Baader *et al.*, 2002]. The task of this section is to investigate to what extent and how *FO-PCC* can be approximated by a theory expressed in a description logic. For this task, we consider DLs with different expressive capabilities, some of which are better suited than others for formulating properties of parthood, componenthood and containment relations. Notice, that it is not the purpose of this paper to provide a complexity analysis for these DLs.

4.1 The syntax and semantics of description logics

Basic expressions in description logics are *concept* and *role descriptions*. Concepts are interpreted as sets. Roles are interpreted as binary relations. General rules for forming concept and role descriptions (based on [Baader *et al.*, 2002]) are given below. Note, however, that specific DLs typically allow for the formulation of some, but not all, of the complex concept and role descriptions listed.

Every concept name is a concept description (atomic concept), $\top$ is the *top-concept*. $\perp$ is the *bottom-concept*. If C and D are concept descriptions then $C \sqcap D$ (concept-intersection), $C \sqcup D$ (concept-union), $\sim C$ (concept-complement) are also concept descriptions. Every role name, R , is a role description (an atomic role). If S and T are role descriptions, then $S \sqcap T$ (role-intersection), $S \sqcup T$ (role-union), $\sim S$ (role-complement), $S \circ T$ (role-composition), and R^{-} (role-inverse) are also role descriptions. Id is the name of the identity role. If C is a concept description and R is a role name then $(\exists R.C)$, $(\forall R.C)$, and $(= 1R)$ are concept descriptions. The semantics of the various constructors is given in Table 3.

A *terminology* is a set of terminological axioms of the form $C \doteq D$ and $S \doteq T$ (called equalities) or $C \sqsubseteq D$ and $S \sqsubseteq T$ (called inclusions), where C and D are concept descriptions and S and T are role descriptions. An interpretation $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $S \sqsubseteq T$ iff $S^{\mathcal{I}} \subseteq T^{\mathcal{I}}$. (See [Baader *et al.*, 2002].) It satisfies an equality $C \doteq D$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ and $S \doteq T$ iff $S^{\mathcal{I}} = T^{\mathcal{I}}$.

(ia - b)	$\top^{\mathcal{I}} = \Delta, \perp^{\mathcal{I}} = \emptyset;$
(iia - c)	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}},$ $(\sim C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}};$
(iii)	$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta \mid (\exists b)((a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}})\}$
(iv)	$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta \mid (b)((a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}})\}$
(v)	$(= 1R)^{\mathcal{I}} = \{a \in \Delta \mid \{b \mid (a, b) \in R^{\mathcal{I}}\} = 1\}$
(via - c)	$(S \sqcap T)^{\mathcal{I}} = S^{\mathcal{I}} \cap T^{\mathcal{I}}, (S \sqcup T)^{\mathcal{I}} = S^{\mathcal{I}} \cup T^{\mathcal{I}},$ $(\sim S)^{\mathcal{I}} = \Delta \times \Delta \setminus S^{\mathcal{I}};$
(vii)	$(S \circ T)^{\mathcal{I}} = \{(a, c) \in \Delta \times \Delta \mid$ $(\exists b)((a, b) \in S^{\mathcal{I}} \wedge (b, c) \in T^{\mathcal{I}})\}$
(viii)	$\text{Id}^{\mathcal{I}} = \{(a, a) \mid a \in \Delta\}$
(ix)	$(R^{-})^{\mathcal{I}} = \{(b, a) \in \Delta \times \Delta \mid (a, b) \in R^{\mathcal{I}}\}$

Table 3: Concept and role constructors with their semantics. $\mathcal{I}$ is the interpretation function and Δ is the domain of individuals.

4.2 Stating ontological principles

Let $\mathcal{L}_{WSP}$ be a language that includes at least the constructors (ia, iia, iii, via-c, vii, viii, ix [Table 3]). In this language we can state a DL-version of *FO-PCC*. In particular, if R is the name of a relation R then we are able to state in this language that R has the WSP property, we are able to define the relation R_i in terms of R , and we are able to state that R is a discrete (or dense) relation:

(WSP)	$R^- \sqsubseteq R^- \circ \sim ((R^- \sqcup \text{Id}) \circ (R \sqcup \text{Id}))$
(def-i)	$R_i \doteq R \sqcap \sim (R \circ R)$
(discrete)	$R \sqsubseteq R_i \sqcup (R \circ R_i \sqcap R_i \circ R)$
(dense)	$R \sqsubseteq R \circ R$

But since $\mathcal{L}_{WSP}$ is undecidable [Schmidt-Schauß, 1989], it is important to identify less complex sub-languages of $\mathcal{L}_{WSP}$ that are still sufficient to state axioms distinguishing parthood, componenthood, and containment relations. Otherwise the DL version of *FO-PCC* would have no computational advantages over the first order theory.

Let $\mathcal{L}$ be the DL which includes only the constructors (ia-b, iia, iii, vii, ix, and v [Table 3]) and in which the role composition operator (vii) only occurs in acyclic role terminologies with inclusion axioms of the form $R \circ R \sqsubseteq R$, $S \circ R \sqsubseteq R$, and $R \circ S \sqsubseteq R$. Unlike $\mathcal{L}_{WSP}$ the DL $\mathcal{L}$ is decidable [Horrocks and Sattler, 2004].

If R is the name of the relation R then we are able to state in $\mathcal{L}$ that R is transitive ($R \circ R \sqsubseteq R$). Moreover, in $\mathcal{L}$ we can very naturally represent DL-versions of the axioms ACP2 and ACT4-5. Unfortunately, in $\mathcal{L}$ we are not able to state either that R is asymmetric, that R has the WSP property, or that R has the NPO property. Also we cannot state a DL-version of the definition of R_i in terms of R (as in def-i).

Let R_i be an undefined relation name interpreted as R_i in the R -structure (Δ, R) (e.g., as *contained-in_i* in a containment structure). In $\mathcal{L}$ we are able to use this additional primitive to say that R has the no-single-immediate-predecessor property (NSIP) and the single-immediate-successor property (SIS).

(SIS)	$\exists R_i. \top \sqsubseteq (=1) R_i. \top$
(NSIP)	$(=1) R_i. \top \sqsubseteq \perp$

Notice however that, since we introduced R_i as an undefined relation name we do not know that the interpretation of R_i is an intransitive subrelation of R unless additional axioms are included in the theory. In $\mathcal{L}$ we can state that R_i is a subrelation of R but we cannot not say that R_i is intransitive. Notice also, that in $\mathcal{L}$, we cannot say that R is irreflexive ($R \circ \text{Id} \sqsubseteq \perp$) since $\mathcal{L}$ does not include a constructor for the identity relation.

Let $\mathcal{L}^{\sim \text{Id} \sqcup}$ be the DL obtained by extending $\mathcal{L}$ with the identity relation (viii), negation restricted to relation names (a restricted version of vic), and role union (vib). In this DL we can say that R_i is intransitive, that R is asymmetric, and that R has the NPO property.

(intrans)	$R_i \circ R_i \sqsubseteq (\sim R_i)$
(asym)	$R^- \sqsubseteq (\sim R)$
(NPO)	$(R^- \circ R) \sqsubseteq R \sqcup \text{Id} \sqcup R^-$

Unfortunately, including role negation into a DL-language significantly increases the complexity of the underlying reasoning [Lutz and Sattler, 2000]. Though $\mathcal{L}^{\sim \text{Id} \sqcup}$ is less expressive than $\mathcal{L}_{WSP}$ (we cannot state WSP or discreteness axioms or define R_i) it is an open question whether $\mathcal{L}^{\sim \text{Id} \sqcup}$ is decidable. (It is known though that $\mathcal{ALC}$ -DLs that include axioms of the form $R \circ S \sqsubseteq T_1 \sqcup \dots \sqcup T_n$ are undecidable [Wessel, 2001].)

4.3 Describing parthood-composition-containment structures in $\mathcal{L}$

We chose $\mathcal{L}$ as the DL to formulate an *approximation* of *FO-PCC* because $\mathcal{L}$ is decidable and does include the composition operator which is important for expressing interrelations between relation and for reasoning (particularly in biomedical ontologies) [Spackman, 2001; Rogers and Rector, 2000; Horrocks and Sattler, 2004].

We add the symbols CP, PP and CT as well as CP_i , PP_i and CT_i to $\mathcal{L}$. The intended interpretations of these symbols are the relations **CmpOf**, **PP**, **CntIn**, **CmpOf_i**, **PP_i**, and **CntIn_i** of parthood-composition-containment structures. We then include the following axioms for CP and PP:

component-of	proper-part-of
(A1) $\text{CP}_i \sqsubseteq \text{CP}$	(A5) $\text{PP}_i \sqsubseteq \text{PP}$
(A2) $\text{CP} \circ \text{CP} \sqsubseteq \text{CP}$	(A6) $\text{PP} \circ \text{PP} \sqsubseteq \text{PP}$
(A3) $(=1)\text{CP}_i. \top \sqsubseteq \perp$	(A7) $(=1)\text{PP}_i. \top \sqsubseteq \perp$
(A4) $\exists \text{CP}_i. \top \sqsubseteq (=1)\text{CP}_i. \top$	—

For CT we include a subrelation axiom and a transitivity axiom:

$$\text{A8 } \text{CT}_i \sqsubseteq \text{CT} \quad \text{A9 } \text{CT} \circ \text{CT} \sqsubseteq \text{CT}$$

We include also axioms A10-12 corresponding to (iv-vi) in Section 2.3.

$$\text{A10 } \text{CP} \sqsubseteq \text{PP} \quad \text{A11 } \text{PP} \circ \text{CT} \sqsubseteq \text{CT} \quad \text{A12 } \text{CT} \circ \text{PP} \sqsubseteq \text{CT}$$

We call the theory formed by A1-12 *DL-PCC*. The sub-theory formed by A1-4 is similar to the theories proposed in [Sattler, 2000] and [Lambrix and Padgham, 2000].

But, as discussed in the previous subsection, we are not able to add to *DL-PCC* the following axioms and definitions that are needed to constrain the models to parthood-composition-containment structures: (1) We are not able to state that CP, PP, and CT are asymmetric and irreflexive; (2) We are not able to state a discreteness axiom for CP or CT or a density axiom for PP; (3) We are not able to define CP_i , PP_i , and CT_i in terms of CP, PP, and CT respectively; (4) We are not able to state the weak supplementation principle (WSP) for interpretations of PP.

Consider (1). Since *DL-PCC* lacks asymmetry axioms it admits models in which CP, PP, and CT are interpreted as reflexive relations. In those models CP_i , PP_i , and CT_i are all interpreted as the empty relation (making the axioms A3, A4, and A7 trivially true). (See also F4 in Section 2.2.) For example the structure $(\Delta_C, \text{identical-to})$ is a model of *DL-PCC* (but not of *FO-PCC*) if we interpret CP, PP, and CT as *identical-to* and CP_i , PP_i , and CT_i as $\emptyset$. Clearly, this model is not a parthood-component-containment structure.

Consider (3). We included CP_i , PP_i , and CT_i as undefined primitives in *DL-PCC* and added axioms (A1, A5, and

A8) that require their interpretations to be sub-relations of the interpretations of CP, PP, and CT. Unfortunately, *DL-PCC* admits models in which PP_i and PP are the same relation (similarly for CP and CP_i or CT and CT_i). Consider Figure 2 and interpret CP and CP_i as the relation $icr = \{(C_2, C_1), (C_3, C_1)\}$ (*immediately-contained-in-the-root-container*), and PP, PP_i , CT, CT_i all as *contained-in*. Then $(\Delta_C, \text{contained-in}, icr)$ is a model of *DL-PCC* (but not of *FO-PCC*). This particular kind of unintended interpretations of PP_i and CT_i can be avoided by requiring that the interpretation of these relations are intransitive. However in $\mathcal{L}$ we are not able to require that a given relation is intransitive.

Consider (4). The closest we can get to requiring that the interpretation of PP has the WSP property is to require that the NSIP property holds (axiom 7). However the NSIP property is strictly weaker than the WSP property.¹ Consequently, *DL-PCC* admits models that would have been rejected by a theory including an axiom that requires WSP for interpretations of PP (e.g. *FO-PCC*). Similar comments apply to (2).

These are strong limitations if the purpose of the presented theory is to serve as an *ontology* that specifies the meaning of the terms ‘proper part of’, ‘component of’ and ‘contained in’ rather than to support automatic reasoning in some specific and possibly finite domain.² If the DL $\mathcal{L}^{\sim Id \sqcup}$ is decidable we can get a better DL approximation of *FO-PCC* that is computationally tractable. But even a $\mathcal{L}^{\sim Id \sqcup}$ version of *FO-PCC* will fall short of *FO-PCC* in expressivity since we cannot state WSP for PP.

5 Conclusions

We studied formal properties of parthood, componenthood and containment relations. Since it is the purpose of an ontology to make explicit the semantics of terminology systems, it is important to explicitly distinguish relations such as proper parthood, componenthood, and containment. We demonstrated that first order logic has the expressive power required to distinguish important properties of these relations. In description logics like $\mathcal{L}$ several important properties of these relations cannot be specified.

DLs are best used as reasoning tools for specific tasks in specific domains (as suggested in [Sattler, 1995; 2000; Lambrix and Padgham, 2000]). DLs are not appropriate for formulating complex interrelations between relations. Thus we need to understand a computational ontology as consisting of two complementary components: (1) a DL based ontology that enables automatic reasoning and constrains meaning as much as possible and (2) a first order ontology that serves as meta-data and makes explicit properties of relations that cannot be expressed in computationally efficient description logics. The first order theory then can be used by a human

¹NSIP entails WSP only in conjunction with discreteness and NPO. We cannot require that interpretations of PP are discrete or have the NPO property since then the proper parthood relation on the domain of spatial objects could not be an interpretation for PP.

²If we constrain our models to finite domains then, for example, it is indeed sufficient to include (A3) and (A4) as axioms to require the WSP and the NPO properties for CP (F1-3).

being to decide whether or not the DL-ontology in question is applicable to her domain. Moreover, meta-data can also be used to write special-purpose programs that phrase knowledge bases and enforce the usage of relations in accordance to the meta-data.

References

- [Baader *et al.*, 2002] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2002.
- [Guarino, 1998] N. Guarino. Formal ontology and information systems. In Nicola Guarino, editor, *Formal Ontology and Information Systems, (FOIS’98)*. IOS Press, 1998.
- [Horrocks and Sattler, 2004] I. Horrocks and U. Sattler. Decidability of SHIQ with complex role inclusion axioms. *Artificial Intelligence*, 120(1-2):79–104, 2004.
- [Lambrix and Padgham, 2000] P. Lambrix and L. Padgham. Conceptual modelling in a document management environment using part-of reasoning in description logics. *Data and Knowledge Engineering*, 32(1):51–86, 2000.
- [Lutz and Sattler, 2000] C. Lutz and U. Sattler. Mary likes all cats. In F. Baader and U. Sattler, editors, *International Workshop in Description Logics (DL2000)*, number 33 in CEUR-WS, pages 213–226.
- [Rogers and Rector, 2000] J. Rogers and A. Rector. GALEN’s model of parts and wholes: experience and comparisons. In *AMIA Symp 2000*, pages 714–8, 2000.
- [Sattler, 1995] Ulrike Sattler. A concept language for an engineering application with part-whole relations. In A. Borgida, M. Lenzerini, D. Nardi, and B. Nebel, editors, *International Workshop on Description Logics*, pages 119–123, Rome, 1995.
- [Sattler, 2000] U. Sattler. Description logics for the representation of aggregated objects. In W. Horn, editor, *14th European Conference on Artificial Intelligence*. IOS Press, Amsterdam, 2000.
- [Schmidt-Schauß, 1989] Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In *1st International Conference on Principles of Knowledge Representation and Reasoning (KR89)*, pages 421–431, 1989.
- [Simons, 1987] P. Simons. *Parts, A Study in Ontology*. Clarendon Press, Oxford, 1987.
- [Smith, 2001] B. Smith. Fiat objects. *Topoi*, 20(2):131–48, 2001.
- [Spackman, 2001] K.A. Spackman. Normal forms for description logic expressions of clinical concepts in SNOMED RT. *Journal of the American Medical Informatics Association*, pages 627–631, 2001. Symposium Supplement.
- [Wessel, 2001] M. Wessel. Obstacles on the way to qualitative spatial reasoning with description logics: Some undecidability results. In *International Workshop in Description Logics 2001*, 2001.

Propositional Argumentation and Causal Reasoning

Alexander Bochman

Computer Science Department,
Holon Academic Institute of Technology, Israel
e-mail: bochmana@hait.ac.il

Abstract

The paper introduces a number of propositional argumentation systems obtained by gradually extending the underlying language and associated monotonic logics. An assumption-based argumentation framework [Bondarenko *et al.*, 1997] will constitute a special case of this construction. In addition, a stronger argumentation system in a full classical language will be shown to be equivalent to a system of causal reasoning [Giunchiglia *et al.*, 2004]. The implications of this correspondence for the respective nonmonotonic theories of argumentation and causal reasoning are discussed.

1 Introduction

In this study we will demonstrate, among other things, that two relatively recent approaches to nonmonotonic reasoning, namely argumentation theory [Dung, 1995b; Bondarenko *et al.*, 1997] and a theory of causal nonmonotonic reasoning [McCain and Turner, 1997; Giunchiglia *et al.*, 2004; Bochman, 2003b; 2004a] are actually ‘two faces of the same coin’. The common basis of these formalisms is a notion of explanation, which takes the form of an attack relation in the argumentation theory, and a causal relation in the theory of causal reasoning. It will be shown that, at some level of generalization, these two relations will be directly interdefinable.

Both the argumentation theory and theory of causal nonmonotonic reasoning have proved to be powerful frameworks for representing different kinds of nonmonotonic reasoning, including traditional nonmonotonic logics, logic programming, abduction and reasoning about actions and change. The present study should hopefully provide a basis for clarifying the representation capabilities of these formalisms, and a step toward a unified theory of nonmonotonic reasoning.

Another objective of this study consists in a systematic development of a *propositional* approach to argumentation, in which arguments are represented as (special kinds of) propositions. This approach will naturally lead us to developing the underlying *logics* of argumentation (cf. [Chesñevar *et al.*, 2000; Prakken and Vreeswijk, 2001]) and their associated semantics. As will be shown, the resulting argumentation systems will directly represent a number of important nonmonotonic formalisms.

The plan of the paper is as follows. We introduce first an abstract argumentation theory, and describe its four-valued semantics based on acceptance and rejection of arguments. Adding a ‘global’ negation to this language will give us a formalism that generalizes the assumption-based argumentation framework from [Bondarenko *et al.*, 1997]. Then we describe an alternative extension of the argumentation theory to the full classical propositional language. A special case of the latter formalism will be shown to be equivalent to a system of causal reasoning. Finally, we will consider the correspondence between argumentation and causal reasoning on the level of their associated nonmonotonic semantics.

2 Collective argumentation

As a general formal basis of argumentation theory, we will adopt the formalism of collective argumentation suggested in [Bochman, 2003a] as a ‘disjunctive’ generalization of Dung’s argumentation theory. In this formalism, a primitive attack relation holds between *sets* of arguments: in the notation introduced below, $a \hookrightarrow b$ says that a set a of arguments attacks a set of arguments b . This fact implies, of course, that these two sets arguments are incompatible. $a \hookrightarrow b$ says, however, more than that, namely that the set of arguments a , being accepted, provides a reason, or explanation, for rejection of the set of arguments b . Accordingly, the attack relation will not in general be symmetric, since acceptance of b need not give reasons for rejection of a . In addition, the attack relation is not reducible to attacks between individual arguments. For instance, we can disprove some conclusion jointly supported by a disputed set of arguments, though no particular argument in the set, taken alone, could be held responsible for this.

In what follows, $a, b, c, \dots$ will denote finite sets of arguments, while $u, v, w, \dots$ arbitrary such sets. We will use the same agreements for the attack relation as for usual consequence relations. Thus, $a, a_1 \hookrightarrow b, B$ will have the same meaning as $a \cup a_1 \hookrightarrow b \cup \{B\}$, etc.

Definition 2.1. Let $\mathcal{A}$ be a set of arguments. A (collective) *attack relation* is a relation $\hookrightarrow$ on finite sets of arguments satisfying the following postulate:

Monotonicity If $a \hookrightarrow b$, then $a, a_1 \hookrightarrow b, b_1$.

Though defined primarily on finite sets of arguments, the attack relation will be extended to arbitrary such sets by imposing the compactness requirement: for any $u, v \subseteq \mathcal{A}$,

(Compactness) $u \hookrightarrow v$ if and only if $a \hookrightarrow b$, for some finite $a \subseteq u, b \subseteq v$.

The original Dung's argumentation theory [Dung, 1995b] can be seen as a special case of collective argumentation that satisfies additional properties (cf. [Kakas and Toni, 1999]):

Definition 2.2. An attack relation is *normal* if no set of arguments attacks $\emptyset$, and the following condition is satisfied:

(Locality) If $a \hookrightarrow b, b_1$, then either $a \hookrightarrow b$, or $a \hookrightarrow b_1$.

For a normal attack, $a \hookrightarrow b$ holds iff $a \hookrightarrow A$, for some $A \in b$. As was shown in [Bochman, 2003a], the resulting argumentation theory coincides with that given in [Dung, 1995a].

By an *argument theory* we will mean an arbitrary set of attacks $a \hookrightarrow b$ between sets of arguments. Any argument theory Δ generates a unique least attack relation that we will denote by $\hookrightarrow_\Delta$. The latter can be described directly as follows:

$$u \hookrightarrow_\Delta v \text{ iff } a \hookrightarrow b \in \Delta, \text{ for some } a \subseteq u, b \subseteq v.$$

It can be easily verified that an attack relation is normal if and only if it is generated in this sense by an argument theory consisting only of attack rules of the form $a \hookrightarrow A$.

2.1 Four-valued semantics

Collective argumentation can be given a four-valued semantics that will be instructive in describing the meaning of the attack relation. This semantics stems from the following understanding of an attack $a \hookrightarrow b$:

If all arguments in a are accepted, then at least one of the arguments in b should be rejected.

The argumentation theory does not impose, however, the classical constraints on acceptance and rejection of arguments, so an argument can be both accepted and rejected, or neither accepted, nor rejected. Such an understanding can be captured formally by assigning any argument a *subset* of the set $\{t, f\}$, where t denotes acceptance (truth), while f denotes rejection (falsity). This is nothing other than the well-known *Belnap's interpretation* of four-valued logic (see [Belnap, 1977]). As a result, collective argumentation acquires a natural four-valued semantics described below.

Definition 2.3. An attack $a \hookrightarrow b$ will be said to *hold* in a four-valued interpretation ν of arguments, if either $t \notin \nu(A)$, for some $A \in a$, or $f \in \nu(B)$, for some $B \in b$.

An interpretation ν will be called a *model* of an argument theory Δ if every attack from Δ holds in ν .

Since an attack relation can be seen as a special kind of an argument theory, the above definition determines also the notion of a model for an attack relation.

For a set I of four-valued interpretations, we will denote by $\hookrightarrow_I$ the set of all attacks that hold in each interpretation from I . Then the following result is actually a representation theorem showing that the four-valued semantics is adequate for collective argumentation (see [Bochman, 2003a]).

Theorem 2.1. $\hookrightarrow$ is an attack relation iff it coincides with $\hookrightarrow_I$, for some set of four-valued interpretations I .

2.2 Assumption-based argumentation

The notion of an argument is often taken as primitive in argumentation theory, which allows for a possibility of considering arguments that are not propositional in character (e.g., arguments as inference rules, or derivations). As has been shown in [Bondarenko *et al.*, 1997], however, a powerful version of argumentation theory can be obtained by identifying arguments with propositions of a special kind called *assumptions*¹. This representation can be refined further to a full-fledged theory of *propositional* argumentation in a certain well-defined propositional language.

Let us extend the language of arguments with a negation connective $\sim$ having the following semantic interpretation:

$\sim A$ is accepted iff A is rejected

$\sim A$ is rejected iff A is accepted.

The connective $\sim$ will be called a *global negation*, since it switches the evaluation contexts between acceptance and rejection. An axiomatization of this connective in argumentation theory can be obtained by imposing the following rules on the attack relation (see [Bochman, 2003a]):

$$A \hookrightarrow \sim A \quad \sim A \hookrightarrow A$$

If $a \hookrightarrow A, b$ and $a, \sim A \hookrightarrow b$, then $a \hookrightarrow b$ (AN)

If $a, A \hookrightarrow b$ and $a \hookrightarrow b, \sim A$, then $a \hookrightarrow b$

Attack relations satisfying the above postulates will be called *N-attack relations*. It turns out that the latter are interdefinable with certain consequence relations.

A *Belnap consequence relation* in a propositional language with a global negation $\sim$ is a Scott (multiple-conclusion) consequence relation $\Vdash$ satisfying the postulates

(Reflexivity) $A \Vdash A$;

(Monotonicity) If $a \Vdash b$ and $a \subseteq a', b \subseteq b'$, then $a' \Vdash b'$;

(Cut) If $a \Vdash b, A$ and $a, A \Vdash b$, then $a \Vdash b$,

as well as the following Double Negation rules for $\sim$:

$$A \Vdash \sim \sim A \quad \sim \sim A \Vdash A.$$

For any set u of propositions, we will denote by $\sim u$ the set $\{\sim A \mid A \in u\}$. Now, for a given N-attack relation, we can define the following consequence relation:

$$a \Vdash b \equiv a \hookrightarrow \sim b \quad (\text{CA})$$

Similarly, for any Belnap consequence relation we can define the corresponding attack relation as follows:

$$a \hookrightarrow b \equiv a \Vdash \sim b \quad (\text{AC})$$

As has been shown in [Bochman, 2003a], the above definitions establish an exact equivalence between N-attack relations and Belnap consequence relations. This correspondence allows us to represent an assumption-based argumentation framework from [Bondarenko *et al.*, 1997] entirely in the framework of attack relations.

Slightly changing the definitions from [Bondarenko *et al.*, 1997], an assumption-based argumentation framework can

¹See also [Kowalski and Toni, 1996].

defined as a triple consisting of an underlying deductive system (including the current set of beliefs), a distinguished subset of propositions Ab called *assumptions*, and a mapping from Ab to the set of all propositions of the language that determines the *contrary* $\bar{A}$ of any assumption A .

Now, the underlying deductive system can be expressed directly in the framework of N-attack relations by identifying deductive rules $a \vdash A$ with attacks of the form $a \hookrightarrow \sim A$. Furthermore, the global negation $\sim$ can serve as a faithful logical formalization of the operation of taking the contrary. More precisely, given an arbitrary language $\mathcal{L}$ that does not contain $\sim$, we can *define* assumptions as propositions of the form $\sim A$, where $A \in \mathcal{L}$. Then, since $\sim$ satisfies double negation, a negation of an assumption will be a proposition from $\mathcal{L}$. Moreover, such a ‘negative’ representation of assumptions will agree with the applications of the argumentation theory to other nonmonotonic formalisms, described in [Bondarenko *et al.*, 1997]. Accordingly, N-attack relations can be seen as a proper generalization of the assumption-based framework.

3 Classical propositional argumentation

Taking seriously the idea of propositional argumentation, it is only natural to make a further step and extend the underlying language of arguments to a full classical propositional language. The latter step should be coordinated, however, with the inherently four-valued nature of an attack relation. And the way to do this amounts to requiring that the relevant classical connectives should behave in a usual classical way with respect to both acceptance and rejection of arguments.

As a first connective of this kind, we introduce the *conjunction* connective $\wedge$ on arguments that is determined by the following familiar semantic conditions:

- $A \wedge B$ is accepted iff A is accepted and B is accepted
- $A \wedge B$ is rejected iff A is rejected or B is rejected

As can be seen, $\wedge$ behaves as an ordinary classical conjunction with respect to acceptance and rejection of arguments. On the other hand, it is a four-valued connective, since the above conditions determine a four-valued truth-table for conjunction in the Belnap’s interpretation of four-valued logic (see [Belnap, 1977]). The following postulates provide a syntactic characterization of this connective for attack relations:

$$\begin{aligned} a, A \wedge B \hookrightarrow b & \text{ iff } a, A, B \hookrightarrow b \\ a \hookrightarrow A \wedge B, b & \text{ iff } a \hookrightarrow A, B, b \end{aligned} \quad (A_{\wedge})$$

Collective attack relations satisfying these postulates will be called *conjunctive*. The next result shows that they give a complete description of the four-valued conjunction.

Corollary 3.1. *An attack relation is conjunctive if and only if it coincides with $\hookrightarrow_I$, for some set of four-valued interpretations I in a language with a four-valued conjunction $\wedge$.*

An immediate benefit of introducing conjunction into the language of argumentation is that any finite set of arguments a becomes reducible to a single argument $\bigwedge a$:

$$a \hookrightarrow b \text{ iff } \bigwedge a \hookrightarrow b.$$

As a result, the collective attack relation in this language is reducible to an attack relation between individual arguments, as suggested already in [Dung, 1995b].

Having a conjunction at our disposal, we only have to add a classical negation $\neg$ in order to obtain a classical language. Moreover, since sets of arguments are reducible to their conjunctions, we can represent the resulting argumentation theory using just a binary attack relation on classical formulas.

As a basic condition on argumentation in the classical propositional language, we will require only that the attack relation should respect the classical entailment $\models$ in the precise sense of being monotonic with respect to $\models$ on both sides.

Definition 3.1. A *propositional attack relation* is a relation $\hookrightarrow$ on the set of classical propositions satisfying the following postulates:

- (Left Strengthening)** If $A \models B$ and $B \hookrightarrow C$, then $A \hookrightarrow C$;
- (Right Strengthening)** If $A \hookrightarrow B$ and $C \models B$, then $A \hookrightarrow C$;
- (Truth)** $t \hookrightarrow f$;
- (Falsity)** $f \hookrightarrow t$.

Left Strengthening says that logically stronger arguments should attack any argument that is attacked already by a logically weaker argument, and similarly for Right Strengthening. Truth and Falsity postulates characterize the limit cases of argumentation by stipulating that any tautological argument attacks any contradictory one, and vice versa.

There exists a simple definitional way of extending the above attack relation to a collective attack relation between arbitrary sets of propositions. Namely, for any sets u, v of propositions, we can define $u \hookrightarrow v$ as follows:

$$u \hookrightarrow v \equiv \bigwedge a \hookrightarrow \bigwedge b, \text{ for some finite } a \subseteq u \text{ and } b \subseteq v$$

The resulting attack relation will satisfy the properties of collective argumentation, as well as the postulates $(A_{\wedge})$ for conjunction.

Finally, in order to acquire full expressive capabilities of an argumentation theory, we can add the global negation $\sim$ to the language. Actually, a rather simple characterization of the resulting collective argumentation theory can be obtained by accepting the postulates AN for $\sim$, plus the following rule that permits the use of classical entailment in attacks:

Classicality If $a \models A$, then $a \hookrightarrow \sim A$ and $\sim A \hookrightarrow a$.

It can be verified that the resulting system satisfies all the postulates for propositional argumentation. The system will be used later for a direct representation of default logic.

3.1 Semantics

A semantic interpretation of propositional attack relations can be obtained by generalizing four-valued interpretations to pairs (u, v) of deductively closed theories, where u is the set of accepted propositions, while v the set of propositions that are not rejected. Such pairs will be called *bimodels*, while a set of bimodels will be called a *binary semantics*.

Definition 3.2. An attack $A \hookrightarrow B$ will be said to be *valid* in a binary semantics $\mathcal{B}$ if there is no bimodel (u, v) from $\mathcal{B}$ such that $A \in u$ and $B \in v$.

We will denote by $\hookrightarrow_{\mathcal{B}}$ the set of attacks that are valid in a semantics $\mathcal{B}$. This set forms a propositional attack relation. Moreover, the following result shows that propositional attack relations are actually complete for the binary semantics.

Theorem 3.2. $\hookrightarrow$ is a propositional attack relation if and only if it coincides with $\hookrightarrow_{\mathcal{B}}$, for some binary semantics $\mathcal{B}$.

3.2 Probative and causal argumentation

We will introduce now some stronger propositional attack relations that will be shown to correspond to systems of causal inference. These stronger attack relations are obtained by adding the following quite reasonable postulates:

(Left Or) If $A \hookrightarrow C$ and $B \hookrightarrow C$, then $A \vee B \hookrightarrow C$;

(Right Or) If $A \hookrightarrow B$ and $A \hookrightarrow C$, then $A \hookrightarrow B \vee C$;

(Self-Defeat) If $A \hookrightarrow A$, then $\mathbf{t} \hookrightarrow A$.

Definition 3.3. A propositional attack relation will be called *probative* if it satisfies Left Or, *basic*, if it also satisfies Right Or, and *causal*, if it is basic and satisfies Self-Defeat.

Probative argumentation allows for reasoning by cases. Its semantic interpretation can be obtained by restricting bimodels to pairs (α, v) , where α is a world (maximal classically consistent set). The corresponding binary semantics will also be called *probative*. Similarly, the semantics for basic argumentation is obtained by restricting bimodels to world pairs (α, β) ; such a binary semantics will be called *basic*. Finally, the *causal* binary semantics is obtained from the basic semantics by requiring further that (α, β) is a bimodel only if (β, β) is also a bimodel.

Corollary 3.3. A propositional attack relation is probative [basic, causal] iff it is determined by a probative [resp. basic, causal] binary semantics.

Basic propositional argumentation can already be given a purely four-valued semantic interpretation, in which the classical negation $\neg$ has the following semantic description²:

$\neg A$ is accepted iff A is not accepted

$\neg A$ is rejected iff A is not rejected

A syntactic characterization of this connective in collective argumentation can be obtained by imposing the rules

$$\begin{aligned} A, \neg A &\hookrightarrow \quad \hookrightarrow A, \neg A \\ \text{If } a, A &\hookrightarrow b \text{ and } a, \neg A &\hookrightarrow b \text{ then } a &\hookrightarrow b & (A_{\neg}) \\ \text{If } a &\hookrightarrow b, A \text{ and } a &\hookrightarrow b, \neg A \text{ then } a &\hookrightarrow b \end{aligned}$$

Then a basic propositional attack relation can be alternatively described as a collective attack relation satisfying the rules $(A_{\wedge})$ and $(A_{\neg})$. Moreover, the global negation $\sim$ can be added to this system just by adding the corresponding postulates (AN) . It turns out, however, that the global negation is *eliminable* in this setting via to the following reductions:

$$\begin{aligned} a, \sim A &\hookrightarrow b \equiv a \hookrightarrow b, \neg A & a &\hookrightarrow \sim A, b \equiv a, \neg A \hookrightarrow b \\ a, \neg \sim A &\hookrightarrow b \equiv a \hookrightarrow b, A & a &\hookrightarrow \neg \sim A, b \equiv a, A \hookrightarrow b \end{aligned} \quad (R_{\sim})$$

²This connective coincides with the local four-valued negation from [Bochman, 1998].

As a result, the basic attack relation can be safely restricted to an attack relation in a classical language.

Finally, the rule Self-Defeat of causal argumentation gives a formal representation for an often expressed desideratum that self-conflicting arguments should not participate in defeating other arguments (see, e.g., [Bondarenko *et al.*, 1997]). This aim is achieved in our setting by requiring that such arguments are attacked even by tautologies, and hence by any argument whatsoever.

3.3 Argumentation vs. causal inference

Probative attack relations will now be shown to be equivalent to general production inference relations from [Bochman, 2003b; 2004a], a variant of input-output logics from [Makinson and van der Torre, 2000].

A *production inference relation* is a relation $\Rightarrow$ on the set of classical propositions satisfying the following rules:

(Strengthening) If $A \models B$ and $B \Rightarrow C$, then $A \Rightarrow C$;

(Weakening) If $A \Rightarrow B$ and $B \models C$, then $A \Rightarrow C$;

(And) If $A \Rightarrow B$ and $A \Rightarrow C$, then $A \Rightarrow B \wedge C$;

(Truth) $\mathbf{t} \Rightarrow \mathbf{t}$;

(Falsity) $\mathbf{f} \Rightarrow \mathbf{f}$.

A production rule $A \Rightarrow B$ can be informally interpreted as saying that A *produces*, or *explains* B . A characteristic property of production inference is that reflexivity $A \Rightarrow A$ does not hold for it. Production rules are extended to rules with sets of propositions in premises by requiring that $u \Rightarrow A$ holds for a set u of propositions iff $\bigwedge a \Rightarrow A$, for some finite $a \subseteq u$. $\mathcal{C}(u)$ will denote the set of propositions produced by u :

$$\mathcal{C}(u) = \{A \mid u \Rightarrow A\}$$

The production operator $\mathcal{C}$ plays much the same role as the usual derivability operator for consequence relations.

A production inference relation is called *basic*, if it satisfies

(Or) If $A \Rightarrow C$ and $B \Rightarrow C$, then $A \vee B \Rightarrow C$.

and *causal*, if it is basic and satisfies, in addition

(Coherence) If $A \Rightarrow \neg A$, then $A \Rightarrow \mathbf{f}$.

It has been shown in [Bochman, 2003b] that causal inference relations provide a complete description of the underlying logic of causal theories from [McCain and Turner, 1997] (see also [Giunchiglia *et al.*, 2004]).

It turns out that the binary semantics, introduced earlier, is appropriate also for interpreting production inference:

Definition 3.4. A rule $A \Rightarrow B$ is *valid* in a binary semantics $\mathcal{B}$ if, for any bimodel $(u, v) \in \mathcal{B}$, $A \in u$ only if $B \in v$.

As has been shown in [Bochman, 2004a], the above semantics is adequate for production inference relations. Moreover, the semantics for basic production inference can be obtained by restricting bimodels to world pairs (α, β) , while the semantics for causal inference is obtained by requiring, in addition, that (α, β) is a bimodel only if (α, α) is also a bimodel.

Now, the correspondence between probative argumentation and production inference can be established directly on the syntactic level using the following definitions:

$$A \Rightarrow B \equiv \neg B \hookrightarrow A; \quad (\text{PA})$$

$$A \hookrightarrow B \equiv B \Rightarrow \neg A. \quad (\text{AP})$$

Under these correspondences, the rules of a probative attack relation correspond precisely to the postulates for production relations. Moreover, the correspondence extends also to a correspondence between basic and causal argumentation, on the one hand, and basic and causal production inference, on the other. Hence the following result is straightforward.

Lemma 3.4. *If $\hookrightarrow$ is a probative [basic, causal] attack relation, then (PA) determines a [basic, causal] production inference relation, and vice versa, if $\Rightarrow$ is a [basic, causal] production inference relation, then (AP) determines a probative [basic, causal] attack relation.*

Remark. A seemingly more natural correspondence between propositional argumentation and production inference can be obtained using the following definitions:

$$A \Rightarrow B \equiv A \hookrightarrow \neg B \quad A \hookrightarrow B \equiv A \Rightarrow \neg B.$$

By these definitions, A explains B if it attacks $\neg B$, and vice versa, A attacks B if it explains $\neg B$. Unfortunately, this correspondence, though plausible by itself, does not take into account the intended understanding of arguments as (negative) assumptions. As a result, it cannot be extended directly to the correspondence between the associated nonmonotonic semantics, described below.

4 Nonmonotonic semantics

The correspondence between argumentation and causal reasoning has been established above on the level of underlying logical (monotonic) formalisms. In this section we will describe how this correspondence can be extended to the associated nonmonotonic semantics.

In [Bondarenko *et al.*, 1997], the assumption-based argumentation framework has been *instantiated* to capture existing nonmonotonic formalisms. In other words, it has been shown how particular nonmonotonic systems can be viewed as assumption-based frameworks just by defining assumptions and their contraries. We will show, however, that the propositional argumentation theory allows us to give a direct representation of Reiter's default logic [Reiter, 1980].

Given a system of propositional argumentation in the classical language augmented with the global negation $\sim$, we will interpret Reiter's default rule $a:b/A$ as an attack

$$a, \sim \neg b \hookrightarrow \sim A,$$

or, equivalently, as a rule $a, \sim \neg b \Vdash A$ of the associated Belnap consequence relation. Similarly, an axiom A of a default theory will be interpreted as an attack $t \hookrightarrow \sim A$. For a default theory Δ , we will denote by $tr(\Delta)$ the corresponding argument theory obtained by this translation.

By our general agreement, by *assumptions* we will mean propositions of the form $\sim A$, where A is a classical proposition. For a set u of classical propositions, we will denote by $\tilde{u}$ the set of assumptions $\{\sim A \mid A \notin u\}$. Finally, a set w of assumptions will be called *stable* in an argument theory Δ if, for any assumption A , $A \in w$ iff $w \not\vdash_{\Delta} A$. Then we have

Theorem 4.1. *A set u of classical propositions is an extension of a default theory Δ if and only if $\tilde{u}$ is a stable set of assumptions in $tr(\Delta)$.*

The above result is similar to the corresponding representation result in [Bondarenko *et al.*, 1997] (Theorem 3.10), but it is much simpler, and is formulated entirely in the framework of propositional attack relations. The simpler representation was made possible due to the fact that propositional attack relations already embody the deductive capabilities treated as an additional ingredient in assumption-based frameworks.

As our next result, we will establish a correspondence between the nonmonotonic semantics of causal inference relations and that of causal argumentation.

The nonmonotonic semantics of a causal inference relation is a set of its *exact worlds*, namely worlds α such that $\alpha = \mathcal{C}(\alpha)$ (see [Bochman, 2004a]). Such a world satisfies the rules of the causal relation, and any proposition that holds in it is explained by the causal rules.

A *causal theory* is an arbitrary set of production rules. By a nonmonotonic semantics of a causal theory Δ we will mean the exact worlds of the least causal relation containing Δ .

The correspondence between exact worlds and stable sets of assumptions is established in the next theorem.

Theorem 4.2. *If Δ is a causal theory, and Δ_a its corresponding argument theory given by (AP), then a world α is an exact world of Δ iff $\tilde{\alpha}$ is a stable set of assumptions in Δ_a .*

The above result shows, in effect, that propositional argumentation subsumes causal reasoning as a special case. Moreover, it can be shown that causal attack relations constitute a strongest argumentation system suitable for this kind of nonmonotonic semantics.

As an application of the above correspondence, we will describe now an alternative argumentation-based representation of logic programming.

A *general logic program* Π is a set of rules of the form

$$\text{not } d, c \leftarrow a, \text{not } b \quad (*)$$

where a, b, c, d are sets of propositional atoms. These are program rules of a most general kind that contain disjunctions and negations as failure **not** in their heads. As has been shown in [Bochman, 2004b], general logic programs are representable as causal theories obtained by translating program rules (*) as causal rules

$$d, \neg b \Rightarrow \bigwedge a \rightarrow \bigvee c,$$

and adding a formalization of the Closed World Assumption:

(Default Negation) $\neg p \Rightarrow \neg p$, for any propositional atom p .

Now, due to the correspondence between causal reasoning and argumentation, this causal theory can be transformed (using (PA)) into an argument theory that consists of attacks

$$a, \neg c \hookrightarrow \neg b, d \quad (\text{AL})$$

plus the 'argumentative' Closed World Assumption:

(Default Assumption) $p \hookrightarrow \neg p$, for any atom p .

Let $tr(\Pi)$ denote the argument theory obtained by this translation from a logic program Π . Then we obtain

Theorem 4.3. *A set u of propositional atoms is a stable model of a logic program Π iff $\tilde{u}$ is a stable set of assumptions in $tr(\Pi)$.*

It is interesting to note that, due to the reduction rules ($R_{\sim}$) for the global negation $\sim$, described earlier, the above representation (AL) of program rules is equivalent to $a, \sim b \hookrightarrow \sim c, d$, and therefore to the inference rules

$$a, \sim b \vdash c, \sim d$$

of the associated Belnap consequence relation. For normal logic programs, this latter representation coincides with that given in [Bondarenko *et al.*, 1997].

5 Conclusions

Two main objectives have been pursued in this study. The first consisted in showing that propositional argumentation suggests a viable and useful extension of an abstract argumentation theory that allows us to endow argumentation with full-fledged logical capabilities. It has been shown, in particular, that propositional argumentation subsumes assumption-based frameworks, and provides a direct representation of default logic, causal reasoning and logic programming. It could be expected that further development of this propositional approach to argumentation may bring us additional theoretical and practical benefits.

The second aim was to demonstrate that causal reasoning can be seen as an important kind of argumentation. It has been shown in this respect that causal inference relations and their semantics exactly correspond to a special, quite strong, kind of propositional attack relations and their associated stable semantics. This correspondence established a basic link between argumentation and causal reasoning, and it can be extended in both directions. To begin with, the argumentation theory has suggested a number of weaker semantic models, such as admissible sets and complete extensions (see [Bondarenko *et al.*, 1997; Dung, 1995a]), and it seems worth to inquire whether such models correspond to reasonable semantics for causal reasoning. On the other hand, a number of alternative models for causal reasoning has been suggested in [Geffner, 1992; 1997] (see also [Darwiche and Pearl, 1994]), and here it seems plausible to suppose that the correspondence between causal reasoning and argumentation could be helpful in analyzing and evaluating such models. These are, however, the topics for further study.

References

- [Belnap, 1977] N. D. Belnap, Jr. A useful four-valued logic. In M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 8–41. D. Reidel, 1977.
- [Bochman, 1998] A. Bochman. Biconsequence relations: A four-valued formalism of reasoning with inconsistency and incompleteness. *Notre Dame Journal of Formal Logic*, 39(1):47–73, 1998.
- [Bochman, 2003a] A. Bochman. Collective argumentation and disjunctive logic programming. *Journal of Logic and Computation*, 9:55–56, 2003.
- [Bochman, 2003b] A. Bochman. A logic for causal reasoning. In *Proceedings IJCAI’03*, Acapulco, 2003. Morgan Kaufmann.
- [Bochman, 2004a] A. Bochman. A causal approach to non-monotonic reasoning. *Artificial Intelligence*, 160:105–143, 2004.
- [Bochman, 2004b] A. Bochman. A causal logic of logic programming. In D. Dubois, C. Welty, and M.-A. Williams, editors, *Proc. Ninth Conference on Principles of Knowledge Representation and Reasoning, KR’04*, pages 427–437, Whistler, 2004.
- [Bondarenko *et al.*, 1997] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic framework for default reasoning. *Artificial Intelligence*, 93:63–101, 1997.
- [Chesñevar *et al.*, 2000] C. I. Chesñevar, A. G. Marguitman, and R. P. Loui. Logical models of argument. *ACM Computing Surveys*, 32:337–383, 2000.
- [Darwiche and Pearl, 1994] A. Darwiche and J. Pearl. Symbolic causal networks. In *Proceedings AAAI’94*, pages 238–244, 1994.
- [Dung, 1995a] P. M. Dung. An argumentation-theoretic foundation for logic programming. *J. of Logic Programming*, 22:151–177, 1995.
- [Dung, 1995b] P. M. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-persons games. *Artificial Intelligence*, 76:321–358, 1995.
- [Geffner, 1992] H. Geffner. *Default Reasoning. Causal and Conditional Theories*. MIT Press, 1992.
- [Geffner, 1997] H. Geffner. Causality, constraints and the indirect effects of actions. In *Proceedings Int. Joint Conf. on Artificial Intelligence, IJCAI’97*, pages 555–561, 1997.
- [Giunchiglia *et al.*, 2004] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153:49–104, 2004.
- [Kakas and Toni, 1999] A. C. Kakas and F. Toni. Computing argumentation in logic programming. *Journal of Logic and Computation*, 9:515–562, 1999.
- [Kowalski and Toni, 1996] R. A. Kowalski and F. Toni. Abstract argumentation. *Artificial Intelligence and Law*, 4:275–296, 1996.
- [Makinson and van der Torre, 2000] D. Makinson and L. van der Torre. Input/Output logics. *Journal of Philosophical Logic*, 29:383–408, 2000.
- [McCain and Turner, 1997] N. McCain and H. Turner. Causal theories of action and change. In *Proceedings AAAI-97*, pages 460–465, 1997.
- [Prakken and Vreeswijk, 2001] H. Prakken and G. Vreeswijk. Logics for defeasible argumentation. In D. M. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic*, volume 4, pages 219–318. Kluwer, 2nd edition, 2001.
- [Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

Reconstructing an Agent's Epistemic State from Observations

Richard Booth

Macquarie University
Dept. of Computing
Sydney NSW 2109 Australia
rbooth@ics.mq.edu.au

Alexander Nittka

University of Leipzig
Dept. of Computer Science
Leipzig 04109 Germany
nittka@informatik.uni-leipzig.de

Abstract

We look at the problem in belief revision of trying to make inferences about what an agent believed – or *will* believe – at a given moment, based on an observation of how the agent has responded to some sequence of previous belief revision inputs over time. We adopt a “reverse engineering” approach to this problem. Assuming a framework for iterated belief revision which is based on sequences, we construct a model of the agent that “best explains” the observation. Further considerations on this best-explaining model then allow inferences about the agent's epistemic behaviour to be made. We also provide an algorithm which computes this best explanation.

1 Introduction

The problem of belief revision, i.e., of how an agent should modify its beliefs about the world given some new information which possibly contradicts its current beliefs, is by now a well-established research area in AI [Gärdenfors, 1988]. Traditionally, the work in this area is done from the *agent's perspective*, being usually pre-occupied with constructing actual revision operators which the agent might use and with rationality postulates which constrain how these operators should behave. In this paper we change viewpoint and instead cast ourselves in the role of an *observer* of the agent. Imagine the following scenario. Suppose we are given some sequence $(\phi_1, \dots, \phi_n)$ of revision inputs which a particular agent, hereafter $\mathcal{A}$, has received over a certain length of time and suppose we are also given a sequence $(\theta_1, \dots, \theta_n)$ with the interpretation that following the i^{th} input ϕ_i , $\mathcal{A}$ believed (at least) θ_i . Throughout the paper, we make the assumptions that $\mathcal{A}$ received no input between ϕ_1 and ϕ_n other than those listed, and that the θ_i are *correct* (but possibly *partial*) descriptions of $\mathcal{A}$'s beliefs after each input. A couple of questions now suggest themselves:

- What will $\mathcal{A}$ believe following a *further* revision input ϕ_{n+1} ?
- What did $\mathcal{A}$ believe immediately *before* it received the first input ϕ_1 ? What, *apart* from θ_i did $\mathcal{A}$ believe after the i^{th} input ϕ_i ?

One area in which such questions might arise is in human-machine dialogues [del Cerro *et al.*, 1998], where the ϕ_i correspond to inputs given to $\mathcal{A}$ – the user – by a machine, and the θ_i are the user's responses. Here, it can be useful for the machine to keep a model of the evolution of a user's beliefs during a dialogue. Another setting where reconstruction of a prior belief set is needed is in law cases, where guilt and innocence often depend on who knew what, when. Inquiry is a possible method of approaching this type of problem. The above mentioned sequences can be seen as the information the inquiry yields and which is now used for drawing conclusions. Our aim in this paper is to answer these questions.

Our strategy for dealing with these questions will be to adopt a “reverse engineering” approach – constructing a model of the agent. (Similar approaches have already been tried in the context of trying to infer an agent's *goals* from observable *actions*, e.g., [Brafman and Tennenholtz, 1997; Lang, 2004].) Having no access to the agent's internals, we assume a belief revision framework $\mathcal{A}$ uses for determining its beliefs and for incorporating new information, and construct a model of $\mathcal{A}$ that explains the observation about it. By considering this model, we will then be able to make extra inferences or predictions about $\mathcal{A}$'s epistemic behaviour. Of course this raises the problem of which belief revision framework to choose. Such a framework will obviously need to support *iterated* revision [Darwiche and Pearl, 1997; Lehmann, 1995b; Nayak *et al.*, 2003], and preferably also *non-prioritised* revision [Hansson *et al.*, 2001; Makinson, 1997], i.e., revision in which new inputs are allowed to be *rejected*. In this paper, we restrict the investigation to one such framework that has been studied in [Booth, 2005]. The idea behind it is that an agent's epistemic state is made up of *two* components: (i) a sequence ρ of sentences representing the sequence of revision inputs the agent has received thus far, and (ii) a single sentence $\blacktriangle$ standing for the agent's set of *core* beliefs, which intuitively are those beliefs of the agent it considers “untouchable”. The agent's full set of beliefs in the state $[\rho, \blacktriangle]$ is then determined by a particular calculation on ρ and $\blacktriangle$, while new revision inputs are incorporated by simply appending them to the end of ρ . Note that our choice of this framework does not imply that others are less worthy of investigation. The challenge now becomes to find that *particular* model of this form which *best explains* the observation $o = \langle (\phi_1, \dots, \phi_n), (\theta_1, \dots, \theta_n) \rangle$ we have made of $\mathcal{A}$.

The plan of the paper is as follows. In Sect. 2 we describe in more detail the model of epistemic state we will be assuming. This will enable us to pose more precisely the problem we want to solve. We will see that the problem essentially reduces to trying to guess what $\mathcal{A}$'s *initial* epistemic state $[\rho, \blacktriangle]$ (i.e., before it received ϕ_1) was. In Sect. 3, inspired by work done on reasoning with *conditional beliefs*, we propose a way of finding the best initial sequence – or *prefix* – $\rho(\blacktriangle)$ for any given *fixed* $\blacktriangle$. Then, in Sect. 4 we focus on finding the best $\blacktriangle$. This will amount to equating best with logically weakest. The epistemic state $[\rho(\blacktriangle), \blacktriangle]$ obtained by combining our answers will be our proposed best explanation for o , which we will call the *rational explanation*. In Sect. 5 we present an algorithm which *constructs* the rational explanation for any given o , before giving some examples to show the type of inferences this explanation leads to in Sect. 6. In Sect. 7 we briefly mention a related piece of work by Dupin de Saint-Cyr and Lang, before concluding and giving some pointers for future research.

2 Modelling the Agent

We assume sentences $\phi_i, \theta_i, \blacktriangle$, etc. are elements of some finitely-generated propositional language L . In our examples, p, q, r denote distinct propositional variables. The classical logical entailment relation between sentences is denoted by $\vdash$, while $\equiv$ denotes classical logical equivalence. Wherever we use a sentence to describe a *belief set* the intention is that it represents all its logical consequences. The set of all possible observations $o = \langle \iota, \tau \rangle$ which can be made of $\mathcal{A}$, where $\iota = (\phi_1, \dots, \phi_n)$ and $\tau = (\theta_1, \dots, \theta_n)$ are two finite sequences of sentences of the same length, is denoted by O . The operation $\cdot$ on sequences denotes sequence concatenation.

As indicated in the introduction, we follow [Booth, 2005] by assuming that, at any given moment in time, an agent's epistemic state is represented by a pair $[\rho, \blacktriangle]$. ([Konieczny and Pérez, 2000; Lehmann, 1995b] also use sequences to represent epistemic states, but without core beliefs). In order to fully specify the agent's epistemic processes, we also need to formally specify (i) how the agent determines its set of beliefs $Bel([\rho, \blacktriangle])$ in any given state $[\rho, \blacktriangle]$, and (ii) how it incorporates new revision inputs into its epistemic state. Turning first to (i), we can describe $Bel([\rho, \blacktriangle])$ neatly with the help of a function f , which takes as argument a non-empty sequence $\sigma = (\alpha_m, \dots, \alpha_1)$ of sentences, and returns a sentence. f is defined by induction on the length m of σ : if $m = 1$ then $f(\sigma) = \alpha_1$. If $m > 1$ then

$$f(\sigma) = \begin{cases} \varphi = \alpha_m \wedge f(\alpha_{m-1}, \dots, \alpha_1) & \text{if } \varphi \not\vdash \perp \\ f(\alpha_{m-1}, \dots, \alpha_1) & \text{otherwise} \end{cases}$$

In other words $f(\sigma)$ is determined by first taking α_1 and then going backwards through σ , adding each sentence as we go, provided that sentence is consistent with what has been collected so far (cf. the “linear base-revision operation” of [Nebel, 1994] and the “basic memory operator” of [Konieczny and Pérez, 2000].) The belief set associated to the state $[\rho, \blacktriangle]$ is then given by $Bel([\rho, \blacktriangle]) = f(\rho \cdot \blacktriangle)$. Hence when calculating its beliefs from the sentences appearing in its epistemic state, an agent gives highest priority to $\blacktriangle$. After that, it prioritises more recent information received. Note

that $\blacktriangle$ is always believed, and that $Bel([\rho, \blacktriangle])$ is inconsistent if and only if $\blacktriangle$ is inconsistent.

Example 2.1. Consider $\blacktriangle = \neg p$ and $\rho = (q, q \rightarrow p)$. $Bel([\rho, \blacktriangle]) = f(q, q \rightarrow p, \neg p)$. In order to determine $f(q, q \rightarrow p, \neg p)$ we need to know if q is consistent with $f(q \rightarrow p, \neg p)$. As $f(\neg p) = \neg p$ and $q \rightarrow p$ is consistent with $\neg p$, $f(q \rightarrow p, \neg p) = (q \rightarrow p) \wedge \neg p \equiv \neg q \wedge \neg p$. So q is inconsistent with $f(q \rightarrow p, \neg p)$. Consequently we get $f(q, q \rightarrow p, \neg p) = f(q \rightarrow p, \neg p)$ and $Bel([\rho, \blacktriangle]) = f(q \rightarrow p, \neg p) \equiv \neg q \wedge \neg p$.

An agent incorporates a new revision input λ into its epistemic state $[\rho, \blacktriangle]$ by simply appending λ to ρ , i.e., the agent's *revision function* $*$ is specified by setting, for every $\lambda \in L$,

$$[\rho, \blacktriangle] * \lambda = [\rho \cdot \lambda, \blacktriangle].$$

Given this, we see that a new input λ will not always be believed in the new state. Indeed (when $\blacktriangle$ is consistent) it will be so only if it is consistent with $\blacktriangle$. If it contradicts $\blacktriangle$ then it will not be accepted, and in fact in this case the agent's belief set will remain unchanged (c.f. *screened revision* [Makinson, 1997]). Note also that $\blacktriangle$ remains unaffected by a revision input, i.e., $*$ is a *core-invariant* revision operator [Booth, 2005].¹ Core beliefs are needed to ensure that revision inputs can be rejected. If they were not allowed, which corresponds to demanding $\blacktriangle = \top$ in the above definitions, any consistent revision input would belong to the agent's beliefs.

As is shown in [Booth, 2005], the above revision method satisfies several natural properties. In particular, it stays largely faithful to the AGM postulates [Gärdenfors, 1988] (leaving aside the “success” postulate, which forces all new inputs to be accepted), and satisfies slight, “non-prioritised” variants of several postulates for iterated revision which have been proposed, including those of [Darwiche and Pearl, 1997]. One characteristic property of this method is the following variant of the rule “Recalcitrance” from [Nayak *et al.*, 2003]:

$$\text{If } \blacktriangle \not\vdash (\lambda_2 \rightarrow \neg \lambda_1) \text{ then } Bel([\rho, \blacktriangle] * \lambda_1 * \lambda_2) \vdash \lambda_1$$

This entails if the agent *accepts* an input λ_1 , then it does so *wholeheartedly*, in that the only way it can be dislodged from the belief set by a succeeding input λ_2 is if that input contradicts it given the core beliefs $\blacktriangle$.

Returning to our agent $\mathcal{A}$ from the introduction, **from now on we assume $\mathcal{A}$'s epistemic state is always of the form $[\rho, \blacktriangle]$, and that $\mathcal{A}$ determines its belief set and incorporates new inputs into its epistemic state as described above.** Then, suppose we make the observation $o = \langle (\phi_1, \dots, \phi_n), (\theta_1, \dots, \theta_n) \rangle$ about $\mathcal{A}$. Then after receiving the i^{th} input ϕ_i , $\mathcal{A}$'s epistemic state must be $[\rho \cdot (\phi_1, \dots, \phi_i), \blacktriangle]$ and its belief set $f(\rho \cdot (\phi_1, \dots, \phi_i) \cdot \blacktriangle)$, where $[\rho, \blacktriangle]$ is $\mathcal{A}$'s unknown *initial* (i.e., before ϕ_1) epistemic state. Observation o now amounts to the following:

$$f(\rho \cdot (\phi_1, \dots, \phi_i) \cdot \blacktriangle) \vdash \theta_i \quad i = 1, \dots, n \quad (1)$$

We make the following definitions:

Definition 2.2. Let $o = \langle (\phi_1, \dots, \phi_n), (\theta_1, \dots, \theta_n) \rangle \in O$. Then $[\rho, \blacktriangle]$ explains o (or is an explanation for o) iff (1) above holds. We say $\blacktriangle$ is an o -acceptable core iff $[\rho, \blacktriangle]$ explains o for some ρ .

¹In fact the model of [Booth, 2005] allows the core itself to be revisable. We do not explore this possibility here.

Example 2.3. (i) $[\rho, \blacktriangle] = [(p \rightarrow q), r]$ explains $\langle(p, q), (q, r)\rangle$ because $f(p \rightarrow q, p, r) \equiv p \wedge q \wedge r \vdash q$ and $f(p \rightarrow q, p, q, r) \equiv p \wedge q \wedge r \vdash r$.
(ii) $[(p \rightarrow q), \top]$ does not explain $\langle(p, q), (q, r)\rangle$ because $f(p \rightarrow q, p, q, \top) \equiv p \wedge q \not\vdash r$.

If we had some explanation $[\rho, \blacktriangle]$ for o then we would be able to answer the questions in the introduction: following a new input ϕ_{n+1} $\mathcal{A}$ will believe $f(\rho \cdot (\phi_1, \dots, \phi_n, \phi_{n+1}) \cdot \blacktriangle)$, before receiving the first input $\mathcal{A}$ believes $f(\rho \cdot \blacktriangle)$, and the beliefs after the i^{th} input are $f(\rho \cdot (\phi_1, \dots, \phi_i) \cdot \blacktriangle)$.

Note for any $o \in O$ there always exists *some* explanation $[\rho, \blacktriangle]$ for o , since the contradiction $\perp$ is an o -acceptable core using *any* ρ . But this would be a most unsatisfactory explanation, since it means we just infer $\mathcal{A}$ believes everything at every step.

Our job now is to choose, from the space of possible explanations for o , the best one. As a guideline, we consider an explanation good if it only makes necessary (or minimal) assumptions about what $\mathcal{A}$ believes. But how do we find this best one? Our strategy is to split the problem into two parts, handling ρ and $\blacktriangle$ separately. First, (i) given a *fixed* o -acceptable core $\blacktriangle$, find a best sequence $\rho(o, \blacktriangle)$ such that $[\rho, \blacktriangle]$ explains o , then, (ii) find a best o -acceptable core $\blacktriangle(o)$. Our best explanation for o will then be $[\rho(o, \blacktriangle(o)), \blacktriangle(o)]$.

3 Finding ρ

Given $o = \langle(\phi_1, \dots, \phi_n), (\theta_1, \dots, \theta_n)\rangle$, let us assume a fixed core $\blacktriangle$. To find that sequence $\rho(o, \blacktriangle)$ such that $[\rho(o, \blacktriangle), \blacktriangle]$ is the best explanation for o , *given* $\blacktriangle$, we will take inspiration from work done in the area of non-monotonic reasoning on reasoning with *conditional* information.

Let's say a pair (λ, χ) of sentences is a *conditional belief* in the state $[\rho, \blacktriangle]$ iff χ would be believed after revising $[\rho, \blacktriangle]$ by λ , i.e., $Bel([\rho, \blacktriangle] * \lambda) \vdash \chi$. In this case we will write $\lambda \Rightarrow_{[\rho, \blacktriangle]} \chi$.² This relation plays an important role, because it turns out $\mathcal{A}$'s beliefs following *any* sequence of revision inputs starting from $[\rho, \blacktriangle]$ is determined *entirely* by the set $\Rightarrow_{[\rho, \blacktriangle]}$ of conditional beliefs in $[\rho, \blacktriangle]$. This is because, for *any* sequence of revision inputs $\phi_1, \dots, \phi_m$, our revision method satisfies

$$Bel([\rho, \blacktriangle] * \phi_1 * \dots * \phi_m) = Bel([\rho, \blacktriangle] * f(\phi_1, \dots, \phi_m, \blacktriangle)).$$

Thus, as far as their effects on the belief set go, a sequence of revision inputs starting from $[\rho, \blacktriangle]$ can always be reduced to a single input. (But note the set of conditional beliefs $\Rightarrow_{[\rho, \blacktriangle]} \lambda$ in the state $[\rho, \blacktriangle] * \lambda$ following revision by λ will generally *not* be the same as $\Rightarrow_{[\rho, \blacktriangle]} \lambda$.)

All this means observation o may be translated into a partial description of the set of conditional beliefs that $\mathcal{A}$ has in its initial epistemic state:

$$C_{\blacktriangle}(o) = \{f(\phi_1, \dots, \phi_i, \blacktriangle) \Rightarrow \theta_i \mid i = 1, \dots, n\}.$$

Clearly, if we had access to the *complete* set of $\mathcal{A}$'s conditional beliefs in its initial state, this would give another way to answer the questions of the introduction. Now, the problem of determining which conditional beliefs *follow from* a given set

²The relation $\Rightarrow_{[\rho, \blacktriangle]}$ *almost* satisfies all the rules of a rational inference relation [Lehmann and Magidor, 1992]. More precisely the modified version does, viz., $\lambda \Rightarrow_{[\rho, \blacktriangle]} \chi$ iff $[\blacktriangle \vdash \neg \lambda \text{ or } \lambda \Rightarrow_{[\rho, \blacktriangle]} \chi]$.

C of such beliefs has been well-studied and several solutions have been proposed, e.g., [Geffner and Pearl, 1992; Lehmann, 1995a]. One particularly elegant and well-motivated solution is to take the *rational closure* of C [Lehmann and Magidor, 1992]. Furthermore, as is shown in, e.g., [Freund, 2004], this construction is amenable to a relatively simple representation as a sequence of sentences! Our idea is essentially to take $\rho(o, \blacktriangle)$ to be this sequence corresponding to the rational closure of $C_{\blacktriangle}(o)$. First let us describe the general construction.

3.1 The rational closure of a set of conditionals

Given a set of conditionals $C = \{\lambda_i \Rightarrow \chi_i \mid i = 1, \dots, l\}$ we denote by $\tilde{C}$ the set of *material counterparts* of all the conditionals in C , i.e., $\tilde{C} = \{\lambda_i \rightarrow \chi_i \mid i = 1, \dots, l\}$. Then a sentence ν is *exceptional* for C iff $\tilde{C} \vdash \neg \nu$, and a conditional $\nu \Rightarrow \mu$ is exceptional for C iff its antecedent ν is. To find the (sequence corresponding to the) rational closure $\rho_R(C)$ of C , we first define a decreasing sequence of sets of conditionals $C_0 \supseteq C_1 \supseteq \dots \supseteq C_m$ by setting (i) $C_0 = C$, (ii) C_{i+1} equals the set of conditionals in C_i which are exceptional for C_i , and (iii) m is minimal such that $C_m = C_{m+1}$. Then we set

$$\rho_R(C) = (\bigwedge \tilde{C}_m, \bigwedge \tilde{C}_{m-1}, \dots, \bigwedge \tilde{C}_0).$$

Writing α_i for $\bigwedge \tilde{C}_i$, the rational closure of C is then the relation $\Rightarrow_R$ given by $\lambda \Rightarrow_R \chi$ iff either $\alpha_m \vdash \neg \lambda$ or $[\alpha_j \wedge \lambda \vdash \chi]$ where j is minimal such that $\alpha_j \not\vdash \neg \lambda$. Since $\alpha_m \vdash \dots \vdash \alpha_0$ it is easy to check that in fact this second disjunct is equivalent to $f(\alpha_m, \dots, \alpha_0, \lambda) \vdash \chi$.

We now make the following definition:

Definition 3.1. Let $o \in O$ and $\blacktriangle \in L$. We call $\rho_R(C_{\blacktriangle}(o))$ the rational prefix of o with respect to $\blacktriangle$, and will denote it by $\rho_R(o, \blacktriangle)$.

Example 3.2. Let $o = \langle(p, q), (r, \neg p)\rangle$ and $\blacktriangle = \neg p$. Then

$$\begin{aligned} C_{\blacktriangle}(o) &= \{f(p, \neg p) \Rightarrow r, f(p, q, \neg p) \Rightarrow \neg p\} \\ &= \{\neg p \Rightarrow r, (q \wedge \neg p) \Rightarrow \neg p\}. \end{aligned}$$

Since neither of the individual conditionals are exceptional for $C_{\blacktriangle}(o)$ we get $C_0 = C_{\blacktriangle}(o)$ and $C_1 = \emptyset$. Clearly then also $C_2 = \emptyset = C_1$ so we obtain $\rho_R(o, \blacktriangle) = (\bigwedge \emptyset, \bigwedge \tilde{C}_{\blacktriangle}(o))$. Rewriting the sequence using logically equivalent sentences we get $\rho_R(o, \blacktriangle) = (\top, \neg p \rightarrow r)$.

Now, an interesting thing to note about the rational prefix construction is that it actually goes through *independently* of whether $\blacktriangle$ is o -acceptable. In fact a useful side-effect of the construction is that it actually *reveals* whether $\blacktriangle$ is o -acceptable. Given we have constructed $\rho_R(o, \blacktriangle) = (\alpha_m, \dots, \alpha_0)$, all we have to do is to look at sentence α_m and check if it is a tautology:

Proposition 3.3. Let $o \in O$ and $\blacktriangle \in L$, and let $\rho_R(o, \blacktriangle) = (\alpha_m, \dots, \alpha_0)$ be the rational prefix of o w.r.t. $\blacktriangle$. Then

- (i) if $\alpha_m \equiv \top$ then $[\rho_R(o, \blacktriangle), \blacktriangle]$ is an explanation for o .
- (ii) if $\alpha_m \not\equiv \top$ then $\blacktriangle$ is not an o -acceptable core.

Thus this proposition gives us a necessary and sufficient condition for $\blacktriangle$ to be an o -acceptable core. This will be used in the algorithm of Sect. 5.

In Example 3.2 $\rho_R(o, \blacktriangle) = (\top, \neg p \rightarrow r)$ was calculated. The above proposition implies $[(\top, \neg p \rightarrow r), \neg p]$

is an explanation for $o = \langle (p, q), (r, \neg p) \rangle$. This is verified by $f(\top, \neg p \rightarrow r, p, \neg p) \equiv \neg p \wedge r \vdash r$ and $f(\top, \neg p \rightarrow r, p, q, \neg p) \equiv \neg p \wedge q \wedge r \vdash \neg p$.

3.2 Justification for using the rational prefix

In the rest of this section we assume $\blacktriangle$ to be some fixed *o*-acceptable core. As we just saw, $[\rho_R(o, \blacktriangle), \blacktriangle]$ then provides an explanation for o given this $\blacktriangle$. In this section we want to show in precisely what sense it could be regarded as a *best* explanation given $\blacktriangle$. Let $\Sigma = \{\sigma \mid [\sigma, \blacktriangle] \text{ explains } o\}$.

One way to compare sequences in Σ is by focusing on the *trace* of belief sets they (in combination with $\blacktriangle$) induce through o , i.e., for each $\sigma \in \Sigma$ we can consider the sequence $(Bel_0^\sigma, Bel_1^\sigma, \dots, Bel_n^\sigma)$, where Bel_i^σ is defined to be the beliefs after the i^{th} input in o (under the explanation $[\sigma, \blacktriangle]$). In other words $Bel_i^\sigma = f(\sigma \cdot (\phi_1, \dots, \phi_i) \cdot \blacktriangle)$. (So Bel_0^σ gives the initial belief set.)

Example 3.4. Let $o, \blacktriangle$ and $\rho_R(o, \blacktriangle)$ be as in Example 3.2. Then the belief trace is $(\neg p \wedge r, \neg p \wedge r, \neg p \wedge q \wedge r)$.

The idea would then be to define a preference relation $\preceq_1$ over the sequences in Σ (with more preferred sequences corresponding to those “lower” in the ordering) via some preference relation over their set of associated belief traces. Given any two possible belief traces $(\beta_0, \dots, \beta_n)$ and $(\gamma_0, \dots, \gamma_n)$, let us write $(\beta_0, \dots, \beta_n) \leq_{\text{lex}} (\gamma_0, \dots, \gamma_n)$ iff, for all $i = 0, \dots, n$, $[\beta_j \equiv \gamma_j \text{ for all } j < i \text{ implies } \gamma_i \vdash \beta_i]$. Then we define, for any $\rho, \sigma \in \Sigma$:

$$\rho \preceq_1 \sigma \text{ iff } (Bel_0^\rho, \dots, Bel_n^\rho) \leq_{\text{lex}} (Bel_0^\sigma, \dots, Bel_n^\sigma).$$

($\preceq_1$ is a pre-order (i.e., reflexive and transitive) on Σ .) Thus, given two sequences in Σ , we prefer that one which leads to $\mathcal{A}$ having fewer (i.e., weaker) beliefs before any of the inputs ϕ_i were received. If the two sequences lead to equivalent beliefs at this initial stage, then we prefer that which leads to $\mathcal{A}$ having fewer beliefs after ϕ_1 was received. If they lead to equivalent beliefs also after this stage, then we prefer that which leads to $\mathcal{A}$ having fewer beliefs after ϕ_2 was received, and so on. Thus, under this ordering, we prefer sequences which induce $\mathcal{A}$ to have *fewer* beliefs, *earlier* in o . The next result shows $\rho_R(o, \blacktriangle)$ is a best element in Σ under this ordering.

Proposition 3.5. $\rho_R(o, \blacktriangle) \preceq_1 \sigma$ for all $\sigma \in \Sigma$.

Another way to compare sequences is to look at their consequences for predicting what will happen at the next step after o .

$$\rho \preceq_2 \sigma \text{ iff } Bel([\sigma, \blacktriangle] * \phi_1 * \dots * \phi_n * \lambda) \vdash Bel([\rho, \blacktriangle] * \phi_1 * \dots * \phi_n * \lambda) \text{ for all } \lambda$$

Thus, according to *this* preference criterion we prefer ρ to σ if it always leads to fewer beliefs being predicted after the next revision input. It turns out $\rho_R(o, \blacktriangle)$ is a most preferred element under $\preceq_2$ amongst all minimal elements under $\preceq_1$.

Proposition 3.6. For all $\sigma \in \Sigma$, if $\sigma \preceq_1 \rho_R(o, \blacktriangle)$ then $\rho_R(o, \blacktriangle) \preceq_2 \sigma$.

Thus if we take a lexicographic combination of $\preceq_1$ and $\preceq_2$ (with $\preceq_1$ being considered as more important), $\rho_R(o, \blacktriangle)$ emerges overall as a best, most preferred, member of Σ . Having provided a method for finding the best explanation $[\rho, \blacktriangle]$ given $\blacktriangle$, we now turn our attention to finding the best $\blacktriangle$ itself.

4 Minimising $\blacktriangle$

As argued earlier, core beliefs are needed, but at the same time we try to minimise the assumptions about the agent’s beliefs. This includes minimising $\blacktriangle$. The first idea would be to simply take the disjunction of all possible *o*-acceptable cores, i.e., to take $\blacktriangle_\vee(o)$, defined by

$$\blacktriangle_\vee(o) \equiv \bigvee \{ \blacktriangle \mid \blacktriangle \text{ is an } o\text{-acceptable core} \}.$$

But is $\blacktriangle_\vee(o)$ itself *o*-acceptable? Thankfully the answer is yes, a result which follows (in our finite setting) from the following proposition which says that the family of *o*-acceptable cores is closed under disjunctions.

Proposition 4.1. If $\blacktriangle_1$ and $\blacktriangle_2$ are *o*-acceptable then so is $\blacktriangle_1 \vee \blacktriangle_2$.

So as a corollary $\blacktriangle_\vee(o)$ does indeed satisfy:

(Acceptability) $\blacktriangle_\vee(o)$ is an *o*-acceptable core

What other properties does $\blacktriangle_\vee(o)$ satisfy? Clearly, $\blacktriangle_\vee(o)$ will always be consistent provided at least one consistent *o*-acceptable core exists:

(Consistency) If $\blacktriangle(o) \equiv \perp$ then $\blacktriangle' \equiv \perp$ for every *o*-acceptable core $\blacktriangle'$.

Acceptability and Consistency would appear to be absolute rock-bottom properties which we would expect of *any* method for finding a good *o*-acceptable core. However for $\blacktriangle_\vee$ we can say more. Given two observations $o = \langle \iota, \tau \rangle$ and $o' = \langle \iota', \tau' \rangle$, let us denote by $o \cdot o'$ the concatenation of o and o' , i.e., $o \cdot o' = \langle \iota \cdot \iota', \tau \cdot \tau' \rangle$. We shall use $o \sqsubseteq_{\text{right}} o'$ to denote that o' *right extends* o , i.e., $o' = o \cdot o''$ for some (possibly empty) $o'' \in O$, and $o \sqsubseteq_{\text{left}} o'$ to denote o' *left extends* o , i.e., $o' = o'' \cdot o$ for some (possibly empty) $o'' \in O$.

Proposition 4.2. Suppose $o \sqsubseteq_{\text{right}} o'$ or $o \sqsubseteq_{\text{left}} o'$. Then every *o'*-acceptable core is an *o*-acceptable core.

As a result of this we see $\blacktriangle_\vee$ satisfies the following 2 properties, which say extending the observation into the future or past leads only to a logically stronger core being returned.

(Right Monotony) If $o \sqsubseteq_{\text{right}} o'$ then $\blacktriangle(o') \vdash \blacktriangle(o)$

(Left Monotony) If $o \sqsubseteq_{\text{left}} o'$ then $\blacktriangle(o') \vdash \blacktriangle(o)$.

Right- and Left Monotony provide ways of expressing that $\blacktriangle(o)$ leads only to *safe* conclusions that something is a core belief of $\mathcal{A}$ – conclusions that cannot be “defeated” by additional information about $\mathcal{A}$ that might come along in the form of observations prior to, or after o .

We should point out, though, that it is *not* the case that by inserting any observation *anywhere* in o , $\blacktriangle_\vee$ will always lead to a logically stronger core. Consider $o_1 = \langle (p, q), (p, \neg p) \rangle$ and $o_2 = \langle (p, \neg p, q), (p, \neg p, \neg p) \rangle$, i.e., $\langle (\neg p), (\neg p) \rangle$ was inserted in the middle of o_1 . $\blacktriangle_\vee(o_1) \equiv q \rightarrow \neg p$ whereas $\blacktriangle_\vee(o_2) \equiv \top$. So although o_2 extends o_1 in a sense, the corresponding $\blacktriangle_\vee$ is actually weaker. Looking at o_1 , assuming as we do that $\mathcal{A}$ received *no* inputs between p and q , the *only* way to explain the end belief in $\neg p$ is to ascribe core belief $q \rightarrow \neg p$ to $\mathcal{A}$ (cf. the “Recalcitrance” rule in Sect. 2). However, looking at o_2 , the information that $\mathcal{A}$ received (and accepted) intermediate input $\neg p$ is enough to “explain away” this end belief without recourse to core beliefs. Our assumption that $\mathcal{A}$ received no other inputs between ϕ_1 and ϕ_n during an observation $o = \langle (\phi_1, \dots, \phi_n), (\theta_1, \dots, \theta_n) \rangle$ is rather

strong. It amounts to saying that, during o , we kept our eye on $\mathcal{A}$ the whole time. The above example shows that relaxing this assumption gives us an extra degree of freedom with which to explain o , via the inference of intermediate inputs. This will be a topic for future work.

It turns out the above four properties are enough to actually characterise $\Delta_{\vee}$. In fact, given the first two, just *one* of Right- and Left Monotony is sufficient for this task:

Proposition 4.3. *Let $\Delta : O \rightarrow L$ be any function which returns a sentence given any $o \in O$. Then the following are equivalent:*

- (i) Δ satisfies Acceptability, Consistency and Right Monotony.
- (ii) Δ satisfies Acceptability, Consistency and Left Monotony.
- (iii) $\Delta(o) \equiv \Delta_{\vee}(o)$ for all $o \in O$.

Note that as a corollary to this proposition we get the surprising result that, in the presence of Acceptability and Consistency, Right- and Left Monotony are in fact equivalent.

Combining the findings of the last two sections, we are now ready to announce our candidate for the best explanation for o . By analogy with “rational closure”, we make the following definition:

Definition 4.4. *Let $o \in O$ be an observation. Then we call $[\rho_R(o, \Delta_{\vee}(o)), \Delta_{\vee}(o)]$ the rational explanation for o .*

In Sect. 6 we will give some examples of what we can infer about $\mathcal{A}$ under the rational explanation. But how might we find it in practice? The next section gives an algorithm for just that.

5 Constructing the Rational Explanation

The idea behind the algorithm is as follows. Given an observation o , we start with the weakest possible core $\Delta_0 = \top$ and construct the rational prefix $(\alpha_m, \dots, \alpha_0) = \rho_0$ of o w.r.t. Δ_0 . We then check whether α_m is a tautology. If it is then we know by Prop. 3.3 that $[\rho_0, \Delta_0]$ is an explanation for o and so we stop and return this as output. If it isn’t then Prop. 3.3 tells us Δ_0 cannot be o -acceptable. In this case, we modify Δ_0 by conjoining α_m to it, i.e., by setting $\Delta_1 = \Delta_0 \wedge \alpha_m$. Constructing the rational prefix of o w.r.t. the new core then leads to a *different* prefix, which can be dealt with the same way.

Algorithm 1 Calculation of the rational explanation

Input: observation o

Output: the rational explanation for o

```

 $\Delta \leftarrow \top$ 
repeat
   $\rho \leftarrow \rho_R(o, \Delta)$        $\{\rho = (\alpha_m, \dots, \alpha_0)\}$ 
   $\Delta \leftarrow \Delta \wedge \alpha_m$ 
until  $\alpha_m \equiv \top$ 
Return  $[\rho, \Delta]$ 

```

Before showing that the output of this algorithm matches the rational explanation, we need to be sure it always terminates. This is a consequence of the following:

Lemma 5.1. *Let Δ and α_m be as after the calculation of $\rho_R(o, \Delta)$. If $\alpha_m \not\equiv \top$ then $\Delta \not\equiv \Delta \wedge \alpha_m$.*

This result assures us that if the termination condition of the algorithm does not hold, the new core will be *strictly* logically stronger than the previous one. Thus the cores generated by the algorithm become progressively strictly stronger.

In our setting, in which we assumed a *finite* propositional language, this means, in the worst case, the process will continue until $\Delta \equiv \perp$. However in this case it can be shown the rational prefix of o w.r.t. $\perp$ is just $(\top)$, and so the termination condition will be satisfied at the very next step.

Now, to show the output matches the rational explanation, consider the sequence $[\rho_0, \Delta_0], \dots, [\rho_k, \Delta_k]$ of episodic states generated by the algorithm. We need to show $\Delta_k \equiv \Delta_{\vee}(o)$. The direction $\Delta_k \vdash \Delta_{\vee}(o)$ follows from the fact that $[\rho_k, \Delta_k]$ is an explanation for o and so Δ_k is an o -acceptable core. The converse $\Delta_{\vee}(o) \vdash \Delta_k$ is proved by showing inductively that $\Delta_{\vee}(o) \vdash \Delta_i$ for each $i = 0, \dots, k$: the case $i = 0$ clearly holds since $\Delta_0 \equiv \top$. The inductive step uses the following property:

Lemma 5.2. *Let $0 < i \leq k$ and suppose $\rho_{i-1} = (\alpha_m, \dots, \alpha_0)$. Then, for any o -acceptable core Δ' , if $\Delta' \vdash \Delta_{i-1}$ then $\Delta' \vdash \alpha_m$.*

This enables us to prove that, given $\Delta_{\vee}(o) \vdash \Delta_{i-1}$, we must also have $\Delta_{\vee}(o) \vdash \Delta_i$. Thus $\Delta_{\vee}(o) \vdash \Delta_k$ as required. Since obviously ρ_k is the rational prefix of o w.r.t. Δ_k by construction, we have:

Proposition 5.3. *Given input observation o , the algorithm outputs the rational explanation for o .*

Example 5.4. Let $o = \langle (p, q), (r, \neg p) \rangle$. Starting with $\Delta = \top$, in the first run $\tilde{C}_0 = \{f(p, \top) \rightarrow r, f(p, q, \top) \rightarrow \neg p\} = \{p \rightarrow r, p \wedge q \rightarrow \neg p\}$. Only the second conditional is exceptional, so $\tilde{C}_1 = \{p \wedge q \rightarrow \neg p\}$. Now the remaining conditional is exceptional for itself, so $\tilde{C}_2 = \tilde{C}_1$. Δ is updated to $\Delta \equiv p \rightarrow \neg q$ because $\rho = (p \rightarrow \neg q, p \rightarrow (r \wedge \neg q))$.

The next calculation yields $\tilde{C}_0 = \{f(p, p \rightarrow \neg q) \rightarrow r, f(p, q, p \rightarrow \neg q) \rightarrow \neg p\} = \{p \wedge \neg q \rightarrow r, q \wedge \neg p \rightarrow \neg p\}$. This time none of the conditionals are exceptional, so $\tilde{C}_1 = \emptyset$. As this means $\alpha_1 = \top$, no further run is necessary and the result is $\rho = (\top, (p \wedge \neg q) \rightarrow r)$, $\Delta = p \rightarrow \neg q$. $[(\top, (p \wedge \neg q) \rightarrow r), p \rightarrow \neg q]$ is the rational explanation for o .

6 Some Examples

In this section we want to give a few simple examples to illustrate the rational explanation.

For $o = \langle (p), (q) \rangle$, the rational explanation is $[(\top, p \rightarrow q), \top]$. So we infer $\mathcal{A}$ ’s initial belief set is $p \rightarrow q$. Indeed to explain $\mathcal{A}$ ’s belief in q following receipt of p it is clear $\mathcal{A}$ *must* initially believe *at least* $p \rightarrow q$ since p itself does not entail q . It seems fair to say we are not justified in ascribing to $\mathcal{A}$ any initial beliefs beyond this. After $\mathcal{A}$ receives p we assume $\mathcal{A}$ *accepts* this input – we have no reason to expect otherwise – and so has belief set $p \wedge q$. If $\mathcal{A}$ is given a *further* input $\neg(p \wedge q)$ we predict $\mathcal{A}$ will also accept this input, but will hold on to its belief in p . The reason being we assume $\mathcal{A}$, having only just been told p , now has stronger reasons to believe p than q . If, instead, $\mathcal{A}$ is given further input $\neg p$ we predict its belief set will be just $\neg p$, i.e., we do *not* assume $\mathcal{A}$ ’s belief in q persists. Essentially the rational explanation assumes the prior input p must have been *responsible* for $\mathcal{A}$ ’s prior belief in q . And with this input now being “overruled” by the succeeding input, $\mathcal{A}$ can no longer draw any conclusions about the truth of q .

Another illustrative example is $o = \langle (p), (\neg p) \rangle$, for which the rational explanation is $[(\top), \neg p]$. Indeed $\neg p$ must be a

core belief, as that is the only possibility for p to be rejected. And if p was not rejected, the agent could not consistently believe $\neg p$.

In some cases the rational explanation gives only the trivial explanation, i.e., $\Delta_{\vee}(o) \equiv \perp$. One of the simplest examples extends the prior one: $o = \langle (p, \neg p), (\neg p, p) \rangle$. The first part of the observation tells us that $\neg p$ must be a core belief, but when confirmed of that belief $\mathcal{A}$ changes its opinion. This behaviour of always believing the opposite of what one is told can be called rational only in very specific circumstances that are not in the scope of this investigation. Hence, failing to provide a satisfactory explanation for this example is not to be seen as a failure of the method.

7 Conclusion

Before concluding, one paper which deserves special mention as having similarities with the present one is [de Saint-Cyr and Lang, 2002] on *belief extrapolation* (itself an instance of the general framework of [Friedman and Halpern, 1999]). A belief extrapolation operator takes as input a sequence of sentences representing a sequence of partial observations of a possibly changing world, and outputs another sequence which “completes” the input. These operators proceed by trying to determine some history of the world which “best fits” (according to various criteria) the observations. A fundamental difference between that work and ours is that belief extrapolation is, like traditional operators of revision and update, an “agent’s perspective” operator – it is concerned with how an agent should form a picture of how the external world is evolving, whereas we are interested in forming a picture of how an *observed agent’s beliefs* are evolving. Nevertheless the precise connections between these two works seems worthy of further study.

To conclude, in this paper we made an attempt at reconstructing an agent’s initial epistemic state in order to explain a given observation of the agent and make predictions about its future beliefs. We did so by assuming a simple yet powerful model for epistemic states allowing for iterated non-prioritised revision. The algorithm we provided constructs a best explanation based on the rational closure of conditional beliefs. This answer should be applicable to problems requiring the modelling of agents’ beliefs, for example in the area of user modelling.

Generalisations of this approach which are object of future work include (i) relaxing the assumption that we are given an unbroken sequence of revision inputs, i.e., allowing also for intermediate inputs as an explanation for what the core accounts for now, (ii) allowing our observations to incorporate information about what the agent did *not* believe after a given revision step, and (iii) allowing the core beliefs to be revised. Further, it is of interest to compare our results with what other models of epistemic states would yield as explanation.

Acknowledgements

Thanks are due to the reviewers for helpful comments. A.N. acknowledges support by the EC (IST-2001-37004, WASP).

References

[Booth, 2005] R. Booth. On the logic of iterated non-prioritised revision. In *Conditionals, Information and In-*

ference – Selected papers from the Workshop on Conditionals, Information and Inference, 2002, pages 86–107. Springer’s LNAI 3301, 2005.

[Brafman and Tennenholtz, 1997] R. I. Brafman and M. Tennenholtz. Modeling agents as qualitative decision makers. *Artificial Intelligence*, 94(1-2):217–268, 1997.

[Darwiche and Pearl, 1997] A. Darwiche and J. Pearl. On the logic of iterated belief revision. *Artificial Intelligence*, 89:1–29, 1997.

[de Saint-Cyr and Lang, 2002] F. Dupin de Saint-Cyr and J. Lang. Belief extrapolation (or how to reason about observations and unpredicted change). In *Proceedings of KR’02*, pages 497–508, 2002.

[del Cerro *et al.*, 1998] L. Fariñas del Cerro, A. Herzig, D. Longin, and O. Rifi. Belief reconstruction in cooperative dialogues. In *Proceedings of AIMSA’98*, pages 254–266. Springer’s LNCS 1480, 1998.

[Freund, 2004] M. Freund. On the revision of preferences and rational inference processes. *Artificial Intelligence*, 152(1):105–137, 2004.

[Friedman and Halpern, 1999] N. Friedman and J. Halpern. Modeling belief in dynamic systems, part II: Revision and update. *Journal of Artificial Intelligence Research*, 10:117–167, 1999.

[Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux*. MIT Press, 1988.

[Geffner and Pearl, 1992] H. Geffner and J. Pearl. Conditional entailment: Bridging two approaches to default entailment. *Artificial Intelligence*, 53:209–244, 1992.

[Hansson *et al.*, 2001] S. O. Hansson, E. Fermé, J. Cantwell, and M. Falappa. Credibility-limited revision. *Journal of Symbolic Logic*, 66(4):1581–1596, 2001.

[Konieczny and Pérez, 2000] S. Konieczny and R. Pino Pérez. A framework for iterated revision. *Journal of Applied Non-Classical Logics*, 10(3-4):339–367, 2000.

[Lang, 2004] J. Lang. A preference-based interpretation of other agents’ actions. In *Proceedings of KR’04*, pages 644–653, 2004.

[Lehmann and Magidor, 1992] D. Lehmann and M. Magidor. What does a conditional knowledge base entail? *Artificial Intelligence*, 55(1):1–60, 1992.

[Lehmann, 1995a] D. Lehmann. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence*, 15(1):61–82, 1995.

[Lehmann, 1995b] D. Lehmann. Belief revision, revised. In *Proceedings of IJCAI’95*, pages 1534–1540, 1995.

[Makinson, 1997] D. Makinson. Screened revision. *Theoria*, 63:14–23, 1997.

[Nayak *et al.*, 2003] A. Nayak, M. Pagnucco, and P. Peppas. Dynamic belief revision operators. *Artificial Intelligence*, 146:193–228, 2003.

[Nebel, 1994] B. Nebel. Base revision operations and schemes: Semantics, representation and complexity. In *Proceedings of ECAI’94*, pages 342–345, 1994.

Scale-Based Monotonicity Analysis in Qualitative Modelling with Flat Segments

Martin Brooks¹ and Yuhong Yan¹ and Daniel Lemire²

¹National Research Council
200 Montreal Road, M-50
Ottawa, ON K1A 0R6
martin.brooks@nrc.gc.ca
yuhong.yan@nrc.gc.ca

²University of Quebec
4750 avenue Henri-Julien
Montréal, QC H2T 3E4
lemire@ondelette.com

Abstract

Qualitative models are often more suitable than classical quantitative models in tasks such as Model-based Diagnosis (MBD), explaining system behavior, and designing novel devices from first principles. Monotonicity is an important feature to leverage when constructing qualitative models. Detecting monotonic pieces robustly and efficiently from sensor or simulation data remains an open problem. This paper presents scale-based monotonicity: the notion that monotonicity can be defined relative to a scale. Real-valued functions defined on a finite set of reals e.g. sensor data or simulation results, can be partitioned into quasi-monotonic segments, i.e. segments monotonic with respect to a scale, in linear time. A novel segmentation algorithm is introduced along with a scale-based definition of “flatness”.

1 Introduction

Qualitative models are used in applications such as model-based diagnosis [Yan, 2003; Struss, 2002], explaining system behavior [Šuc, 2003; Forbus, 1984; Kuipers, 1986], and designing novel devices from first principles [Williams, 1992]. It is a challenge to build qualitative models for complex real world engineering systems. Current research efforts are on automatic generation of qualitative models from numerical data obtained by numerical simulation or sensors. Many proposed methods, such as in [Struss, 2002] and [Console *et al.*, 2003], work only when the functions are piecewise monotonic. Hence, partitioning data arrays into quasi-monotonic segments is an important problem [Yan *et al.*, 2004b; 2004a].

One could segment a data array in monotonic segments using a naïve algorithm: simply segment wherever there is an extremum. For example, it is easy to segment the array $\{0, 1, 2, 3, 2, 1, 0\}$ in two monotonic segments. However, monotonic segments must be significant for the application at hand and algorithms must be robust because the data will unavoidably be subject to noise or insignificant features. Hence, we might want to consider that the array $\{0, 1.2, 1.1, 3, 2, 1, 0\}$ has only two “significant” monotonic segments since the drop from 1.2 to 1.1 is not large enough to indicate a downward trend. Also algorithms must

be fast ($O(n)$) and not require excessive amounts of memory ($O(n)$).

In this paper, we address the “scale-based monotonicity” problem: monotonicity can be defined relative to a scale. We define a metric for monotonic approximation. We solve the following problem: given a number of segments K , find K segments having the smallest monotonic approximation error.

In short, the main novel results of this paper are:

1. A novel optimal segmentation algorithm;
2. A novel definition of scale-based flatness.

2 Monotonicity in Qualitative Modelling

We just list a few examples to show that the monotonic features are important to leverage in qualitative modeling. In QSIM [Kuipers, 1986], the qualitative value $QV(f, t)$, is the pair $\langle \text{qmag}, \text{qdir} \rangle$, where qmag is a landmark value l_i or an interval made up of landmark values (l_i, l_{i+1}) , and qdir takes a value from the sign domain $\{+, 0, -\}$ according to the $f'(t)$ is increasing, steady, or decreasing respectively. In Model-based Diagnosis (MBD), qualitative models are used to describe system behavior. Two kinds of qualitative models, based respectively on absolute or relative quantities, rely on the monotonicity of the function. The first one is *Finite Relation Qualitative Model* (FRQ) [Struss, 2002; Yan, 2003], where the qualitative relation is represented by tuples of real valued intervals. For a monotonic segment $[x_a, x_b]$ of f , the tuple is determined by the bounding rectangle $\square = [x_a, x_b] \times [\min(f(x_a), f(x_b)), \max(f(x_a), f(x_b))]$, that is the smallest rectangle such that $y = f(x) \wedge x_a \leq x \leq x_b \Rightarrow (x, y) \in \square$. The second one is *Qualitative Deviation Model* [Console *et al.*, 2003] where the qualitative relation is represented by the sign of deviation $[\Delta y]$ from a reference point x_{ref} defined as $+, -,$ or 0 , according to whether f is increasing, decreasing or flat. For example, if f is monotonic increasing, we have $[\Delta y] = \text{sign}(f(x) - f(x_{ref})) = \text{sign}(x - x_{ref}) = [\Delta x]$.

In this paper, we focus on abstracting qualitative models from scattered data obtained from numerical simulation or sensors. This work is motivated by at least two kinds of applications. First, when applying model-based diagnosis to real world engineering systems, we need to build symbolic qualitative models by a numerical simulation of the engineering models. Second, we need to explain system behaviors from

sensor data. Qualitative model abstraction in this paper is defined as transforming numerical values into qualitative values and functions into qualitative constraints [Šuc and Bratko, 2001]. **Monotonicity Analysis** is defined as partitioning a finite series of real values $x_1, x_2, \dots, x_n$ over an interval D into “monotonic” segments.

Monotonicity analysis is a crucial step to abstract qualitative models from scattered data but it rises a problem. The difficulty is that when the scattered data contains noise, the monotonicity is not absolute in a neighborhood of any point. The monotonic segments need to be significant in the context of the problem. The small fluctuations caused by noise must be ignored. This requires that noise be removed by computationally efficient methods such that the number of remaining segments is dependent only on the characterization of the noise.

Linear splines is an obvious approach to solve this problem. We can use top-down, bottom-up and sliding window algorithms to approximate the data with a set of straight lines [Keogh *et al.*, 2001]. Then the same-sign slopes can be aggregated as monotonic segments. Various algorithms are derived from classic algorithms [Key *et al.*, 2000] [Hunter and McIntosh, 1999]. The downside of these methods is that there is no link between linear spline approximation error and the actual monotonicity of the data (consider $y = e^x$). Hence, using linear splines, it is difficult to specify either the desired number of monotonic segments or some “monotonicity error” threshold. In addition, linear fitting algorithms are relatively expensive.

Inductive learning is used in [Šuc and Bratko, 2001] to automatically construct qualitative models from quantitative examples. The induced qualitative model is a binary tree, called a qualitative tree, which contains internal nodes (called splits) and qualitatively constrained functions at the leaves. A qualitative constrained function takes the form $M^{s_1, \dots, s_m} : \mathbb{R}^m \mapsto \mathbb{R}$, $s_i \in \{+, -\}$ and represents a function with m real-valued attributes strictly monotone increasing with respect to the i -th attribute if $s_i = +$, or strictly monotone decreasing if $s_i = -$. For example, $f = M^{+, -}(x, y)$ means f is increasing when x is increasing, and decreasing when y is increasing. A split is a partition of a variable. The unsupervised learning algorithm eq-QUIN determines the landmarks for the splits. The training data set is all possible pairs of data points. eq-QUIN checks the best split against all possible hypotheses. Its complexity is $O(n^2 2^m)$. QUIN is a more efficient algorithm that uses greedy search. Its complexity is $O(n^2 m^2)$. We do not address multidimensional data in this paper.

3 Monotonicity Error

In this section, we define a measure of monotonicity. Suppose we are given a set of n ordered samples noted $F : D = \{x_1, \dots, x_n\} \subset \mathbb{R} \rightarrow \mathbb{R}$ with real values $F(x_1), \dots, F(x_n)$ and $x_1 < x_2 < \dots < x_n$. We define, $F|_{[a,b]}$ as the restriction of F over $D \cap [a, b]$. We seek the best monotonic (increasing or decreasing) function $f : \mathbb{R} \rightarrow \mathbb{R}$ approximating F . Let $\Omega_\uparrow$ (resp. $\Omega_\downarrow$) be the set of all monotonic increasing (resp. decreasing) functions. The **Optimal Monotonic Approximation Function Error (OMAFE)** of F is given by

$\min_{f \in \Omega} \max_{x \in D} |f - F|$ where Ω is either $\Omega_\uparrow$ or $\Omega_\downarrow$. Sign $+$ or $-$ is associated to $\Omega_\uparrow$ or $\Omega_\downarrow$.

The segmentation of a set D is a sequence $S = X_1, \dots, X_m$ of closed intervals (called “segments”) in D with $[\min D, \max D] = \bigcup_i X_i$ such that $\max X_i = \min X_{i+1}$ and $X_i \cap X_j = \emptyset$ for $j \neq i, i+1, i-1$. Alternatively, we can define a segmentation from the set of points $X_i \cap X_{i+1} = \{y_i\}$. Given $F : \{x_1, \dots, x_n\} \rightarrow \mathbb{R}$ and a segmentation $\{X_i\}$, the Optimal Piecewise Monotonic Approximation Function Error (OPMAFE) of the segmentation is given by $\max_i \text{OMAFE}(F|_{X_i})$ where the directions of the segments X_i are alternating and such that the direction of the first segment is chosen so as to minimize the OPMAFE.

Solving for a best monotonic function can be done as follows. If we seek the best monotonic increasing function, we first define $\bar{f}_\uparrow(x) = \max\{F(y) : y \leq x\}$ (the maximum of all previous values) and $\underline{f}_\uparrow(x) = \min\{F(y) : y \geq x\}$ (the minimum of all values to come). If we seek the best monotonic decreasing function, we define $\bar{f}_\downarrow(x) = \max\{F(y) : y \geq x\}$ (the maximum of all values to come) and $\underline{f}_\downarrow(x) = \min\{F(y) : y \leq x\}$ (the minimum of all previous values). These functions which can be computed in linear time are all we need to solve for the best approximation function as shown by the next theorem which is a well-known result [Brooks, 1994; Ubhaya, 1974].

Theorem 1. Given $F : D = \{x_1, \dots, x_n\} \rightarrow \mathbb{R}$, a best monotonic increasing approximation function to F is given by $f_\uparrow = \frac{\bar{f}_\uparrow + \underline{f}_\uparrow}{2}$ and a best monotonic decreasing approximation function is given by $f_\downarrow = \frac{\bar{f}_\downarrow + \underline{f}_\downarrow}{2}$. The corresponding error (OMAFE) is given by $\max_{x \in D} \frac{|\bar{f}_\uparrow(x) - \underline{f}_\uparrow(x)|}{2}$ (monotonic increasing) or $\max_{x \in D} \frac{|\bar{f}_\downarrow(x) - \underline{f}_\downarrow(x)|}{2}$ (monotonic decreasing).

The implementation of the algorithm suggested by the theorem is straight-forward. Given a segmentation, we can compute the OPMAFE in $O(n)$ time using at most two passes. The functions $f_\uparrow$ and $f_\downarrow$ are sometimes called the standard optimal monotone functions as the solution is not unique in general.

4 Scale-Based Monotonicity

We present the notion of scale-based monotonicity. The intuition is that the fluctuations within a certain scale can be ignored.

Given an ordered set of real values $D = \{x_1, x_2, \dots, x_n\}$, consider $F : D = \{x_1, x_2, \dots, x_n\} \mapsto \mathbb{R}$. Given some tolerance value $\delta > 0$, we could say that the data points are *not going down* or are *upward monotone*, if consecutive measures do not go down by more than δ , that is, are such that $F(x_i) - F(x_{i+1}) < \delta$. However, this definition is not very useful because measures can repeatedly go down and thus the end value can be substantially lower than the start value. A more useful definition of *upward monotonicity* would be to require that we cannot find two values x_i and x_j ($x_i < x_j$) such that $F(x_j)$ is lower than $F(x_i)$ by δ (i.e. $F(x_i) - F(x_j) < \delta$). This definition is more useful because in the worst case, the

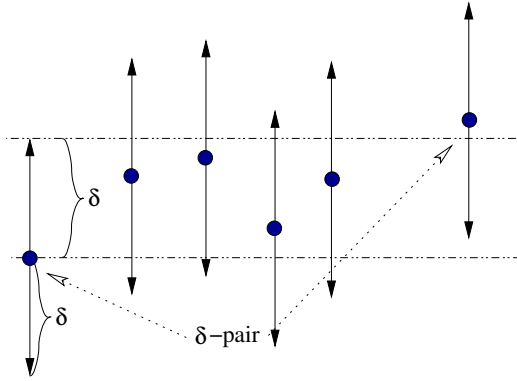


Figure 1: A δ -pair.

last measure will be only δ smaller than the first measure. However, we are still not guaranteed that the data does in fact increase at any point. Hence, we add the constraint that we can find at least two data points $x_k < x_l$ such that $F(x_l)$ is greater than $F(x_k)$ by at least δ ($F(x_l) - F(x_k) \geq \delta$).

To summarize, given some value $\delta > 0$, we say that a sequence of measures is *upward δ -monotone* if no two successive measures decrease by as much as δ , and at least one pair of successive measures increases by at least δ . Similarly, we say that a set of measures is *downward δ -monotone* if no two successive measures increase by as much as δ , and at least two measures decrease by at least δ .

This generalized definition of monotonicity was introduced in [Brooks, 1994] using δ -pairs (see Fig. 1):

Definition 1.

- $x < y \in D$ is a **δ -pair** (or a pair of scale δ) for F if $|F(y) - F(x)| \geq \delta$ and for all $z \in D$, $x < z < y$ implies $|F(z) - F(x)| < \delta$ and $|F(y) - F(z)| < \delta$.
- A δ -pair's direction is **increasing** or **decreasing** according to whether $F(y) > F(x)$ or $F(y) < F(x)$.

Notice that pairs of scale δ having opposite directions cannot overlap but they may share an endpoint. Pairs of scale δ of the same direction may overlap, but may not be nested for a certain δ .

We can define δ -monotonicity as follows:

Definition 2. Let X be an interval, F is **δ -monotone** on X if all δ -pairs in X have the same direction; F is **strictly δ -monotone** when there exists at least one such δ -pair. In this case:

- F is **δ -increasing** on X if X contains an increasing δ -pair.
- F is **δ -decreasing** on X if X contains a decreasing δ -pair.

We say that a pair is **significant** at scale δ if it is of scale δ' for $\delta' \geq \delta$.

In the next section, we discuss how to partition the data set into monotonic segments.

5 A Scale-Based Algorithm for Quasi-Monotonic Segmentation

Suppose that D is a finite set of reals having at least two elements. F is a real-valued function on D , i.e., $F : D \rightarrow \mathbb{R}$.

We begin by defining a segmentation at scale δ or a δ -segmentation.

Definition 3. Let $S = X_1, \dots, X_n$ be a segmentation of D , and let $\delta > 0$, then S is a **δ -segmentation** of F when all the following conditions hold.

- Each X_i is δ -monotone.
- Each X_i for $i \neq 1, n$ is strictly δ -monotone.
- At least one X_i is strictly δ -monotone.
- Adjacent strictly δ -monotone segments have opposite directions.
- For each strictly δ -monotone X_i , and for all $x \in X_i$, $F(x)$ lies in the closed interval bounded by $F(\min X_i)$ and $F(\max X_i)$.
- When X_1 is not strictly δ -monotone, then for all $x \in X_1 - \max X_1$, $F(x)$ lies in the open interval bounded by $F(\min X_2)$ and $F(\max X_2)$; and when X_n is not strictly δ -monotone, then for all $x \in X_n - \min X_n$, $F(x)$ lies in the open interval bounded by $F(\min X_{n-1})$ and $F(\max X_{n-1})$.

Each X_i is called a δ -segment of S . When strictly δ -monotone, X_i is a **proper δ -segment**; when not strictly δ -monotone, X_1 or X_n is an **improper δ -segment**. For strictly δ -monotone X_i , $\min X_i$ and $\max X_i$ are δ -extrema.

As the following theorems show, all δ -segmentations are “equivalent” and the monotonic approximation error (OPMAFE) is known precisely.

Theorem 2. Let $\delta > 0$. Let S_1 and S_2 be δ -segmentations of F ; then $|S_1| = |S_2|$. Furthermore, the first δ -segments of S_1 and S_2 are both either proper or improper, and similarly for the last δ -segments.

Theorem 3. Let $\delta > 0$. Let S be any δ -segmentation of F ; then the monotonic approximation error (OPMAFE) $\leq \delta/2$.

Now we want to compute a δ -segmentation given F . There are two approaches depending on the application one has in mind. The first one is to choose δ and then solve for the segments [Brooks, 1994], when the magnitude of noise is known. When one doesn't know how to choose δ , the second approach is to set the maximal number of segments K , especially when one knows the shape of the function. We focus on this second approach. We begin by labelling the extrema with a corresponding scale $\Delta(x)$.

Definition 4. Let $x \in D$ be a local extremum of F . x 's **delta-scale** $\Delta(x)$ is the largest $\delta > 0$ such that there exists a δ -segmentation having x as a δ -extremum.

It is immediate from the definition that if $\delta > \Delta(x)$, then x can't be a δ -extremum, but also that if x is a δ -extremum, then $\Delta(x) \geq \delta$. This is stated in the following proposition.

Proposition 1. Let $\delta > 0$, and let $S = X_1, \dots, X_n$ be a δ -segmentation of F . Then $\Delta(x) \geq \delta$ for every δ -extremum, x , of S .

We observe that there must be a smallest δ such that the cardinality of $X_\delta = \{x | \Delta(x) \geq \delta\}$ is at most K . However the set X_δ might contain repeated maxima or repeated minima and those might be removed before the set X_δ can define a δ -segmentation. This is stated in the next theorem.

Theorem 4. *Let $\delta > 0$ and consider the set of extrema $X_\delta = \{x | \Delta(x) \geq \delta\}$ of F as an ordered set. Each extremum in X_δ is either a maximum or a minimum, and a sequence of two minima or two maxima is possible. Consider a subset $X'_\delta \subset X_\delta$ such that*

- *it has no repeated maxima or minima;*
- *there is no superset X''_δ of X'_δ in X_δ without repeated extrema;*

then there exists a δ -segmentation S of F such that X'_δ is exactly the set of δ -extrema of S .

In order to compute $\Delta(x)$ for all extrema x , it is useful to introduce the following definitions.

Definition 5. *Opposite-sense extrema $x < z \in X \subset D$ are an **extremal pair** for X if for all $y \in X$, $x < y < z$ implies $F(y)$ lies in the closed interval defined by $F(x)$ and $F(z)$. The extremal pair has an extent $[F(x), F(z)]$ or $[F(z), F(x)]$ and increasing or decreasing direction according to $F(x) < F(z)$ or $F(x) > F(z)$. An extremal pair is **maximal** for X when no other extremal pair in X has larger extent. Similarly, an extremal pair is a maximal increasing (decreasing) when no increasing (decreasing) extremal pair has larger extent.*

Intuitively, a maximal extremal pair forms the largest decreasing (increasing) segment inside a larger increasing (decreasing) segment X .

Definition 6. *We recursively define **ordinary** and **special** intervals: D is an ordinary interval. When I is an ordinary interval, then an interval $J \subset I$ is special when J 's endpoints constitute a maximal extremal pair in I ; in this case J has direction inherited directly from its endpoints. Recursively, if J is special and interval $J' \subsetneq J$ has endpoints constituting a maximal extremal pair in J of direction opposite to that of J , then J' is special. Let $J_i, i = 1 \dots n$ be all special intervals in an ordinary or special interval X . Choose any nonempty subset of the collection $\{J_i\}$, and let S be the segmentation of X defined by the endpoints of the special intervals in the subset. Then any segment $I \in S$ that does not contain any of the J_i is an ordinary interval.*

One can see that the special intervals are nested. The endpoints x, z of the special intervals have $\Delta(x) = \Delta(z) = |F(x) - F(z)|$. We can call them "twins" due to the same $\Delta(x)$ value. The ordinary intervals are not nested. The endpoints of the ordinary intervals have different $\Delta(x)$. We call each of them "singleton".

If there are several extrema having an equal value, the way to choose the endpoints to constitute a maximal extremal pair is not unique. Therefore, there are different sets of special intervals. They are equivalent. For simplicity, we do not consider the case of equal valued extrema in Algorithm 1 and Algorithm 2. Instead, we explain how to deal with it verbally after introducing the algorithms.

Algorithm 1 computes $\Delta(x)$ for every extremum $x \in D$. The input to Algorithm 1 is a list of extrema - i.e. the extrema data points in the data array.

Algorithm 1 This algorithm labels the extrema in linear time (as in Definition 4).

INPUT: data - a list of the successive extremal values of F , alternating between maximum and minimum. Each element is an "extremum record", having three fields: value, sense, and index. Index identifies the data point, value gives F 's value at that data point, and sense indicates whether the extremum is a maximum or minimum.

OUTPUT: a list of "scale records". Each scale record has two fields: scale and extrema_list. Extrema_list comprises either one or two extremum records. The semantics of a scale record is that the indicated extrema have the indicated scale as delta-scale.

Notes about the algorithm: 1) Lists are accessed by element numbers; e.g. data(2) is the second element of data. 2) Lists are manipulated with functions Push and Pop. Push(thing, list) adds thing to list, resulting in thing being the first element of list. Pop(list) removes the first element of list, and returns this first element as the value of the function call. 3) **MakeList**(item^{1..*}) pushes the items into a list generated, starting from the first item. 4) **MakeScaleRecord** creates a scale record.

```

LET extrema = empty_list
LET scales = empty_list
Push( Pop(data), extrema )
Push( Pop(data), extrema )
for next_extremum IN data do
  while Length(extrema) > 1 AND {sense.next_extremum =
    "maximum" AND value.next_extremum > value.extrema(2)}
    OR {sense.next_extremum = "minimum" AND
    value.next_extremum < value.extrema(2)} do
    Push(MakeScaleRecord( | value.extrema(1) -
      value.extrema(2) |, MakeList( extrema(2), extrema(1) )),
      scales)
    Pop(extrema)
    Pop(extrema)
  if Length( extrema ) = 0 then
    Push( Pop( extrema_list.scales(1) ), extrema )
    Push( next_extremum, extrema )
  while Length(extrema) > 1 do
    Push( MakeScaleRecord( | value.extrema(1) -
      value.extrema(2) |, MakeList( extrema(1) )), scales)
    Pop(extrema)
  Push( extrema(1), extrema_list.scales(1) )
RETURN scales

```

The principle of Algorithm 1 is as follows: the *while* loop inside the *for* loop detects the special interval and labels the both endpoints with the difference of their values. But if the extrema list is empty after this labelling, which means that the top item in the list needs to be checked against the coming data, this top item is popped back to extrema in the *if* following the above *while*. Then the next extremum is pushed into the extrema list for the next *for* loops. The *while* outside the *for* loop determines the δ for the ordinary intervals. The last *push* states that the last extremum in the extrema list has the same scale as the one just popped into scale. Actually the endpoints of the biggest ordinary interval are always the last two pushed into scale record list and their scales are identical.

If one desires no more than K segments, it is a simple matter of picking δ as small as possible so that you have no more than $K + 1$ significant extrema ($\Delta(x) \geq \delta$) together with the first and last extrema (indexes 0 and $n - 1$). Algorithm 2 shows how once the labelling is complete, one can compute the segmentation in time $O(nK \log K)$ which we consider to be linear time when K is small compared to n . It also uses a fixed amount of memory ($O(K)$). The principle of Algorithm 2 is to select the $K + 1$ data points to be the endpoints of K segments. Since the endpoints of D are by default, the remaining endpoints come from the largest δ -extrema. The *for* is to select the largest δ -extrema. As in Algorithm 1, we know that the labels are either "twins" or "singleton". Thus in the *for* loop, a maximum of $K + 2$ data points are chosen. Then, after the first *if* outside *for*, the smallest δ -extrema are removed to reduce the total endpoints to at most $K + 1$. The rest of the code checks whether the first and the last endpoints of D are included. If not, the smallest δ -extrema will be removed in favor of the first and the last endpoints. The number of segments can be less than K since several extrema can take the same δ value and can be removed together.

Algorithm 2 Given a labelling algorithm (see Algorithm 1), this algorithm returns an optimal segmentation using at most K segments. It is assumed that there are at least $K + 1$ extrema to begin with.

INPUT: an array d containing the values indexed from 0 to $n - 1$
INPUT: K a bound on the number of segments desired
OUTPUT: segmentation points as per Theorem 4 and Proposition 1
 $L \leftarrow$ empty array (capacity $K + 3$)
for e is index of an extremum in d having scale δ , e are visited in increasing order **do**
 insert (e, δ) in L so that L is sorted by scale in decreasing order (sort on δ) using binary search
 if $\text{length}(L) = K + 3$ **then**
 pop last(L)
if $\text{length}(L) > K + 1$ **then**
 remove all elements of L having the scale of last(L).
if indexes $\{0, n - 1\} \not\subset L$ **then**
 if (index $0 \in L$ OR index $n - 1 \in L$) AND $\text{length}(L) = K + 1$ **then**
 remove all elements of L having the scale of last(L).
 if (index $0 \notin L$ AND index $n - 1 \notin L$) AND $\text{length}(L) \geq K$ **then**
 remove all elements of L having the scale of last(L).
RETURN: the indexes in L adding 0 and/or $n - 1$ when not already present

Algorithms 1 and 2 assume that no two extrema can have the same value. In case of equal valued extrema, we can modify these algorithms without increasing their complexity. In Algorithm 1, we can generalize ScaleRecord so that entries in the extrema_list field can also be lists of extrema of the same value. Then in the *while* loop, we can put the extrema of the same value into the corresponding list. For Algorithm 2, we just need to remember that whenever removing a middle point between two equal valued extrema, the two equal valued extrema have to be treated as one extrema to avoid having two adjacent minima or maxima.

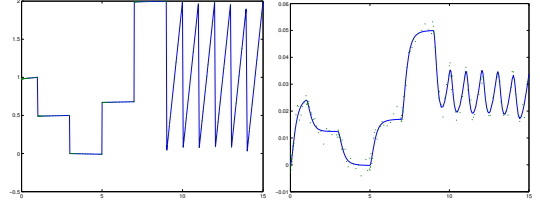


Figure 2: Input flow of tank A (left) and level of tank B (right) with added white noise.

6 Scale-Based Flatness

For applications, it is important to be able to find “flat” segments robustly in a data set. We propose the following definition:

Definition 7. Given $\delta > 0$, consider a δ -monotonic segment in a δ -segmentation, an interval I in this segment is δ -flat if the standard optimal monotonic approximating function (as defined in Theorem 1) is constant over I .

Because we can compute the standard optimal monotonic approximating function in $O(n)$ time, we can find flat segments in $O(n)$ time.

The next proposition addresses the flat segments in different δ -segmentations. The first point is derived from the fact that when the δ increases, the new segments are always the result of mergers of the previous segments. Hence, the segments at a small δ are always subsets of the segments at a larger δ . The second point says that the old flat segments will still be flat in the merged segment.

Proposition 2. Let S be a δ -segmentation and let S' be a δ' -segmentation for $\delta' > \delta$, then 1) any δ -segment $X \in S$ is a subset of some δ' -segment $X' \in S'$, 2) for any δ -flat interval I in X , I is also δ' -flat in X' .

7 Experimental Results

7.1 Cascade Tank Sample Data

As a source of synthetic data, we consider a system which consists of two cascade tanks A and B. Each tank has an input pipe (incoming water) and an output pipe (outgoing water). Tank A’s output pipe is the input pipe of tank B. In the equations below, let A and B be the level of water for the two tanks respectively. The change of water level is proportional to the difference of the input flow and the output flow. Assume in is the input flow of tank A. $f(A)$ is the out flow of tank A. $g(B)$ is the output flow of tank B [Kuipers, 1994]. Thus, we have

$$\begin{aligned} A' &= in - f(A) \\ B' &= f(A) - g(B) \end{aligned}$$

In principle, $f(A)$ and $g(B)$ are increasing functions. We assume $f(A) = k_1 A$ and $g(B) = k_2 B$. If we variate the input flow of tank A, in , we can control the level of tank B. In this way, we generated some sample data and added noise to it. See Fig. 2 for the the input flow, in , of tank A at the left and the level of tank B at the right.

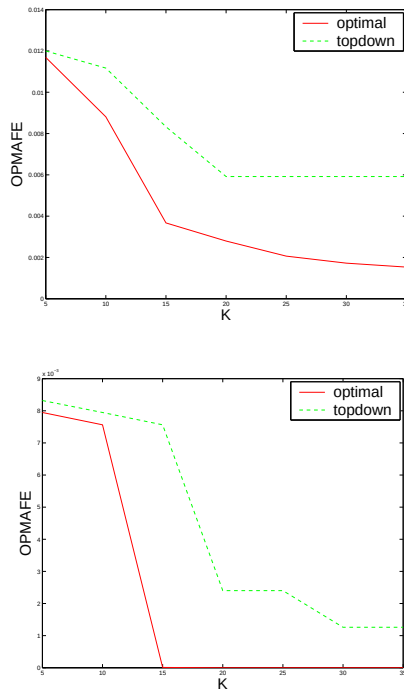


Figure 3: OPMAFE versus number of segments K for tank B with (above) and without white noise (below).

7.2 Optimal Algorithm vs. Top-down Algorithm

We choose the top-down linear spline approximation algorithm to compare the performance with the optimal algorithm developed in this paper. The top-down algorithm is a refined process choosing best split points that minimizes the linear regression error. The top-down algorithm we implemented has complexity of $O(nK^2)$ which is the best complexity on record. The optimal algorithm in this paper is $O(nK \log K)$. The optimal algorithm is faster for K sufficiently large. We also compare the monotonicity approximation error using OPMAFE. Figure 3 compares the OPMAFE for segmentations computed using either algorithm. The optimal algorithm we presented in this paper performs significantly better than the top-down algorithm. Note that OPMAFE is an absolute error measurement (which grows with the amplitude of the data).

8 Conclusion

Monotonicity is the most important feature to leverage when building the qualitative model from the scattered numerical data. This paper gives a solid mathematical foundation to the problem of defining monotonicity based on scale theory, where the values of the data points, not the distance between the data points, determine the monotonicity. It proves to be suitable in qualitative modeling. We present efficient algorithms to segment the data set into monotonic segments in linear time. We will continue to work on the multidimensional case.

References

- [Brooks, 1994] Martin Brooks. Approximation complexity for piecewise monotone functions and real data. *Computers and Mathematics with Applications*, 27(8), 1994.
- [Console et al., 2003] L. Console, G. Correndo, and C. Picardi. Deriving qualitative deviations from matlab models. In *Proc. of 14th Int. Workshop on Principles of Diagnosis*, pages 87–92, 2003.
- [Forbus, 1984] K. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.
- [Hunter and McIntosh, 1999] Jim Hunter and Neil McIntosh. Knowledge-based event detection in complex time series data. In *Proc. of Artificial Intelligence in Medicine and Medical Decision Making (AIMDM99)*, LNAI 1620, pages 271–280, 1999.
- [Keogh et al., 2001] Eamonn J. Keogh, Selina Chu, David Hart, and Michael J. Pazzani. An online algorithm for segmenting time series. In *ICDM*, pages 289–296, 2001.
- [Key et al., 2000] Herbert Key, Bernhard Rinner, and Benjamin Kuipers. Semi-quantitative system identification. *Artificial Intelligence*, 119, 2000.
- [Kuipers, 1986] B.J. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.
- [Kuipers, 1994] Benjamin Kuipers. *Qualitative Reasoning*. the MIT press, 1994.
- [Struss, 2002] P. Struss. Automated abstraction of numerical simulation models - theory and practical experience. In *Proc. of 16th Int. Workshop on Qualitative Reasoning*, pages 161–168, 2002.
- [Ubhaya, 1974] V. A. Ubhaya. Isotone optimization I. *Journal of Approximation Theory*, 12:146–159, 1974.
- [Šuc and Bratko, 2001] D. Šuc and I. Bratko. Induction of qualitative tree. In *Proc. of European Conference of Machine Learning in 2001 (LNCS 2167)*, pages 442–453. Springer, 2001.
- [Šuc, 2003] D. Šuc. *Machine Reconstruction of Human Control Strategies*, volume 99 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands, 2003.
- [Williams, 1992] B. Williams. Interaction-based invention: designing devices from first principles. In *Recent Advances in Qualitative Physics*, pages 413–433. MIT Press, Cambridge, MA, 1992.
- [Yan et al., 2004a] Yuhong Yan, Daniel Lemire, and Martin Brooks. Monotone pieces analysis for qualitative modeling. In *Proceedings of ECAI MONET’04*, 2004.
- [Yan et al., 2004b] Yuhong Yan, Daniel Lemire, and Martin Brooks. Monotonicity analysis for constructing qualitative models. In *Proceedings of MBR’04*, 2004.
- [Yan, 2003] Y. Yan. Qualitative model abstraction for diagnosis. In *Proc. of 17th Int. Workshop on Qualitative Reasoning*, pages 171–179, 2003.

Declarative and Computational Properties of Logic Programs with Aggregates*

Francesco Calimeri and Wolfgang Faber[†] and Nicola Leone and Simona Perri

Department of Mathematics, University of Calabria, I-87030 Rende (CS), Italy

Email: {calimeri,faber,leone,perri}@mat.unical.it

Abstract

We investigate the properties of logic programs with aggregates. We mainly focus on programs with monotone and antimonotone aggregates ($LP_{m,a}^A$ programs). We define a new notion of unfounded set for $LP_{m,a}^A$ programs, and prove that it is a sound generalization of the standard notion of unfounded set for aggregate-free programs. We show that the answer sets of an $LP_{m,a}^A$ program are precisely its unfounded-free models.

We define a well-founded operator $\mathcal{W}_{\mathcal{P}}$ for $LP_{m,a}^A$ programs; we prove that its total fixpoints are precisely the answer sets of $\mathcal{P}$, and its least fixpoint $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is contained in the intersection of all answer sets (if $\mathcal{P}$ admits an answer set). $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is efficiently computable, and for aggregate-free programs it coincides with the well-founded model.

We carry out an in-depth complexity analysis in the general framework, including also nonmonotone aggregates. We prove that monotone and antimonotone aggregates do not increase the complexity of cautious reasoning, which remains in co-NP. Nonmonotone aggregates, instead, do increase the complexity by one level in the polynomial hierarchy. Our results allow also to generalize and speed-up ASP systems with aggregates.

1 Introduction

The introduction of aggregates atoms [Kemp and Stuckey, 1991; Denecker *et al.*, 2001; Gelfond, 2002; Simons *et al.*, 2002; Dell’Armi *et al.*, 2003; Pelov and Truszczyński, 2004; Pelov *et al.*, 2004] is one of the major linguistic extensions to Answer Set Programming of the recent years.

While both semantic and computational properties of standard (aggregate-free) logic programs have been deeply investigated, only few works have focused on logic programs with aggregates; their behaviour, their semantic properties, and their computational features are still far from being fully

clarified. A recent proposal for answer set semantics is receiving a consensus [Faber *et al.*, 2004]. However, unfounded sets and the well-founded operator [Van Gelder *et al.*, 1991], which are important for both the characterization and the computation of standard LPs [Leone *et al.*, 1997; Simons *et al.*, 2002; Calimeri *et al.*, 2002; Koch *et al.*, 2003; Pfeifer, 2004], have not been defined in a satisfactory manner for logic programs with aggregates. Moreover, the impact of aggregates on the computational complexity of the reasoning tasks has not been analyzed in depth.

This paper makes a first step to overcome this deficiency, improving the characterization of programs with aggregates, for both declarative and computational purposes. The main contributions of the paper are as follows.

- We define the notion of unfounded set for logic programs with both monotone and antimonotone aggregates ($LP_{m,a}^A$ programs). This notion is a sound generalization of the concept of unfounded sets previously given for programs without aggregates. We show that our definition coincides with the original definition of unfounded sets of [Van Gelder *et al.*, 1991] on the class of normal (aggregate-free) programs, and shares its nice properties (like, e.g., the existence of the greatest unfounded set).
- We provide a declarative characterization of answer sets in terms of unfounded sets. In particular, answer sets are precisely unfounded-free models of an $LP_{m,a}^A$ program.
- We define a well-founded operator $\mathcal{W}_{\mathcal{P}}$ for logic programs with aggregates, which extends the classical well-founded operator [Van Gelder *et al.*, 1991]. The total fixpoints of $\mathcal{W}_{\mathcal{P}}$ are exactly the answer sets of $\mathcal{P}$, and its least fixpoint $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is contained in the intersection of all answer sets. Importantly, $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is polynomial-time computable.
- We analyze the complexity of logic programs with arbitrary (also nonmonotone) aggregates and fragments thereof. Both monotone and antimonotone aggregates do not affect the complexity of answer set semantics, which remains co-NP-complete (for cautious reasoning). Nonmonotone aggregates, instead, do increase the complexity, jumping to the second level of the polynomial hierarchy (Π_2^P). For space limitations, some proofs are sketched.

2 Logic Programs with Aggregates

In this section, we recall syntax, semantics, and some basic properties of logic programs with aggregates.

*This work was partially supported by the EC under projects IST-2002-33570 INFOMIX and IST-2001-37004 WASP, and by FWF under projects P16536-N04 and P17212-N04.

[†]Funded by APART of the Austrian Academy of Sciences.

2.1 Syntax

We assume that the reader is familiar with standard LP; we refer to atoms, literals, rules, and programs of LP, as *standard atoms*, *standard literals*, *standard rules*, and *standard programs*, respectively. Two literals are said to be complementary if they are of the form p and $\neg p$ for some atom p . Given a literal L , $\neg.L$ denotes its complementary literal. Accordingly, given a set A of literals, $\neg.A$ denotes the set $\{\neg.L \mid L \in A\}$. For further background, see [Baral, 2002; Gelfond and Lifschitz, 1991].

Set Terms. A (LP^A) *set term* is either a symbolic set or a ground set. A *symbolic set* is a pair $\{Vars : Conj\}$, where $Vars$ is a list of variables and $Conj$ is a conjunction of standard atoms.¹ A *ground set* is a set of pairs of the form $\{\bar{t} : Conj\}$, where $\bar{t}$ is a list of constants and $Conj$ is a ground (variable free) conjunction of standard atoms.

Aggregate Functions. An *aggregate function* is of the form $f(S)$, where S is a set term, and f is an *aggregate function symbol*. Intuitively, an aggregate function can be thought of as a (possibly partial) function mapping multisets of constants to a constant.

Example 1 (In the examples, we adopt the syntax of DLV to denote aggregates.) Aggregate functions currently supported by the DLV system are: $\#count$ (number of terms), $\#sum$ (sum of non-negative integers), $\#times$ (product of positive integers), $\#min$ (minimum term, undefined for empty set), $\#max$ (maximum term, undefined for empty set)².

Aggregate Literals. An *aggregate atom* is $f(S) \prec T$, where $f(S)$ is an aggregate function, $\prec \in \{=, <, \leq, >, \geq\}$ is a predefined comparison operator, and T is a term (variable or constant) referred to as *guard*.

Example 2 The following aggregate atoms in DLV notation, where the latter contains a ground set and could be a ground instance of the former:

$$\begin{aligned} \#max\{Z : r(Z), a(Z, V)\} &> Y \\ \#max\{\langle 2 : r(2), a(2, k) \rangle, \langle 2 : r(2), a(2, c) \rangle\} &> 1 \end{aligned}$$

An *atom* is either a standard (LP) atom or an aggregate atom. A *literal* L is an atom A or an atom A preceded by the default negation symbol $\neg$; if A is an aggregate atom, L is an *aggregate literal*.

LP^A Programs. A (LP^A) *rule* r is a construct

$$a :- b_1, \dots, b_k, \text{ not } b_{k+1}, \dots, \text{ not } b_m.$$

where a is a standard atom, $b_1, \dots, b_m$ are atoms, and $m \geq k \geq 0$. The atom a is referred to as the *head* of r while the conjunction $b_1, \dots, b_k, \text{ not } b_{k+1}, \dots, \text{ not } b_m$ is the *body* of r . We denote the head atom by $H(r)$, and the set $\{b_1, \dots, b_k, \text{ not } b_{k+1}, \dots, \text{ not } b_m\}$ of the body literals by $B(r)$.

¹Intuitively, a symbolic set $\{X : a(X, Y), p(Y)\}$ stands for the set of X -values making $a(X, Y), p(Y)$ true, i.e., $\{X \mid \exists Y \text{ s.t. } a(X, Y), p(Y) \text{ is true}\}$.

²The first two aggregates correspond, respectively, to the cardinality and weight constraint literals of Smodels.

A (LP^A) *program* is a set of LP^A rules. A *global* variable of a rule r is a variable appearing in a standard atom of r ; all other variables are *local* variables.

Safety. A rule r is *safe* if the following conditions hold: (i) each global variable of r appears in a positive standard literal in the body of r ; (ii) each local variable of r appearing in a symbolic set $\{Vars : Conj\}$ appears in an atom of $Conj$; (iii) each guard of an aggregate atom of r is a constant or a global variable. A program $\mathcal{P}$ is *safe* if all $r \in \mathcal{P}$ are safe. In the following we assume that LP^A programs are safe.

2.2 Answer Set Semantics

Universe and Base. Given a LP^A program $\mathcal{P}$, let $U_{\mathcal{P}}$ denote the set of constants appearing in $\mathcal{P}$, and $B_{\mathcal{P}}$ be the set of standard atoms constructible from the (standard) predicates of $\mathcal{P}$ with constants in $U_{\mathcal{P}}$. Given a set X , let $\bar{2}^X$ denote the set of all multisets over elements from X . Without loss of generality, we assume that aggregate functions map to $\mathbb{I}$ (the set of integers).

Example 3 $\#count$ is defined over $\bar{2}^{U_{\mathcal{P}}}$, $\#sum$ over $\bar{2}^{\mathbb{N}}$, $\#times$ over $\bar{2}^{\mathbb{N}^+}$, $\#min$ and $\#max$ are defined over $\bar{2}^{\mathbb{N}} - \{\emptyset\}$.

Instantiation. A *substitution* is a mapping from a set of variables to $U_{\mathcal{P}}$. A substitution from the set of global variables of a rule r (to $U_{\mathcal{P}}$) is a *global substitution for r* ; a substitution from the set of local variables of a symbolic set S (to $U_{\mathcal{P}}$) is a *local substitution for S* . Given a symbolic set without global variables $S = \{Vars : Conj\}$, the *instantiation of S* is the following ground set of pairs $inst(S)$:

$$\{\langle \gamma(Vars) : \gamma(Conj) \rangle \mid \gamma \text{ is a local substitution for } S\}.$$

A *ground instance* of a rule r is obtained in two steps: (1) a global substitution σ for r is first applied over r ; (2) every symbolic set S in $\sigma(r)$ is replaced by its instantiation $inst(S)$. The instantiation $Ground(\mathcal{P})$ of a program $\mathcal{P}$ is the set of all possible instances of the rules of $\mathcal{P}$.

Example 4 Consider the following program $\mathcal{P}_1$:

$$\begin{aligned} q(1) \vee p(2, 2). & & q(2) \vee p(2, 1). \\ t(X) :- q(X), \#sum\{Y : p(X, Y)\} &> 1. \end{aligned}$$

The instantiation $Ground(\mathcal{P}_1)$ is the following:

$$\begin{aligned} q(1) \vee p(2, 2).t(1) :- q(1), \#sum\{\langle 1 : p(1, 1) \rangle, \langle 2 : p(1, 2) \rangle\} &> 1. \\ q(2) \vee p(2, 1).t(2) :- q(2), \#sum\{\langle 1 : p(2, 1) \rangle, \langle 2 : p(2, 2) \rangle\} &> 1. \end{aligned}$$

Interpretations. An *interpretation* for a LP^A program $\mathcal{P}$ is a consistent set of standard ground literals, that is $I \subseteq (B_{\mathcal{P}} \cup \neg.B_{\mathcal{P}})$ such that $I \cap \neg.I = \emptyset$. A standard ground literal L is true (resp. false) w.r.t I if $L \in I$ (resp. $L \in \neg.I$). If a standard ground literal is neither true or false w.r.t I then it is undefined w.r.t I . We denote by I^+ (resp. I^-) the set of all standard positive (resp. negative) literals occurring in I . We also denote with $\bar{I}$ the set of undefined standard literals w.r.t I . An interpretation I is *total* if $\bar{I}$ is empty (i.e., $I^+ \cup \neg.I^- = B_{\mathcal{P}}$), otherwise I is *partial*.

³Given a substitution σ and a LP^A object Obj (rule, set, etc.), we denote by $\sigma(Obj)$ the object obtained by replacing each variable X in Obj by $\sigma(X)$.

An interpretation also provides a meaning for aggregate literals. Their truth value is first defined for total interpretations, and then induced for partial ones.

Let I be a total interpretation. A standard ground conjunction is true (resp. false) w.r.t. I if all its literals are true. The meaning of a set, an aggregate function, and an aggregate atom under an interpretation, is a multiset, a value, and a truth-value, respectively. Let $f(S)$ be an aggregate function. The valuation $I(S)$ of S w.r.t. I is the multiset of the first constant of the elements in S whose conjunction is true w.r.t. I . More precisely, let $I(S)$ denote the multiset $[t_1 \mid \langle t_1, \dots, t_n : Conj \rangle \in S \wedge Conj \text{ is true w.r.t. } I]$. The valuation $I(f(S))$ of an aggregate function $f(S)$ w.r.t. I is the result of the application of f on $I(S)$. If the multiset $I(S)$ is not in the domain of f , $I(f(S)) = \perp$ (where $\perp$ is a fixed symbol not occurring in $\mathcal{P}$).⁴

An instantiated aggregate atom $A = f(S) \prec k$ is true w.r.t. I if: (i) $I(f(S)) \neq \perp$, and, (ii) $I(f(S)) \prec k$ holds; otherwise, A is false. An instantiated aggregate literal not $A = \text{not } f(S) \prec k$ is true w.r.t. I if (i) $I(f(S)) \neq \perp$, and, (ii) $I(f(S)) \prec k$ does not hold; otherwise, A is false.

If I is a *partial* interpretation, an aggregate literal A is true (resp. false) w.r.t. I if it is true (resp. false) w.r.t. *each* total interpretation J extending I (i.e., $\forall J \text{ s.t. } I \subseteq J, A \text{ is true w.r.t. } J$); otherwise it is undefined.

Example 5 Consider the atom $A = \#sum\{\langle 1 : p(2, 1) \rangle, \langle 2 : p(2, 2) \rangle\} > 1$ from Example 4. Let S be the ground set in A . For the interpretation $I = \{p(2, 2)\}$, each extending total interpretation contains either $p(2, 1)$ or not $p(2, 1)$. Therefore, either $I(S) = [2]$ or $I(S) = [1, 2]$ and the application of $\#sum$ yields either $2 > 1$ or $3 > 1$, hence A is true w.r.t. I .

Minimal Models. Given an interpretation I , a rule r is *satisfied* w.r.t. I if the head atom is true w.r.t. I whenever all body literals are true w.r.t. I . A total interpretation M is a *model* of a LP^A program $\mathcal{P}$ if all $r \in \text{Ground}(\mathcal{P})$ are satisfied w.r.t. M . A model M for $\mathcal{P}$ is (subset) minimal if no model N for $\mathcal{P}$ exists such that $N^+ \subset M^+$. Note that, under these definitions, the word *interpretation* refers to a possibly partial interpretation, while a *model* is always a total interpretation.

Answer Sets. We now recall the generalization of the Gelfond-Lifschitz transformation for LP^A programs from [Faber *et al.*, 2004].

Definition 1 ([Faber *et al.*, 2004]) Given a ground LP^A program $\mathcal{P}$ and a total interpretation I , let $\mathcal{P}^I$ denote the transformed program obtained from $\mathcal{P}$ by deleting all rules in which a body literal is false w.r.t. I . I is an answer set of a program $\mathcal{P}$ if it is a minimal model of $\text{Ground}(\mathcal{P})^I$.

Example 6 Consider the following two programs:

$$\begin{aligned} P_1 : \{ & p(a) : -\#count\{X : p(X)\} > 0. \} \\ P_2 : \{ & p(a) : -\#count\{X : p(X)\} < 1. \} \end{aligned}$$

$\text{Ground}(P_1) = \{p(a) : -\#count\{a : p(a)\} > 0.\}$ and $\text{Ground}(P_2) = \{p(a) : -\#count\{a : p(a)\} < 1.\}$, and interpretation $I_1 = \{p(a)\}$, $I_2 = \emptyset$. Then, $\text{Ground}(P_1)^{I_1} =$

⁴In this paper, we assume that the value of an aggregate function can be computed in time polynomial in the size of the input multiset.

$\text{Ground}(P_1)$, $\text{Ground}(P_1)^{I_2} = \emptyset$, and $\text{Ground}(P_2)^{I_1} = \emptyset$, $\text{Ground}(P_2)^{I_2} = \text{Ground}(P_2)$ hold.

I_2 is the only answer set of P_1 (because I_1 is not a minimal model of $\text{Ground}(P_1)^{I_1}$), while P_2 admits no answer set (I_1 is not a minimal model of $\text{Ground}(P_2)^{I_1}$, and I_2 is not a model of $\text{Ground}(P_2) = \text{Ground}(P_2)^{I_2}$).

Note that any answer set A of $\mathcal{P}$ is also a model of $\mathcal{P}$ because $\text{Ground}(\mathcal{P})^A \subseteq \text{Ground}(\mathcal{P})$, and rules in $\text{Ground}(\mathcal{P}) - \text{Ground}(\mathcal{P})^A$ are satisfied w.r.t. A .

Monotonicity. Given two interpretations I and J we say that $I \leq J$ if $I^+ \subseteq J^+$ and $J^- \subseteq I^-$. A ground literal ℓ is *monotone*, if for all interpretations I, J , such that $I \leq J$, we have that: (i) ℓ true w.r.t. I implies ℓ true w.r.t. J , and (ii) ℓ false w.r.t. J implies ℓ false w.r.t. I . A ground literal ℓ is *antimonotone*, if the opposite happens, that is, for all interpretations I, J , such that $I \leq J$, we have that: (i) ℓ true w.r.t. J implies ℓ true w.r.t. I , and (ii) ℓ false w.r.t. I implies ℓ false w.r.t. J . A ground literal ℓ is *nonmonotone*, if it is neither monotone nor antimonotone.

Note that positive standard literals are monotone, whereas negative standard literals are antimonotone. Aggregate literals may be monotone, antimonotone or nonmonotone, regardless whether they are positive or negative. Nonmonotone literals include the sum over (possibly negative) integers and the average.

Example 7 All ground instances of $\#count\{Z : r(Z)\} > 1$ and $\text{not } \#count\{Z : r(Z)\} < 1$ are monotone, while for $\#count\{Z : r(Z)\} < 1$, and $\text{not } \#count\{Z : r(Z)\} > 1$ they are antimonotone.

We denote by LP_m^A (LP_a^A/LP_n^A) the fragment of LP^A where only monotone (antimonotone/nonmonotone) aggregates are allowed; $LP_{m,a}^A$ allows for both monotone and antimonotone aggregates.

Remark 1 Observe that our definitions of interpretation and truth values preserve “knowledge monotonicity”. If an interpretation J extends I (i.e., $I \subseteq J$), then each literal which is true w.r.t. I is true w.r.t. J , and each literal which is false w.r.t. I is false w.r.t. J as well.

3 Unfounded Sets

In this section, we extend the notion of unfounded sets, given in [Van Gelder *et al.*, 1991] for aggregate-free programs, to the framework of $LP_{a,m}^A$ programs. Let us denote by $S1 \dot{\cup} \neg.S2$ the set $(S1 - S2) \cup \neg.S2$, where $S1$ and $S2$ are sets of standard ground literals.

Definition 2 (Unfounded Set) A set X of ground atoms is an unfounded set for an $LP_{a,m}^A$ program $\mathcal{P}$ w.r.t. I if, for each rule $r \in \mathcal{P}$ having the head atom belonging to X , at least one of the following conditions holds: 1. some antimonotone body literal of r is false w.r.t. I , and 2. some monotone body literal of r is false w.r.t. $I \dot{\cup} \neg.X$.

Example 8 Consider the following program $\mathcal{P}$:

$$\begin{aligned} r_1 : & a(1) : -\#count\{\langle 1 : a(1) \rangle, \langle 2 : a(2) \rangle, \langle 3 : a(3) \rangle\} > 2. \\ r_2 : & a(2). \\ r_3 : & a(3) : -\#count\{\langle 1 : a(1) \rangle, \langle 2 : a(2) \rangle, \langle 3 : a(3) \rangle\} > 2. \end{aligned}$$

and $I = \{a(1), a(2), a(3)\}$. Then $X = \{a(1)\}$ is an unfounded set w.r.t. $I, \mathcal{P}$, since condition 2 holds for r_1 . Also $\{a(3)\}$, and $\{a(1), a(3)\}$ are unfounded.

We can show that Definition 2 generalizes the one given in [Van Gelder *et al.*, 1991] for aggregate-free programs.

Theorem 9 For an aggregate-free $\text{LP}_{a,m}^A$ program $\mathcal{P}$, Definition 2 is equivalent to the one of [Van Gelder *et al.*, 1991].

Thus, Definition 2 is an alternative characterization of unfounded sets for standards literals. In fact, while condition 1 of Definition 2 does not exactly cover the first one in [Van Gelder *et al.*, 1991], condition 2 catches all cases of the second in [Van Gelder *et al.*, 1991] and those "lost" by condition 1. This separates positive and negative literals, allowing to distinguish between the behavior of monotone and anti-monotone literals.

Theorem 10 If X_1 and X_2 are unfounded sets w.r.t. an interpretation I for a $\text{LP}_{a,m}^A$ program $\mathcal{P}$, then also $X_1 \cup X_2$ is an unfounded set w.r.t. I for $\mathcal{P}$.

Proof sketch. For Condition 2 of Def. 2, observe that $I \dot{\cup} \neg.(X_1 \cup X_2) \leq I \dot{\cup} \neg.X_1$. Therefore, if a monotone literal ℓ is false w.r.t. $I \dot{\cup} \neg.X_1$, then it is false w.r.t. $I \dot{\cup} \neg.(X_1 \cup X_2)$. Symmetrically for X_2 . $\square$

By virtue of Theorem 10, the union of all unfounded sets for $\mathcal{P}$ w.r.t. I is an unfounded set. We call it the *Greatest Unfounded Set* of $\mathcal{P}$ w.r.t. I , and denote it by $\text{GUS}_{\mathcal{P}}(I)$.

Proposition 1 Let I and J be interpretations for an $\text{LP}_{a,m}^A$ program $\mathcal{P}$. If $I \subseteq J$, then $\text{GUS}_{\mathcal{P}}(I) \subseteq \text{GUS}_{\mathcal{P}}(J)$.

Proof sketch. Follows from Remark 1, since $I \dot{\cup} \neg.X \subseteq J \dot{\cup} \neg.X$. $\square$

4 Answer Sets and Unfounded Sets

In this section, we provide a couple of characterizations of answer sets in terms of unfounded sets.

Definition 3 (Unfounded-free Interpretation) Let I be an interpretation for a program $\mathcal{P}$. I is unfounded-free if $I \cap X = \emptyset$ for each unfounded set X for $\mathcal{P}$ w.r.t. I .

The next lemma gives an equivalent characterization of the unfounded-free property for total interpretations.

Lemma 11 Let I be a total interpretation for a program $\mathcal{P}$. I is unfounded-free iff no nonempty set of atoms contained in I is an unfounded set for $\mathcal{P}$ w.r.t. I .

Proof sketch. If I is not unfounded-free, an unfounded set X for $\mathcal{P}$ w.r.t. I s.t. $X \cap I \neq \emptyset$ exists. Then, $X \cap I$ is also an unfounded set. $\square$

Theorem 12 A model M is an answer set of an $\text{LP}_{a,m}^A$ program $\mathcal{P}$ if and only if M is unfounded-free.

Proof sketch. If M is a model and $X \subseteq M$ is a non-empty unfounded set w.r.t. M , then it can be shown that $M \dot{\cup} \neg.X$ is a model of $\mathcal{P}^M$, hence M is no answer set. On the other hand, if M is an unfounded-free model but not an answer set, a model N of $\mathcal{P}^M$ s.t. $N^+ \subset M^+$ must exist. We can show that $M^+ - N^+$ is an unfounded set. $\square$

Now we give another interesting characterization of answer sets. A total interpretation is an answer set if and only if its false literals are unfounded.

Lemma 13 A total interpretation M is a model for $\mathcal{P}$ iff $\neg.M^-$ is an unfounded set for $\mathcal{P}$ w.r.t. M .

Proof sketch. The result follows from the fact that $M = M \dot{\cup} \neg.(\neg.M^-)$. $\square$

Theorem 14 Let M be a total interpretation for a program $\mathcal{P}$. M is an answer set iff $\neg.M^- = \text{GUS}_{\mathcal{P}}(M)$.

Proof sketch. Can be shown using Lemmata 13 and 11 and Theorem 12. $\square$

5 Well-Founded Semantics

In this section we extend the $\mathcal{W}_{\mathcal{P}}$ defined in [Van Gelder *et al.*, 1991] for aggregate-free programs to $\text{LP}_{a,m}^A$ programs. Then, we show that the answer sets of an $\text{LP}_{a,m}^A$ program $\mathcal{P}$ coincide exactly with the total fixpoints of $\mathcal{W}_{\mathcal{P}}$.

We start by providing an extension to $\text{LP}_{a,m}^A$ programs of the immediate consequence operator $T_{\mathcal{P}}$ defined in [Van Gelder *et al.*, 1991] for three-valued interpretations of standard logic programs.

Definition 4 Let $\mathcal{P}$ be a $\text{LP}_{a,m}^A$ program. Define the operators $T_{\mathcal{P}}$ and $\mathcal{W}_{\mathcal{P}}$ from $2^{B_{\mathcal{P}} \cup \neg.B_{\mathcal{P}}}$ to $2^{B_{\mathcal{P}}}$ and $2^{B_{\mathcal{P}} \cup \neg.B_{\mathcal{P}}}$, respectively, as follows.

$$\begin{aligned} T_{\mathcal{P}}(I) &= \{a \in B_{\mathcal{P}} \mid \exists r \in \text{Ground}(\mathcal{P}) \text{ s.t. } a = H(r) \\ &\quad \text{and } B(r) \text{ is true w.r.t. } I\} \\ \mathcal{W}_{\mathcal{P}}(I) &= T_{\mathcal{P}}(I) \cup \neg.\text{GUS}_{\mathcal{P}}(I). \end{aligned}$$

Theorem 15 Let M be a total interpretation for a program $\mathcal{P}$. M is an answer set for $\mathcal{P}$ iff M is a fixpoint of $\mathcal{W}_{\mathcal{P}}$.

Proof sketch. $M^- = \neg.\text{GUS}_{\mathcal{P}}(M)$ holds by virtue of Theorem 14, $M^+ = T_{\mathcal{P}}(M)$ can be shown using Lemma 11 and Definition 2. $\square$

The $\mathcal{W}_{\mathcal{P}}$ operator is clearly monotone on a meet semilattice, and it therefore admits a least fixpoint [Tarski, 1955]. This fixpoint can be computed iteratively starting from the empty set, and approximates the intersection of all answer sets (if any).

Theorem 16 Given an $\text{LP}_{m,a}^A$ program $\mathcal{P}$, let $\{W_n\}_{n \in \mathbb{N}}$ be the sequence whose n -th term is the n -fold application of the $\mathcal{W}_{\mathcal{P}}$ operator on the empty set (i.e., $W_0 = \emptyset$, $W_n = \mathcal{W}_{\mathcal{P}}(W_{n-1})$). Then (a) $\{W_n\}_{n \in \mathbb{N}}$ converges to a limit $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$, and (b) for each answer set M for $\mathcal{P}$, $M \supseteq \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$.

Proof sketch. (a) follows from the monotonicity of $\mathcal{W}_{\mathcal{P}}$ and the finiteness of $B_{\mathcal{P}}$. (b) holds since all atoms computed by $T_{\mathcal{P}}$ belong to any answer set and because of Theorem 14. $\square$

From Theorem 15 and 16, the following easily follows.

Corollary 17 If $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is a total interpretation, then it is the unique answer set of $\mathcal{P}$.

The following proposition confirms the intuition that Definition 4 extends the $\mathcal{W}_{\mathcal{P}}$ operator, as defined in [Van Gelder *et al.*, 1991] for standard programs, to $\text{LP}_{m,a}^A$ programs.

Proposition 2 Let $\mathcal{P}$ be an aggregate-free program. Then the $\mathcal{W}_{\mathcal{P}}$ operator of Definition 4 exactly coincides with $\mathcal{W}_{\mathcal{P}}$ operator defined in [Van Gelder *et al.*, 1991].

Corollary 18 On aggregate-free programs, $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ coincides with the well-founded model of [Van Gelder *et al.*, 1991].

Moreover, there are simple cases where $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ captures the intended meaning of the program.

Example 19 Consider the following program $\mathcal{P}$:

$a(1) :- \# \text{sum}\{\langle 1 : a(1) \rangle, \langle 2 : a(2) \rangle\} > 1.$
 $b :- \text{not } a(1). \quad a(2) :- b. \quad b :- \text{not } c.$

We have $\mathcal{W}_{\mathcal{P}}(\emptyset) = \{\text{not } c\}$, then $\mathcal{W}_{\mathcal{P}}(\{\text{not } c\}) = \{b, \text{not } c, \text{not } a(1)\}$, then $\mathcal{W}_{\mathcal{P}}(\{b, \text{not } c, \text{not } a(1)\}) = \{b, \text{not } c, \text{not } a(1), a(2)\} = \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$.

It is easy to verify that $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ (which here is total) is an answer set for $\mathcal{P}$.

6 Computational Complexity

We first show the tractability of the well-founded semantics for $\text{LP}_{m,a}^A$, and we then analyze the complexity of answer set semantics for general LP^A programs. We consider the propositional case, hence, throughout this section we assume that programs are ground.

Theorem 20 Given a ground $\text{LP}_{m,a}^A$ program $\mathcal{P}$: 1. The greatest unfounded set $\text{GUS}_{\mathcal{P}}(I)$ of $\mathcal{P}$ w.r.t. a given interpretation I is polynomial-time computable; 2. $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is polynomial-time computable.

Proof sketch. We define an operator Φ_I from $B_{\mathcal{P}}$ to $B_{\mathcal{P}}$ as follows: $\Phi_I(Y) = \{a \mid \exists r \in \mathcal{P} \text{ with } a = H(r) \text{ s.t. no antimonotone body literal of } r \text{ false w.r.t. } I \wedge \text{all monotone body literals of } r \text{ are true w.r.t. } Y - \neg I^-\}$. The sequence $\phi_0 = \emptyset, \phi_k = \Phi_I(\phi_{k-1})$ is monotonically increasing and converges finitely to a limit ϕ_{λ} , for which $\phi_{\lambda} = B_{\mathcal{P}} - \text{GUS}_{\mathcal{P}}(I)$ can be shown. Furthermore, each application of Φ_I is polynomial in our setting⁵, and also λ is polynomial in $|B_{\mathcal{P}}|$. From this, the result follows. $\square$

This result has a positive impact also for the computation of the answer set semantics of logic programs with aggregates. Indeed, as stated in Theorem 16, $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ approximates the intersection of all answer sets from the bottom, and can be therefore used to efficiently prune the search space.

We next analyze the complexity of the answer set semantics of general LP^A programs. In [Faber *et al.*, 2004], the authors have shown that arbitrary (including nonmonotone) aggregates do not increase the complexity of disjunctive programs. However, nonmonotone aggregates do increase the complexity of reasoning on Or-free programs.⁶

Theorem 21 *Cautious reasoning over $\text{LP}_{m,a,n}^A$ is Π_2^P -complete.*

Proof. Membership follows directly from the results in [Faber *et al.*, 2004]. Concerning hardness, we provide a reduction from 2QBF. Let $\Psi = \forall x_1, \dots, x_m \exists y_1, \dots, y_n : \Phi$, where w.l.o.g. Φ is a propositional formula in 3CNF format, over precisely the variables $x_1, \dots, x_m, y_1, \dots, y_n$. Observe that Ψ is equivalent to $\neg \exists x_1, \dots, x_m \forall y_1, \dots, y_n : \neg \Phi$, and that $\neg \Phi$ is equivalent to a 3DNF where every literal has reversed polarity w.r.t. Φ . Let the $\text{LP}_{m,a,n}^A$ program Π^{Ψ} be:

⁵Recall that we are dealing only with aggregates whose function is computable in polynomial time.

⁶In [Ferraris, 2004] it was independently shown that deciding answer set existence for a program with weight constraints (possibly containing negative weights) is Σ_2^P -complete.

$t(x_i, 1) : -\# \text{sum}\{V : t(x_i, V)\} >= 0.$
 $t(x_i, -1) : -\# \text{sum}\{V : t(x_i, V)\} <= 0.$
 $t(y_i, 1) : -\# \text{sum}\{V : t(y_i, V)\} >= 0.$
 $t(y_i, -1) : -\# \text{sum}\{V : t(y_i, V)\} <= 0.$
 $t(y_i, 1) : -\text{unsat}. \quad t(y_i, -1) : -\text{unsat}.$

For each clause $c_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ of the original Φ , we add: $\text{unsat} : -\mu(l_{i,1}), \mu(l_{i,2}), \mu(l_{i,3})$, where $\mu(l)$ is $t(l, -1)$ if l is positive, and $t(l, 1)$ otherwise.

The query not unsat? holds for Π^{Ψ} , iff Ψ is satisfiable. $\square$

Monotone and antimonotone aggregates, instead, behave as in the disjunctive case, not causing any complexity gap.

Theorem 22 *Cautious reasoning over $\text{LP}_{m,a}^A$ is co-NP-complete.*

Proof. Hardness follows from the co-NP-hardness of cautious reasoning over normal (aggregate-free) logic programs [Marek and Truszczyński, 1991; Schlipf, 1995]. For membership, we guess a total interpretation M , and check that: (i) $A \in M$, and (ii) $\neg M^- = \text{GUS}_{\mathcal{P}}(M)$ (by Theorem 14 M is then an answer set). By Theorem 20, the checks are feasible in polynomial time. $\square$

7 Related Work

To our knowledge, the only other work in which the notion of unfounded set has been defined for programs with aggregates is [Kemp and Stuckey, 1991]. However, their definition ignores aggregates in the second condition for unfounded sets. For the program $a(1) :- \# \text{count}\{X : a(X)\} > 0$, the well-founded model of [Kemp and Stuckey, 1991] is $\emptyset$, leaving $a(1)$ undefined. Our well-founded model is $\{\neg a(1)\}$. Most of the results reported in this paper do not hold for unfounded sets as defined in [Kemp and Stuckey, 1991].

There have been several attempts to define well-founded semantics for programs with aggregates, not relying on unfounded sets. Several early approaches which are defined on a limited framework are discussed in [Kemp and Stuckey, 1991]. In [Van Gelder, 1992] a semantics is defined by compiling aggregates to rules with standard atoms. This approach was generalized in [Osorio and Jayaraman, 1999]. In any case, this strategy works only for certain classes of programs. In [Ross and Sagiv, 1997] an operator-based definition is given, which also works only on a restricted set of programs. In [Pelov, 2004] a well-founded semantics has been defined based on an approximating operator. Since this definition is substantially different from the one in this paper, we leave a comparison for future work.

Other works attempted to define stronger notions of well-founded semantics (also for programs with aggregates), among them the Ultimate Well-Founded Semantics of [Decker *et al.*, 2001] and WFS^1 and WFS^2 of [Dix and Osorio, 1997]. Whether a characterization in terms of unfounded sets can exist for these semantics is not clear, and even if such generalized unfounded sets would exist, it is likely that some of the theorems in this paper will no longer hold, given that these semantics assign truth or falsity for more atoms.

In [Ferraris, 2004] it was shown that the semantics of Smodels programs with positive weight constraints is equal to answer sets as defined in [Faber *et al.*, 2004] on the respective fragment. Since by Theorem 16 $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ approximates

answer sets of [Faber *et al.*, 2004], $\mathcal{W}_P^\omega(\emptyset)$ can be used also as an approximating operator for the respective Smodels programs. Indeed, we can show that the *AtMost* pruning operator of Smodels [Simons *et al.*, 2002] is a special case of the Φ_I operator (defined in the proof sketch for Theorem 20).

8 Applications and Conclusion

The semantics of logic programs with aggregates is not straightforward, especially in presence of recursive aggregates. The declarative and fixpoint characterizations of answer sets, provided in Sections 4 and 5, allow for a better understanding of the meaning of programs with aggregates, and provide a handle on effective methods for computing answer sets for programs with (recursive) aggregates. In particular, the operator $\mathcal{W}_P^\omega(\emptyset)$ can be used first to compute what must be in any answer set. Later in the computation, it can be used as a pruning operator and for answer set checking (as described in [Koch *et al.*, 2003; Pfeifer, 2004]).

Furthermore, since loop formulas encode unfounded sets [Lee, 2004], our results should be adaptable also to SAT-based ASP systems, all of which rely on loop formulas.

The complexity results make a clear demarcation between aggregates from the computational viewpoint, which is very useful to pick the appropriate techniques to be employed for the computation. The well-founded semantics of $\text{LP}_{m,a}^A$ is efficiently computable. Answer set semantics is in co-NP for $\text{LP}_{m,a}^A$, while nonmonotone aggregates bring about a complexity gap, and cannot be easily accommodated in NP systems.

A main concern for future work is therefore the exploitation of our results for the implementation of recursive aggregates in ASP systems.

References

- [Baral, 2002] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. CUP, 2002.
- [Calimeri *et al.*, 2002] F. Calimeri, W. Faber, N. Leone, and G. Pfeifer. Pruning Operators for Answer Set Programming Systems. In *NMR'2002*, pp. 200–209, 2002.
- [Dell'Armi *et al.*, 2003] T. Dell'Armi, W. Faber, G. Ielpa, N. Leone, and G. Pfeifer. Aggregate Functions in DLV. In *ASP'03*, pp. 274–288, Messina, Italy, 2003. Online at <http://CEUR-WS.org/Vol-78/>.
- [Denecker *et al.*, 2001] M. Denecker, N. Pelov, and M. Bruynooghe. Ultimate Well-Founded and Stable Model Semantics for Logic Programs with Aggregates. In *ICLP-2001*, pp. 212–226, 2001.
- [Dix and Osorio, 1997] J. Dix and M. Osorio. On Well-Behaved Semantics Suitable for Aggregation. In *ILPS '97*, Port Jefferson, N.Y., 1997.
- [Faber *et al.*, 2004] W. Faber, N. Leone, and G. Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *JELIA 2004*, pp. 200–212, 2004.
- [Ferraris, 2004] P. Ferraris. Answer Sets for Propositional Theories. <http://www.cs.utexas.edu/users/otto/papers/proptheories.ps>, 2004.
- [Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *NGC*, 9:365–385, 1991.
- [Gelfond, 2002] M. Gelfond. Representing Knowledge in A-Prolog. In *Computational Logic. Logic Programming and Beyond*, LNCS 2408
- [Kemp and Stuckey, 1991] D.B. Kemp and P.J. Stuckey. Semantics of Logic Programs with Aggregates. In *ISLP'91*, pp. 387–401. MIT Press, 1991.
- [Koch *et al.*, 2003] C. Koch, N. Leone, and G. Pfeifer. Enhancing Disjunctive Logic Programming Systems by SAT Checkers. *Artificial Intelligence*, 15(1–2):177–212, 2003.
- [Lee, 2004] J. Lee. A Model-Theoretic Counterpart of Loop Formulas. <http://www.cs.utexas.edu/users/appsmurf/papers/mtclf.pdf>, 2004.
- [Leone *et al.*, 1997] N. Leone, P. Rullo, and F. Scarcello. Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics and Computation. *Information and Computation*, 135(2):69–112, 1997.
- [Marek and Truszczyński, 1991] V.W. Marek and M. Truszczyński. Computing Intersection of Autoepistemic Expansions. In *LPNMR'91*, pp. 37–50, 1991.
- [Osorio and Jayaraman, 1999] M. Osorio and B. Jayaraman. Aggregation and Negation-As-Failure. *NGC*, 17(3):255–284, 1999.
- [Pelov and Truszczyński, 2004] N. Pelov and M. Truszczyński. Semantics of disjunctive programs with monotone aggregates - an operator-based approach. In *NMR 2004*, pp. 327–334, 2004.
- [Pelov *et al.*, 2004] N. Pelov, M. Denecker, and M. Bruynooghe. Partial stable models for logic programs with aggregates. In *LPNMR-7*, LNCS 2923
- [Pelov, 2004] N. Pelov. *Semantics of Logic Programs with Aggregates*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 2004.
- [Pfeifer, 2004] G. Pfeifer. Improving the Model Generation/Checking Interplay to Enhance the Evaluation of Disjunctive Programs. In *LPNMR-7*, LNCS, pp. 220–233, 2004.
- [Ross and Sagiv, 1997] K. A. Ross and Y. Sagiv. Monotonic Aggregation in Deductive Databases. *JCSS*, 54(1):79–97, 1997.
- [Schlipf, 1995] J.S. Schlipf. The Expressive Powers of Logic Programming Semantics. *JCSS*, 51(1):64–86, 1995.
- [Simons *et al.*, 2002] P. Simons, I. Niemelä, and T. Soininen. Extending and Implementing the Stable Model Semantics. *Artificial Intelligence*, 138:181–234, 2002.
- [Tarski, 1955] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math*, 5:285–309, 1955.
- [Van Gelder *et al.*, 1991] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The Well-Founded Semantics for General Logic Programs. *JACM*, 38(3):620–650, 1991.
- [Van Gelder, 1992] Allen Van Gelder. The Well-Founded Semantics of Aggregation. In *PODS'92*, pp. 127–138. ACM Press, 1992.

Parameterized Compilability

Hubie Chen

Departament de Tecnologia
Universitat Pompeu Fabra
Barcelona, Spain
hubie.chen@upf.edu

Abstract

Compilability is a measure of how effectively compilation (or preprocessing) can be applied to knowledge bases specified in a particular knowledge representation formalism; the aim of compilation is to allow for efficient, on-line query processing. A theory of compilability has been established for organizing knowledge representation formalisms according to a scheme of “compilability classes”, and bears strong analogies to the classical theory of complexity, which permits the organization of computational problems according to complexity classes. We develop a novel theory of compilability, called *parameterized compilability*, which incorporates the notion of parameterization as used in parameterized complexity and permits for refined analysis of compilability.

1 Introduction

Compilability. Many computational reasoning problems involve deciding whether or not a query is entailed by a knowledge base (or database). Such reasoning problems are often computationally intractable. A methodology for coping with this intractability is *compilation* (also called *preprocessing*), which involves translating a knowledge base into a form that allows for more efficient query processing. Compilation is appropriate for knowledge bases that remain stable over time and will be queried numerous times, even if the compilation itself cannot be performed efficiently, since the cost of the compilation will be amortized by the speed-up in query processing. Although recent years have seen many interesting results on compilation, the basic idea of simplifying a sequence of computations by preprocessing a set of useful values is as old as mathematics, as discussed in [Cadoli *et al.*, 2002].

A systematic theory. A number of results in the nineties demonstrated non-compilability results, which show, for instance, that the knowledge bases of a particular knowledge representation formalism are not compilable to those of another formalism in which query processing can be performed in polynomial time. The initial results of this type, including

[Cadoli *et al.*, 1996; 1997; 1999; Gogic *et al.*, 1995], were based on *ad hoc* proofs that did not explicitly use a unified proof technique.

In [Cadoli *et al.*, 2002], a robust theory of compilability was developed that made it possible to systematically organize knowledge representation formalisms according to their compilability, much in the way that classical complexity theory made it possible to systematically organize languages according to their (in)tractability. In particular, the theory of compilability provides a way to define, for every classical complexity class C , an analogous compilability class $\text{comp-}C$; the theory also provides a notion of reduction for use with the compilability classes $\text{comp-}C$. The compilability class comp-P , which is the analog of P (those languages decidable in polynomial time), contains those formalisms that are compilable to a second formalism in which query processing can be performed in polynomial time. In saying that a formalism is compilable to a second formalism, we mean (roughly) that for every knowledge base x of the first formalism, there is a knowledge base $f(x)$ in the second formalism having the same answers to queries as x and such that the size of $f(x)$ is always polynomial in the size of x . This “polynomial size” requirement models the intuition that compilation is only useful if it does not greatly increase the size of a knowledge base.

Demonstration that a formalism is in comp-P is a positive compilation result, analogous to demonstration that a language is in P , a positive tractability result. Likewise, demonstration that a formalism is comp-NP-hard is a negative compilation result and implies that the formalism is not in comp-P (under a complexity-theoretic assumption), just as demonstration that a language is NP-hard implies that the language is not in P (under the standard assumption that $P \neq NP$).

First in [Cadoli *et al.*, 2002; 2000] and later in other papers including [Liberatore, 1997; Liberatore and Schaerf, 2000; Liberatore, 2000; 2001], the theory of compilability presented in [Cadoli *et al.*, 2002] was applied successfully to classify knowledge representation formalisms that have appeared in the artificial intelligence literature. In particular, a multitude of formalisms, including propositional logic, circumscription, and default logic, have been demonstrated to be contained in comp-P or complete for the analog $\text{comp-}C$ of a standard complexity class C . The downside of these

classification results is that many of them are negative, showing that a formalism is complete for the compilability version of an intractable complexity class, such as comp-NP or comp-co-NP.

Parameterized complexity. The abundance of negative compilability results mirrors a state of affairs in complexity theory: although many problems of interest can be classified according to the developed scheme of complexity classes (P, NP, co-NP, etc.), there is an abundance of negative results that show completeness or hardness for classes believed to be strictly larger than P. Because such generally intractable problems do need to be dealt with in practice, a number of theoretical frameworks have been developed to provide a finer analysis of hard problems than that offered by standard complexity theory. Examples of such frameworks include the theory of approximation algorithms and average-case analysis. A relatively recent alternative framework is *parameterized complexity* [Downey and Fellows, 1999b].

Parameterized complexity is a theory that, as with classical complexity theory and compilability theory, provides classes and a notion of reduction for categorizing objects. In the case of parameterized complexity, the objects categorized are parameterized languages: languages with instances consisting of two parts, a *main part* and a *parameter*. Many languages that have been classically studied can be viewed naturally as parameterized languages. For example, each instance of the VERTEX COVER problem can be viewed as consisting of a graph G as the main part, and a natural number k as the parameter; the question is to decide whether or not the graph G has a vertex cover of size at most k . Similarly, the INDEPENDENT SET problem may be viewed as a parameterized language: an instance is again a graph G paired with a natural number k , and the question is to decide whether or not the graph G has an independent set of size at least k .

The parameterized flavor of tractability is called *fixed-parameter tractability*; formally, a parameterized language is fixed-parameter tractable (or, in the class FPT) if there exists a constant c such that for every k , instances with main part of size n and parameter k can be solved in time $O(n^c)$. As an example, VERTEX COVER can be solved in time $O(r^k k^2 + kn)$, where r is a constant, k is the parameter, and n is the size of the main part; consequently, for any fixed k , instances of VERTEX COVER with main part of size n and parameter k can be solved in time $O(n)$, and so VERTEX COVER is considered to be fixed-parameter tractable. Because the running time of fixed-parameter tractable languages exhibits a restricted form of dependence on the parameter, such languages are often solvable in practice when the parameter falls into a limited range. (In fact, one can see from the given time bound for VERTEX COVER that this problem can be solved in polynomial time when the parameter k is $O(\log n)$.) On the other hand, INDEPENDENT SET is believed to require time where the main part exhibits an exponential dependence on the parameter – that is, a time of form similar to $\Omega(n^k)$. Indeed, INDEPENDENT SET is complete for a parameterized complexity class believed to properly contain FPT, and accordingly is believed to be outside of FPT.

Viewed classically, these two example languages are both NP-complete, and hence demonstrate that parameterized complexity is somewhat orthogonal to classical complexity; moreover, the example of VERTEX COVER indicates that parameterized complexity can provide tractability results where classical complexity does not. Indeed, the allowance of a non-polynomial dependence on the parameter of an input instance in fixed-parameter tractability permits the use of algorithmic ideas not utilized in proving standard polynomial-time tractability results, and the field of parameterized complexity has developed a rich and mathematically deep suite of algorithmic techniques.

Parameterized compilability. The contribution of this paper is *parameterized compilability*, a new theory of compilability incorporating the notion of parameterization as used in parameterized complexity. Our theory provides a novel set of concepts for proving compilability results, allowing the use of the sophisticated toolkit that has been developed for proving fixed-parameter tractability and intractability results, and provides an avenue for refining the many negative non-compilability results that have been proved using the established theory of compilability. As articulated in [Downey *et al.*, 1999], one benefit of using a parameterized notion of computational tractability is that it becomes possible to engage in an extended dialogue with a single problem, by the investigation of a variety of parameters. We thus provide a theoretical framework in which we anticipate that the formulation and exploration of interesting parameters in the context of parameterized compilability will lead to refined analysis of compilability results.

The results in this paper are as follows. For every parameterized complexity class C , we define a compilability variant $\text{par-comp-}C$. We also define a notion of reduction for use with the $\text{par-comp-}C$ classes, and demonstrate its robustness – in particular, that it is transitive and compatible (in a formal sense) with the classes $\text{par-comp-}C$. We also develop a general method for demonstrating hardness of a formalism for a class $\text{par-comp-}C$, and study the structure of the classes $\text{par-comp-}C$, relating this structure directly to the structure of non-uniform versions of parameterized complexity classes C .

2 Preliminaries

While we have attempted to make this paper as self-contained as possible, some familiarity with the basic notions of complexity theory (including non-uniform complexity classes), parameterized complexity, and compilability theory will be helpful; we name [Balcázar *et al.*, 1995], [Downey and Fellows, 1999b], and [Cadoli *et al.*, 2002] as references for these topics, respectively. We also recommend the surveys [Downey and Fellows, 1999a; Downey *et al.*, 1999] on parameterized complexity.

2.1 Definitions and Notation

Strings. We assume Σ to be a fixed finite alphabet that is used to form strings. The length of a string x is denoted by $|x|$. We will at times assume that pairs of strings (that

is, elements of $\Sigma^* \times \Sigma^*$ are represented as strings (that is, elements of Σ^*) via a pairing function $\langle \cdot, \cdot \rangle$. We make standard assumptions about this pairing function, namely, that the length of $\langle x, y \rangle$ is linear in $|x| + |y|$; $\langle x, y \rangle$ can be computed in time polynomial in $|x| + |y|$; there are polynomial-time computable projection functions $\pi_1, \pi_2 : \Sigma^* \rightarrow \Sigma^*$ such that $\pi_1(\langle x, y \rangle) = x$ and $\pi_2(\langle x, y \rangle) = y$; and $\langle \cdot, \cdot \rangle$ is length-monotone, that is, if $|y| \leq |z|$ then $|\langle x, y \rangle| \leq |\langle x, z \rangle|$ and $|\langle y, x \rangle| \leq |\langle z, x \rangle|$. We also assume that there is a tripling function $\langle \cdot, \cdot, \cdot \rangle$ for representing triples of strings as single strings, for which similar assumptions hold.

Functions. As usual, we consider a function $p : \mathbb{N} \rightarrow \mathbb{N}$ to be a *polynomial* if there exists a constant $c \in \mathbb{N}$ such that $p(n)$ is $O(n^c)$. A function $p : \mathbb{N} \times \Sigma^* \rightarrow \mathbb{N}$ is a *parameterized polynomial* if there exists a constant $c \in \mathbb{N}$ and a function $h : \Sigma^* \rightarrow \mathbb{N}$ such that for all $(n, k) \in \mathbb{N} \times \Sigma^*$, $p(n, k) \leq h(k)(n + 1)^c$. In other words, $p : \mathbb{N} \times \Sigma^* \rightarrow \mathbb{N}$ is a parameterized polynomial if there exists a constant $c \in \mathbb{N}$ such that for each $k \in \Sigma^*$, the function $p_k : \mathbb{N} \rightarrow \mathbb{N}$ defined by $p_k(n) = p(n, k)$ is $O(n^c)$.

A function $f : \Sigma^* \rightarrow \Sigma^*$ is *polynomial-size* if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $x \in \Sigma^*$, $|f(x)| \leq p(|x|)$. Similarly, a function $f : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is *parameterized polynomial-size* if there exists a parameterized polynomial $p : \mathbb{N} \times \Sigma^* \rightarrow \mathbb{N}$ such that for all $(x, k) \in \Sigma^* \times \Sigma^*$, $|f(x, k)| \leq p(|x|, k)$.

A function $g : \Sigma^* \rightarrow \Sigma^*$ is *polynomial-time computable* if there exist a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a Turing machine M such that for all $x \in \Sigma^*$, the Turing machine M , on input x , produces output $g(x)$ in time less than $p(|x|)$. Similarly, a function $g : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is *parameterized polynomial-time computable* if there exist a parameterized polynomial $p : \mathbb{N} \times \Sigma^* \rightarrow \mathbb{N}$ and a Turing machine M such that for all $(x, k) \in \Sigma^* \times \Sigma^*$, the Turing machine M , on input $\langle x, k \rangle$, produces output $g(x, k)$ in time less than $p(|x|, k)$.

A function $h : \Sigma^* \rightarrow \Sigma^*$ is *recursively computable* if there exists a Turing machine M such that for all $x \in \Sigma^*$, the Turing machine M , on input x , produces output $h(x)$.

Languages and Complexity Classes. A language is a subset of Σ^* , that is, a set of strings. A *parameterized language* is a subset of $\Sigma^* \times \Sigma^*$, that is, a set of pairs of strings. A (parameterized) *complexity class* is a set of (parameterized) languages. When C is a (parameterized) complexity class and $\leq$ is a reduction, we say that C is *compatible* with $\leq$ if for all (parameterized) languages A and B , $A \leq B$ and $B \in C$ imply that $A \in C$. Also, when C is a (parameterized) complexity class and $\leq$ is a reduction, we say that B is *hard* for C (alternatively, is *C-hard*) under $\leq$ reductions if for all $A \in C$, $A \leq B$; and, we say that B is *complete* for C (alternatively, is *C-complete*) under $\leq$ reductions if $B \in C$ and B is *C-hard*.

When C is a complexity class, the non-uniform version (or “advice” version) of C , denoted by C/poly , contains those languages A such that there exists a polynomial-size function $f : 1^* \rightarrow \Sigma^*$ and a language B in C such that for all $x \in \Sigma^*$, $x \in A$ if and only if $\langle f(1^{|x|}), x \rangle \in B$. When C is a parameterized complexity class, the non-uniform version

of C , denoted by C/ppoly , contains those parameterized languages A such that there exists a parameterized polynomial-size function $f : 1^* \times \Sigma^* \rightarrow \Sigma^*$ and a language B in C such that for all $(x, k) \in \Sigma^* \times \Sigma^*$, $(x, k) \in A$ if and only if $(\langle f(1^{|x|}, k), x \rangle, k) \in B$.

When L is a parameterized language, define ϵL to be the set $\{(\epsilon, y, k) : (y, k) \in L\}$.

Knowledge Representation Formalisms. A *knowledge representation formalism (KRF)* is a subset of $\Sigma^* \times \Sigma^*$. A *parameterized knowledge representation formalism (PKRF)* is a subset of $\Sigma^* \times \Sigma^* \times \Sigma^*$.

Conventions. We view a ternary function $f : \Sigma^* \times \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ as a binary function $g : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ by pairing together the first two arguments, that is, $g(\langle x, y \rangle, z) = f(x, y, z)$. For example, a ternary function $f : \Sigma^* \times \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is considered to be parameterized polynomial-size if there exists a parameterized polynomial $p : \mathbb{N} \times \Sigma^* \rightarrow \mathbb{N}$ such that for all $(x, y, k) \in \Sigma^* \times \Sigma^* \times \Sigma^*$, $|f(x, y, k)| \leq p(|\langle x, y \rangle|, k)$. Similarly, a PKRF F is viewed as a parameterized language by pairing together the first two strings of each triple, and so is considered to belong to a parameterized complexity class C if the language $L = \{(\langle x, y \rangle, k) : (x, y, k) \in F\}$ belongs to C .

For readability, we at times use a natural number $m \in \mathbb{N}$ in place of its unary representation 1^m .

2.2 Compilability

In this subsection, we review the theory of compilability; the definitions and theorems of this subsection are all from [Cadoli et al., 2002], on which our presentation is based. Throughout this subsection, the following two assumptions concerning complexity classes and languages are in effect, just as in [Cadoli et al., 2002].

Assumption 1 Every complexity class C is compatible with and has complete problems under the polynomial-time many-one reduction $\leq_m^p$.

Assumption 2 In languages, suffixed blanks are considered to be irrelevant, that is, a string $x\Box$ is considered to be in a language A if x is in A .

For a KRF F , we will refer to each $x \in \pi_1(F)$ as a *knowledge base*; a knowledge base x is considered to represent the knowledge $\{y \in \Sigma^* : (x, y) \in F\}$. Roughly speaking, a KRF is considered to be compilable to a complexity class C if following a preprocessing of each knowledge base x , posing a query y to the knowledge base x – that is, deciding if $(x, y) \in F$ – can be performed in C . The formal definition of those KRFs compilable to a complexity class C includes two ingredients. The first is a preprocessing function f ; because preprocessing of a knowledge base is not useful if it greatly increases the size of the knowledge base, f is required to be of polynomial size. The second is a “target” KRF F' , mandated to be inside the class C , to which the KRF F is translated via the function f .

Definition 3 (comp-C) Let C be a complexity class. A KRF F belongs to *comp-C* if there exist

- a binary polynomial-size function $f : \Sigma^* \times 1^* \rightarrow \Sigma^*$, and
- a KRF F' in C

such that for all pairs $(x, y) \in \Sigma^* \times \Sigma^*$,

$$(x, y) \in F \text{ if and only if } (f(x, |y|), y) \in F'.$$

One can obtain a compilability class $\text{comp-}C$ from any complexity class C . Hence, the $\text{comp-}C$ classes offer a rich scheme by which one can classify KRFs, just as conventional complexity classes offer a rich scheme by which one can classify computational problems. Upon a first reading of Definition 3, it may be conceptually easiest for one to take the complexity class C to be equal to P (polynomial-time). In this case, the requirement that F' is in C means that queries (x, y) posed to F' can be processed efficiently, so f translates each F -knowledge base x – which is not necessarily in a form that allows for efficient resolution of queries (x, y) – into a F' -knowledge base $f(x)$ – which is in a form that allows for efficient resolution of queries (x, y) .¹

The following notion of reduction, associated with the $\text{comp-}C$ classes, allows one to compare the compilability of different KRFs.

Definition 4 (*comp reducibility*) A KRF F is *comp reducible* to a KRF F' (denoted by $F \leq_{\text{comp}} F'$) if there exist

- binary polynomial-size functions $f_1, f_2 : \Sigma^* \times 1^* \rightarrow \Sigma^*$ and
- a binary polynomial-time computable function $g : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

such that for all pairs $(x, y) \in \Sigma^* \times \Sigma^*$,

$$(x, y) \in F \text{ if and only if } (f_1(x, |y|), g(f_2(x, |y|), y)) \in F'.$$

Theorem 5 For every complexity class C , the *comp reduction* is transitive and is compatible with the class $\text{comp-}C$.

The structure of the compilability classes $\text{comp-}C$ can be directly related to the structure of the complexity classes C/poly .

Theorem 6 Let C and C' be complexity classes. The containment $\text{comp-}C \subseteq \text{comp-}C'$ holds if and only if the containment $C/\text{poly} \subseteq C'/\text{poly}$ holds.

When C and C' are complexity classes that obey the proper containment $C \subsetneq C'$, Theorem 6 gives strong evidence that the corresponding compilability classes $\text{comp-}C$ and $\text{comp-}C'$ obey a similar proper containment: $\text{comp-}C \subsetneq \text{comp-}C'$. Note that in making this observation, we appeal to the widely held belief that relationships between “standard” complexity classes C, C' are not affected by non-uniformity,

¹This definition of $\text{comp-}C$ is “non-uniform” in that the preprocessing or translation function f not only takes the knowledge base x as input, but in addition is given the length of the query y . A “uniform” version of the classes $\text{comp-}C$ where the translation function f is only given the knowledge base x is studied and compared to the $\text{comp-}C$ classes in [Cadoli et al., 2002]. In this paper, we present a non-uniform theory of parameterized compilability, and do not further discuss the uniform $\text{comp-}C$ classes. One can use the ideas and concepts we present in this paper to develop a corresponding uniform theory.

that is, C is properly contained in C' if and only if C'/poly is properly contained in C'/poly .²

2.3 Parameterized Complexity

Parameterized complexity is a theory for the classification of parameterized languages – languages whose constituents have two components. In classical complexity theory, the base complexity class modeling feasible computation is P , which is defined using the notion of polynomial-time; in parameterized complexity theory the base complexity class is FPT (“fixed-parameter tractable”), which is definable using the notion of parameterized polynomial-time.

Definition 7 (*FPT*) A parameterized language L belongs to the parameterized complexity class FPT if there exists a parameterized polynomial-time computable function $g : \Sigma^* \times \Sigma^* \rightarrow \{0, 1\}$ such that for all pairs $(x, k) \in \Sigma^* \times \Sigma^*$,

$$(x, k) \in L \text{ if and only if } g(x, k) = 1.$$

Roughly speaking, a parameterized language L is in the class FPT if there exists a constant $c \in \mathbb{N}$ such that each “ k -slice” $L_k \stackrel{\text{def}}{=} \{x : (x, k) \in L\}$ of L is computable in polynomial time via a polynomial of degree c . The degree of the polynomials bounding the time for L_k must be a “universal” constant c and cannot depend on k ; on the other hand, the coefficients of these polynomials may have arbitrary dependence on k .

The following notion of reducibility may be used to compare the complexity of parameterized languages.

Definition 8 (*parameterized reducibility*) A parameterized language L is parameterized reducible to a parameterized language L' (denoted by $L \leq^{\text{par}} L'$) if there exist

- a parameterized polynomial-time computable function $f : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$, and
- a recursively computable function $h : \Sigma^* \rightarrow \Sigma^*$

such that for all pairs $(x, k) \in \Sigma^* \times \Sigma^*$,

$$(x, k) \in L \text{ if and only if } (f(x, k), h(k)) \in L'.$$

We now define two parameterized complexity classes. In contrast to the class FPT , each of these two classes is defined directly as those languages reducible to a particular language.

Definition 9 We define the following parameterized languages; the weight of a truth assignment is defined to be the number of variables it maps to true.

- **WEIGHTED FORMULA SATISFIABILITY**
Input: A boolean formula F .
Question: Does F have a satisfying assignment of weight at most k ?
- **WEIGHTED CIRCUIT SATISFIABILITY**
Input: A boolean circuit C .
Question: Does C have a satisfying assignment of weight at most k ?

²Note that the term “non-uniformity” is intended here to take on the usual complexity-theoretic sense, as opposed to the slightly different sense in which $\text{comp-}C$ was called non-uniform.

Definition 10 We define the following parameterized complexity classes.

- $W[\text{SAT}]$ is the class of all parameterized languages that are parameterized reducible to $\text{WEIGHTED FORMULA SATISFIABILITY}$.
- $W[\text{P}]$ is the class of all parameterized languages that are parameterized reducible to $\text{WEIGHTED CIRCUIT SATISFIABILITY}$.

It is straightforward to verify that the parameterized complexity classes defined here (FPT , $W[\text{SAT}]$, and $W[\text{P}]$) are compatible with the parameterized reduction. Throughout this paper, we will assume that all discussed parameterized complexity classes have this property, and also that they have complete problems.

Assumption 11 Every parameterized complexity class C is compatible with and has complete problems under the parameterized reduction $\leq^{\text{par}}$.

3 Parameterized Compilability

3.1 Classes and reductions

As with the theory of compilability, the starting point for the theory of parameterized compilability is the definition of compilability classes. Whereas a compilability version of each non-parameterized complexity class was defined (Definition 3), here we define a compilability version $\text{par-comp-}C$ of each parameterized complexity class C .

As with the definition of $\text{comp-}C$, the definition of $\text{par-comp-}C$ includes a translation function f with a space bound, as well as a “target” PKRF F' , which is required to be inside C .

Definition 12 Let C be a parameterized complexity class. A PKRF F belongs to $\text{par-comp-}C$ if there exist

- a parameterized polynomial-size function $f : \Sigma^* \times 1^* \times \Sigma^* \rightarrow \Sigma^*$, and
- a PKRF F' in C

such that for all triples $(x, y, k) \in \Sigma^* \times \Sigma^* \times \Sigma^*$ and natural numbers $m \geq |y|$,

$$(x, y, k) \in F \text{ if and only if } (f(x, m, k), y, k) \in F'.$$

An implication of the containment of a PKRF F inside $\text{par-comp-}C$ via the function f and the PKRF F' is the following. Define, for each $k \in \Sigma^*$, the “ k -slice” of a PKRF G to be $G_k \stackrel{\text{def}}{=} \{(x, y) : (x, y, k) \in G\}$. Moreover, define $f_k : \Sigma^* \times 1^* \rightarrow \Sigma^*$ to be the function $f_k(x, m) = f(x, m, k)$. Notice that for each $k \in \Sigma^*$, the KRFs F_k , F'_k and the binary function f_k satisfy the condition of Definition 3. That is, each k -slice F_k of F is compilable to the corresponding k -slice F'_k of F' by the function f_k . Because the size of f is bounded by a parameterized polynomial, for k falling into a limited range, the functions f_k will behave polynomially; likewise, when $F' \in \text{FPT}$, for k falling into a limited range, the KRF slices F'_k will have time complexity behaving polynomially.

Upon initial acquaintance, Definition 12 may appear to be complex. However, we believe that the classes $\text{par-comp-}C$ unify the classes $\text{comp-}C$ with parameterized complexity

classes in a natural way. We substantiate this claim by considering two special cases of Definition 12, which may lend some intuition to the reader.

First, consider the case where the preprocessing function f simply returns as output its first argument, that is, f is equal to the projection π_1 . It follows that F must equal F' ; consequently, a PKRF F is contained in $\text{par-comp-}C$ via such an f if and only if F is contained in C itself. Intuitively, the function f does not perform any preprocessing, so the compilability class $\text{par-comp-}C$ simplifies out to the original class C . Next, consider the case where the PKRF F has the property that for all strings $x, y, k, k' \in \Sigma^*$, it holds that $(x, y, k) \in F$ if and only if $(x, y, k') \in F$; that is, F “ignores” the third component (the “parameter”) of its strings. In this case, the PKRF F naturally induces a KRF $\tilde{F}$, defined to be

$$\{(x, y) : \exists k. (x, y, k) \in F\} = \{(x, y) : \forall k. (x, y, k) \in F\}.$$

Such a PKRF F is contained in par-comp-FPT if and only if the induced KRF $\tilde{F}$ is contained in comp-P .³ In “ignoring” the parameter of the PKRF F , the compilability class par-comp-FPT simplifies out to its non-parameterized variant, namely, comp-P . We mention that by imposing the restrictions discussed in the two cases simultaneously, we effectively obtain the class P . Thus, one can think of FPT and comp-P as two different generalizations of P which are unified together in and can be obtained as specializations of par-comp-FPT .

We have the following notion of reducibility for comparing the compilability of PKRFs.

Definition 13 (*par-comp reducibility*) A PKRF F is *par-comp-reducible* to a PKRF F' (denoted by $F \leq_{\text{comp}}^{\text{par}} F'$) if there exist

- parameterized polynomial-size functions $f_1, f_2 : \Sigma^* \times 1^* \times \Sigma^* \rightarrow \Sigma^*$,
- a parameterized polynomial-time function $g : \Sigma^* \times \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$, and
- a recursively computable function $h : \Sigma^* \rightarrow \Sigma^*$

such that for all triples $(x, y, k) \in \Sigma^* \times \Sigma^* \times \Sigma^*$ and natural numbers $m \geq |y|$,

$$(x, y, k) \in F \text{ if and only if } (f_1(x, m, k), g(f_2(x, m, k), y, k), h(k)) \in F'.$$

We note that our definition of par-comp reducibility is not a direct analog of the definition of comp-reducibility ; in Definition 13 the “preprocessing phase” f is given an upper bound m on the size of the query y , as opposed to the length of the query $|y|$, as in Definition 4. We believe that our notion of reduction is preferable because an assumption like Assumption 2 is not necessary to establish transitivity, and also because it allows for a simpler proof of transitivity. Our proof of transitivity can be specialized to a proof of transitivity for an “upper bound” variant of Definition 4 by removing the parameter k .

³Sketch of one direction of this claim: suppose that the PKRF F is of the described form and in par-comp-FPT . Fix an arbitrary $s \in \Sigma^*$, define the function $\tilde{f} : \Sigma^* \times 1^* \rightarrow \Sigma^*$ by $\tilde{f}(x, m) = f(x, m, s)$, and define the KRF $\tilde{F}$ as $\{(x, y) : (x, y, s) \in F'\}$. It can be verified that $\tilde{F}$ is contained in comp-P via $\tilde{f}$ and $\tilde{F}$.

Theorem 14 *The par-comp reduction is transitive.*

Theorem 15 *For every parameterized complexity class C , the par-comp reduction is compatible with the class par-comp- C .*

3.2 Completeness and Structure

We now study the notion of completeness for the classes par-comp- C , as well as the structure of the classes par-comp- C . First, we make a simple observation about the relationship between $C/ppoly$ and par-comp- C which will aid us in our study.

Proposition 16 *Let C be a parameterized complexity class. If the parameterized language A is in $C/ppoly$, then ϵA is in par-comp- C .*

The following theorem, which will be of help in proving hardness of PKRFs, demonstrates that we can easily obtain complete languages for the class par-comp- C from complete languages for the underlying class C .

Theorem 17 *Let C be a parameterized complexity class. If the parameterized language B is complete for C (under $\leq^{\text{par}}$ reductions), then ϵB is complete for par-comp- C (under $\leq^{\text{par}}_{\text{comp}}$ reductions).*

The next theorem is an analog of Theorem 6, and correspondingly provides strong evidence that the structure of the compilability classes par-comp- C reflects the structure of the parameterized complexity classes C .

Theorem 18 *Let C and C' be parameterized complexity classes. The containment par-comp- $C \subseteq \text{par-comp-}C'$ holds if and only if the containment $C/ppoly \subseteq C'/ppoly$ holds.*

References

- [Balcázar et al., 1995] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Texts in Theoretical Computer Science – An EATCS series. Springer-Verlag, Berlin, 2nd edition, 1995.
- [Cadoli et al., 1996] M. Cadoli, F.M. Donini, and M. Schaerf. Is intractability of non-monotonic reasoning a real drawback? *Artificial Intelligence*, 88:215–251, 1996.
- [Cadoli et al., 1997] M. Cadoli, F. M. Donini, M. Schaerf, and R. Silvestri. On compact representations of propositional circumscription. *Theoretical Computer Science*, 182:183–202, 1997.
- [Cadoli et al., 1999] Marco Cadoli, Francesco M. Donini, Paolo Liberatore, and Marco Schaerf. The size of a revised knowledge base. *Artificial Intelligence*, 115(1):25–64, 1999.
- [Cadoli et al., 2000] Marco Cadoli, Francesco M. Donini, Paolo Liberatore, and Marco Schaerf. Space efficiency of propositional knowledge representation formalisms. *Journal of Artificial Intelligence Research*, 13:1–31, 2000.
- [Cadoli et al., 2002] Marco Cadoli, Francesco M. Donini, Paolo Liberatore, and Marco Schaerf. Preprocessing of intractable problems. *Information and Computation*, 176(2):89–120, 2002.
- [Downey and Fellows, 1999a] R. Downey and M. Fellows. Parameterized complexity after (almost) 10 years: Review and open questions. In *Combinatorics, Computation and Logic, DMTCS'99 and CATS'99*, pages 1–33. Springer-Verlag, 1999.
- [Downey and Fellows, 1999b] R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
- [Downey et al., 1999] R. Downey, M. Fellows, and U. Stege. Parameterized complexity: A framework for systematically confronting computational intractability. In *Contemporary Trends in Discrete Mathematics*, (R. Graham, J. Kratochvil, J. Nešetřil and F. Roberts, eds.), *Proc. DIMACS-DIMATIA Workshop, Prague 1997*, volume 49 of *AMS-DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 49–99, 1999.
- [Gogic et al., 1995] Goran Gogic, Henry A. Kautz, Christos Papadimitriou, and Bart Selman. The comparative linguistics of knowledge representation. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 862–869, 1995.
- [Liberatore and Schaerf, 2000] Paolo Liberatore and Marco Schaerf. Compilability of abduction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000)*, 2000.
- [Liberatore, 1997] Paolo Liberatore. Compilability of domain descriptions in the language A. *Electronic Transactions on Artificial Intelligence*, 1(4):129–132, 1997.
- [Liberatore, 2000] Paolo Liberatore. Compilability and compact representations of revision of horn knowledge bases. *ACM Transactions on Computational Logic*, 1(1):131–161, 2000.
- [Liberatore, 2001] Paolo Liberatore. Monotonic reductions, representative equivalence, and compilation of intractable problems. *Journal of the ACM*, 48(6):1091–1125, 2001.

Updating Action Domain Descriptions*

Thomas Eiter, Esra Erdem, Michael Fink, and Ján Senko

Institute of Information Systems, Vienna University of Technology, Vienna, Austria

Email: (eiter | esra | michael | jan)@kr.tuwien.ac.at

Abstract

How can an intelligent agent update her knowledge base about an action domain, relative to some conditions (possibly obtained from earlier observations)? We study this question in a formal framework for reasoning about actions and change, in which the meaning of an action domain description can be represented by a directed graph whose nodes correspond to states and whose edges correspond to action occurrences. We define the update of an action domain description in this framework, and show among other results that a solution to this problem can be obtained by a divide-and-conquer approach in some cases. We also introduce methods to compute a solution and an approximate solution to this problem, and analyze the computational complexity of these problems. Finally, we discuss techniques to improve the quality of solutions.

1 Introduction

This paper discusses how an intelligent agent can modify her knowledge of the world, subject to some given conditions, in particular, when some new knowledge about the world is added. The given conditions can be domain-dependent (e.g., assertions obtained from earlier observations), or domain-independent (e.g., general properties of a knowledge base). The goal is to obtain a consistent knowledge base with minimal changes only to a variable part of the existing knowledge base such that the given conditions are satisfied.

We study the problem of knowledge base updates, briefly described above, in the context of reasoning about actions and change, so the agent's knowledge base is about domains of actions. For instance, consider an agent having the following knowledge, D , about a TV with remote control:

1. If the power is off, pushing the power button on the TV turns the power on.
2. If the power is on, pushing the power button on the TV turns the power off.
3. The TV is on whenever the power is on.
4. The TV is off whenever the power is off.

*Work supported by the Austrian Science Fund (FWF) grant P16536-N04 and the European Commission IST programme grants FET-2001-37004 WASP and IST-2001-33123 CologNeT.

Note that she does not know how a remote control works (e.g., she does not know the effect of pushing the power button on the remote control). Suppose that later she obtains the following information, Q , about remote controls:

1. If the power is on and the TV is off, pushing the power button on the remote control turns the TV on.
2. If the TV is on, pushing the power button on the remote control turns the TV off.

A desired condition, C , for the effective use of this TV system is the following:

Pushing the power button on the remote control is always possible.

If Q is added to D , then condition C is not satisfied: when the power and the TV are on, pushing the power button on the remote control is not possible, since Item 2 of Q and Item 3 of D above contradict. The question is then how the agent can update D by Q relative to C .

We describe domains of actions, like D above, in a fragment of the action language $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998], by “causal laws.” For instance, the direct effect of the action $PushPB_{TV}$ stated in Item 1 of D in the example above is described by the causal law

caused $PowerON$ **after** $PushPB_{TV} \wedge \neg PowerON$,

which expresses that the action $PushPB_{TV}$ causes the value of the fluent $PowerON$ to change from false to true; the indirect effect of this action stated in Item 3 of D is described by the causal law

caused $TvON$ **if** $PowerON$, (1)

which expresses that if the fluent $PowerON$ is caused to be true, then the fluent $TvON$ is caused to be true as well. An action (domain) description is partitioned into two: D_u (invariable) and D_m (variable). The meaning of an action description can be represented by a “transition diagram”—a directed graph whose nodes correspond to states and whose edges correspond to action occurrences.

We describe given conditions, like C above, in an action query language, similar to $\mathcal{R}$ [Gelfond and Lifschitz, 1998], by “queries”. For instance, condition C above can be described by the query

executable $\{PushPB_{RC}\}$. (2)

The main contributions of this paper briefly are as follows:

- Given an action description $D = (D_u, D_m)$, a set Q of causal laws, and a set C of conditions, what is an update of D by Q relative to C ? We answer this question in Section 3.

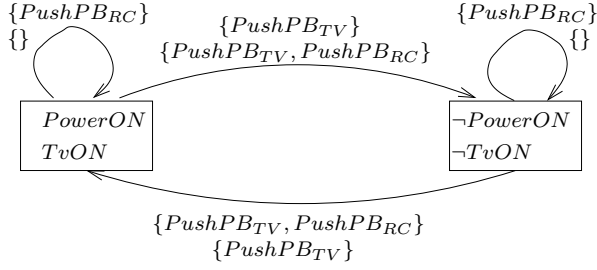


Figure 1: A transition diagram.

- Given an action description $D = (D_u, D_m)$, a set Q of causal laws, and a set C of conditions, consider the problem of computing an update of D by Q relative to C . We show that this problem is difficult in general (both Σ_3^P - and Π_3^P -hard), and that already checking a given solution is Π_3^P -complete. We also show that, under some conditions, these two problems are easier (Section 5).

- We provide conditions under which computing a solution to the problem above can be structurally decomposed. For instance, if the inputs of the problem satisfy the disjointness property, a divide-and-conquer approach is feasible (Section 4). We introduce methods to compute solutions and to compute “near-solutions” approximating them (Section 6).

Variations of the problem of updating knowledge bases have been studied in the AI literature, e.g., [Winslett, 1990; Katsuno and Mendelzon, 1991; Pereira *et al.*, 1991; Li and Pereira, 1996; Liberatore, 2000; Eiter *et al.*, 2002; Sakama and Inoue, 2003]. The problem addressed in this paper is more general (see Section 7 for a more detailed comparison).

2 Preliminaries

Transition diagrams. We start with a (*propositional*) *action signature* that consists of a set $\mathbf{F}$ of fluent names, and a set $\mathbf{A}$ of action names. An *action* is a truth-valued function on $\mathbf{A}$, denoted by the set of action names that are mapped to t .

A (*propositional*) *transition diagram* of an action signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$ consists of a set S of *states*, a function $V : \mathbf{F} \times S \rightarrow \{f, t\}$, and a subset $R \subseteq S \times 2^{\mathbf{A}} \times S$ of *transitions*. We say that $V(P, s)$ is the *value* of P in s . The states s' such that $\langle s, A, s' \rangle \in R$ are the possible *results of the execution* of the action A in the state s . We say that A is *executable* in s , if at least one such state s' exists.

A transition diagram can be thought of as a labeled directed graph. Every state s is represented by a vertex labeled with the function $P \mapsto V(P, s)$ from fluent names to truth values. Every triple $\langle s, A, s' \rangle \in R$ is represented by an edge leading from s to s' and labeled A . An example of a transition diagram is shown in Figure 1.

Action description languages. We consider a subset of the action description language $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998] that consists of two kinds of expressions (called *causal laws*): *static laws* of the form

$$\text{caused } L \text{ if } G, \quad (3)$$

where L is a literal (an expression of the form P or $\neg P$, where P is a fluent name) and G is a propositional combination of fluent names, and *dynamic laws* of the form

$$\text{caused } L \text{ if } G \text{ after } U, \quad (4)$$

caused $PowerON$ **after** $PushPB_{TV} \wedge \neg PowerON$
caused $\neg PowerON$ **after** $PushPB_{TV} \wedge PowerON$
caused $TvON$ **if** $PowerON$
caused $\neg TvON$ **if** $\neg PowerON$
inertial $PowerON, \neg PowerON$
inertial $TvON, \neg TvON$

Figure 2: An action description.

where L and G are as above, and U is a propositional combination of fluent names and action names. In (3) and (4) the part **if** G can be dropped if G is *True*.

An *action description* is a set of causal laws. For instance, the knowledge base about a TV system, D , of the agent in the previous section, can be described by causal laws in Figure 2. Thereby, an expression of the form:

$$\text{inertial } P, \neg P,$$

where P is a fluent name stands for the dynamic laws

$$\begin{aligned} &\text{caused } P \text{ if } P \text{ after } P, \\ &\text{caused } \neg P \text{ if } \neg P \text{ after } \neg P, \end{aligned}$$

describing that the value of the fluent P stays the same unless some action changes it.

The meaning of an action description can be represented by a transition diagram. Let D be an action description with a signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$. Then the transition diagram $\langle S, V, R \rangle$ described by D is defined as follows:

- (i) S is the set of all interpretations s of $\mathbf{F}$ such that, for every static law (3) in D , s satisfies $G \supset L$,
- (ii) $V(P, s) = s(P)$,
- (iii) R is the set of all triples $\langle s, A, s' \rangle$ such that s' is the only interpretation of $\mathbf{F}$ which satisfies the heads L of all
 - static laws (3) in D for which s' satisfies G , and
 - dynamic laws (4) in D for which s' satisfies G and $s \cup A$ satisfies U .

The laws included in (iii) above are those that are *applicable* to the transition from s to s' caused by executing A . For instance, the transition diagram described by the action description of Figure 2 is presented in Figure 1. We say that an action description is *consistent* if it can be represented by a transition diagram with nonempty state set.

In the following we suppose that an action description D consists of two parts: D_u (unmodifiable causal laws) and D_m (modifiable causal laws). Therefore, we sometimes denote an action description D by the pair (D_u, D_m) .

Action query languages. To talk about observations of the world, or assertions about the effects of the execution of actions, we use an action query language similar to $\mathcal{R}$ [Gelfond and Lifschitz, 1998]. We consider a language with *static queries* of one of the two forms:

$$\text{holds } F, \quad (5)$$

$$\text{always } F, \quad (6)$$

where F is a propositional combination of fluent names, and *dynamic queries* of the form:

$$\text{necessarily } F \text{ after } A_0; \dots; A_n, \quad (7)$$

where F is a propositional combination of fluent names, and each A_i ($0 \leq i \leq n$) is an action. A *query* is a propositional combination of both kinds of expressions.

As for the semantics, let $T = \langle S, V, R \rangle$ be the transition diagram representing an action description D . A *history* of T of length n is a sequence

$$s_0, A_1, s_1, \dots, s_{n-1}, A_n, s_n \quad (8)$$

where each $\langle s_i, A_{i+1}, s_{i+1} \rangle$ ($0 \leq i < n$) is a transition in R . The satisfaction relation between a state s in S and a query q relative to D (denoted $D, s \models q$) is defined as follows:

- (i) if q is a static query (5), it is *satisfied* at s if the interpretation $P \mapsto V(P, s)$ satisfies F ;
- (ii) if q is a static query (6), it is *satisfied* at s if, for each state $s' \in S$, the interpretation $P \mapsto V(P, s')$ satisfies F ;
- (iii) if q is a dynamic query (7), it is *satisfied* at s if, for every history $s = s_0, A_0, s_1, \dots, s_{n-1}, A_n, s_n$ of T of length $n + 1$, the interpretation $P \mapsto V(P, s_n)$ satisfies F ;
- (iv) for any other q , *satisfaction* is defined by the truth tables of propositional logic.

We remark that **holds** $F \equiv \neg$ **holds** $\neg F$, but not so for a static query of form (6), i.e., **always** $F \not\equiv \neg$ **always** $\neg F$.

For a set C of queries, we denote by S_C^D the set of all states $s \in S$ such that $D, s \models q$ for every $q \in C$, and by S_{-C}^D the set $S \setminus S_C^D$. We say that D *entails* C (denoted $D \models C$) if $S = S_C^D$. For instance, a set of queries containing the query

$$\text{necessarily } \neg TvON \text{ after } \{PushPB_{RC}\}$$

is not entailed by the action description in Figure 2, because the query above is not satisfied at the state where both fluents $TvON$ and $PowerON$ are mapped to t . On the other hand, this action description entails the queries:

$$\text{always } PowerON \equiv TvON, \quad (9)$$

$$\text{holds } PowerON \wedge TvON \supset \neg \text{necessarily } \neg TvON \text{ after } \{PushPB_{TV}\}. \quad (10)$$

Queries allow us to express various pieces of knowledge about the domain (possibly acquired from earlier observations), like the following.

- **Existence of certain states:** There exists a state of the world where a formula F holds (assuming consistency of D):
 $\neg \text{always } \neg F$.

- **Executability of actions:** An action sequence $A_0, \dots, A_n$ is executable at every state:

$$\neg \text{necessarily } False \text{ after } A_0; \dots; A_n.$$

As in (2), this query can be abbreviated as

$$\text{executable } A_0; \dots; A_n.$$

- **Mandatory effects of actions:** A sequence $A_0, \dots, A_n$ of actions will always have certain effects in a given context:

$$\text{holds } G \supset \text{necessarily } F \text{ after } A_0; \dots; A_n$$

Here F describes the effects and G the context. For instance, the query

$$\text{holds } TvON \supset \text{necessarily } \neg TvON \text{ after } \{PushPB_{RC}\}$$

expresses that the result of executing $\{PushPB_{RC}\}$ at any state where $TvON$ is mapped to t is a state where $TvON$ is mapped to f . Note that queries of this form allow us to express all possible executions of a conformant plan $\langle A_0, \dots, A_n \rangle$ to reach a goal specified by F from an initial state described by G .

- **Possible effects of actions:** A sequence $A_0, \dots, A_n$ of actions may have certain effects in a given context:

$$\text{holds } G \supset \neg \text{necessarily } \neg F \text{ after } A_0; \dots; A_n,$$

like in (10). Here F describes the effects and G the context. Queries of this form allow us to express, e.g., the observation that executing $\langle A_0, \dots, A_n \rangle$ in some initial state described by G leads to a state where F holds (e.g., as for (10), that the TV is on). Formula G may describe a collection of states.

3 Problem Description

We define an *Action Description Update (ADU)* problem by an action description $D = (D_u, D_m)$, a set Q of causal laws, and a set C of queries, all with the same signature. We say that an action description D' *accomplishes an (action description) update* of D by Q relative to C , if

- (i) D' is consistent,
- (ii) $Q \cup D_u \subseteq D' \subseteq D \cup Q$,
- (iii) $D' \models C$,
- (iv) there is no consistent action description D'' such that $D' \subset D'' \subseteq D \cup Q$, and $D'' \models C$.

Such a D' is called a *solution* to the ADU problem (D, Q, C) .

Condition (i) expresses that an action description update modeling a dynamic domain, such as the TV system in Section 1, must have a state. According to Condition (ii), new knowledge about the world and the invariable part of the existing action description are kept, and causal laws in the variable part are considered to be either “correct” or “wrong”, and in the latter case simply disposed.

Condition (iii) imposes semantical constraints C on D' , which comprise further knowledge about the action domain gained, e.g., from experience. It is important to note that C can be modified later for another action description update. If Q is a revision (containing more precise knowledge about the domain), then reasonably C should contain all conditions corresponding to observations made about the domain, while other conditions may be kept or dropped; on the other hand, if Q is a change of the world per se, then conditions corresponding to observations might be dropped.

Finally, Condition (iv) applies Occam’s Razor to select solutions as simple as possible. Simplicity is measured by a smallest set (in terms of inclusion) of causal laws which need to be dropped.

We call any D' which satisfies (i)-(iii) a *near-solution* of the ADU problem (D, Q, C) .

Example 1 For instance, take D the action description in Figure 2, with D_m containing the first four causal laws; consider Q to be the set containing the following two causal laws:

$$\begin{aligned} &\text{caused } TvON \text{ after } PushPB_{RC} \wedge PowerON \wedge \neg TvON, \\ &\text{caused } \neg TvON \text{ after } PushPB_{RC} \wedge TvON, \end{aligned}$$

and let C be the set containing queries (2), (10), and

$$\text{executable } \{PushPB_{TV}\}.$$

The transition diagram described by $D \cup Q$ is shown in Figure 3. Here we can see that at the state where both $PowerON$ and $TvON$ are mapped to t , the action $PushPB_{RC}$ is not executable. Therefore, $D \cup Q$ is not a solution. In fact, a solution to the ADU problem (D, Q, C) is obtained by dropping the static law (1) from $D \cup Q$. $\square$

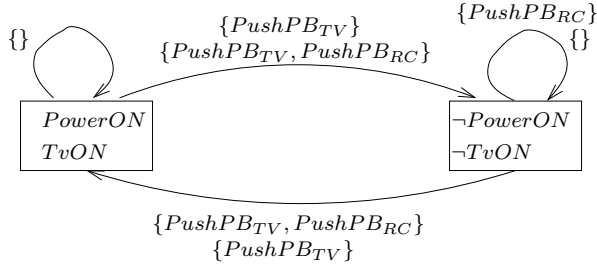


Figure 3: Transition diagram described by $D \cup Q$ of Ex. 1.

4 Properties of Updates

We first consider some basic properties of solutions to an ADU problem. To this end, we define the subsumption relation between an action description D and a causal law, both with the same signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$, as follows:

- A static law (3) is *subsumed* by D if, for every state s in the transition diagram described by D , the interpretation of $\mathbf{F}$ describing the state s satisfies $G \supset L$;
- a dynamic law (4) is *subsumed* by D if, for every transition $\langle s, A, s' \rangle$ in the transition diagram described by D , the following holds: if the interpretation of $\mathbf{F} \cup \mathbf{A}$ describing the state s and the action A satisfies U , then the interpretation of $\mathbf{F}$ describing s' satisfies $G \supset L$.

A set Q of causal laws is *subsumed* by an action description D , if every law in Q is subsumed by D . A causal law is *tautological*, if it is subsumed by every action description D .

Proposition 1 (Subsumption) *Let (D, Q, C) be an ADU problem, such that D is consistent and $D \models C$. If D subsumes Q then $D \cup Q$ is a solution to (D, Q, C) .*

From this result, we obtain the following corollaries.

Corollary 1 (Void Update) *Let $(D, \emptyset, C)$ be an ADU problem. If D is consistent and $D \models C$, then D is the unique solution to $(D, \emptyset, C)$.*

Corollary 2 (Idempotence) *Let (D, D, C) be an ADU problem, such that D is consistent and $D \models C$, then D is a solution to (D, D, C) .*

Corollary 3 (Addition of Tautologies) *Let (D, Q, C) be an ADU problem, such that D is consistent and $D \models C$. If Q is tautological, i.e., every causal law of Q is a tautology, then $D \cup Q$ is a solution to (D, Q, C) .*

Notice that a similar property fails for logic programming updates as in [Alferes et al., 2000; Eiter et al., 2002].

We next consider a structural property of solutions and near-solutions, which can be exploited for a syntactical decomposition of an ADU problem, in a divide-and-conquer manner. Because of the involved semantics of transitions and causation, in general some prerequisites are needed. For any action signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$, we denote by $\mathcal{L}_X$ the part of it which appears in X . We say that an action description D *permits NOP*, if there is some transition $\langle s, \emptyset, s' \rangle$ in the transition diagram described by D . A query F occurs *positively* (resp. *negatively*) in a set C of queries if F occurs in the scope of an even (resp. odd) number of negations, $\neg$, in a query in C .

Theorem 1 (Disjointness) *Let (D, Q, C) be an ADU problem with signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$. Let $\{D^0, D^1\}$, $\{Q^0, Q^1\}$,*

and $\{C^0, C^1\}$ be partitionings of D , Q , and C , respectively, such that (i) $\mathcal{L}_{D^0} \cap \mathcal{L}_{D^1} = \emptyset$, (ii) $\mathcal{L}_{Q^0} \subseteq \mathcal{L}_{D^0} \wedge \mathcal{L}_{Q^1} \subseteq \mathcal{L}_{D^1}$, and (iii) $\mathcal{L}_{C^0} \subseteq \mathcal{L}_{D^0} \wedge \mathcal{L}_{C^1} \subseteq \mathcal{L}_{D^1}$.

- Let X^i be a near-solution to (D^i, Q^i, C^i) , for $i = 0, 1$. Suppose that, for $i = 0, 1$, X^i permits NOP if some dynamic query occurs negatively in C^{1-i} . Then $X^0 \cup X^1$ is a near-solution to (D, Q, C) .
- Let X be a near-solution to (D, Q, C) , and let $\{X^0, X^1\}$ be a partitioning of X such that $X^0 \subseteq D^0$ and $X^1 \subseteq D^1$. Suppose that, for $i = 0, 1$, X^i permits NOP if some dynamic query occurs positively in C^{1-i} . Then, for $i = 0, 1$, X^i is a near-solution to (D^i, Q^i, C^i) .

Informally, the permission of NOP in Theorem 1 is needed to ensure that the transition diagrams of near-solutions to the sub-problems can be “combined”. Here are two useful consequences of Proposition 1:

Corollary 4 *Let (D, Q, C) be an ADU problem, and let $\{D^0, D^1\}$, $\{Q^0, Q^1\}$, $\{C^0, C^1\}$ be partitionings of D , Q , C , respectively, satisfying Conditions (i)–(iii) of Theorem 1.*

- Suppose that no dynamic query occurs in C . If, for $i = 0, 1$, X^i is a solution to (D^i, Q^i, C^i) , then $X^0 \cup X^1$ is a solution to (D, Q, C) . Furthermore, every solution to (D, Q, C) can be represented in this form.
- If no dynamic query occurs in C^1 , and X^1 is a solution to (D^1, Q^1, C^1) permitting NOP, then, for every solution X^0 to (D^0, Q^0, C^0) , $X^0 \cup X^1$ is a solution to (D, Q, C) .

Example 2 Consider the ADU problem $(D \cup D', Q, C)$, with D , Q , and C as in Example 1, and D' consisting of the causal laws:

inertial $LightON, \neg LightON$, (11)

caused $LightON$ after $SwitchLight \wedge \neg LightON$, (12)

caused $\neg LightON$ after $SwitchLight \wedge LightON$. (13)

Since $X^0 = D \cup Q \setminus \{(1)\}$ is a solution to (D, Q, C) (Example 1), $X^1 = D'$ is a solution to $(D', \emptyset, \emptyset)$, and D' permits NOP, by Corollary 4 (ii) $X^0 \cup X^1 = (D \cup D' \cup Q) \setminus \{(1)\}$ is a solution to $(D \cup D', Q, C)$. $\square$

5 Computational Complexity

An important subtask in solving ADU problems is checking whether a set of queries is entailed by an action description.

Theorem 2 *Given an action description D and a set C of queries, deciding $D \models C$ is (i) Π_2^P -complete in general, and (ii) $P_{||}^{NP}$ -complete if C does not involve dynamic queries.*

Here $P_{||}^{NP}$ means polynomial-time with a single parallel evaluation of calls to an NP oracle.

Proof. (Sketch) W.l.o.g., C contains a single query c . As for the membership part of (ii), deciding whether $D \models a$ for a static query a of form (5) is polynomial and of form (6) is in coNP. Hence, for a Boolean combination c of such queries $a_1, \dots, a_n$, deciding whether $D, s \models c$ is feasible in polynomial time with a single parallel evaluation of NP oracles. For (i), deciding whether $D, s \models d$ for a given state s and a dynamic query d of form (7) is in NP. Thus $D, s \models d$ for a general query d is decidable in polynomial time with an NP oracle, which means deciding $D \models d$ is in $NP^{NP} = \Sigma_2^P$.

Hence, deciding $D \models d$ is in $\Pi_2^P = \text{co-}\Sigma_2^P$. The hardness results follow from transformations from deciding whether (i) a Boolean combination of Quantified Boolean Formulas (QBFs) of the form $\exists X E_j$, $1 \leq j \leq m$, is true and (ii) a QBF of the form $\forall X \exists Y E$ is true, respectively. $\square$

As for finding solutions, we note that deciding whether $D \cup Q$ is a solution to an ADU problem (D, Q, C) has the same complexity as deciding $D \models C$ in general. Deciding the existence of an arbitrary solution is harder.

Theorem 3 *Deciding if a given ADU problem (D, Q, C) has a solution (or a near-solution) is (i) Σ_3^P -complete in general, (ii) Σ_2^P -complete if C does not involve dynamic queries, and (iii) NP-complete if $C = \emptyset$.*

Proof. (Sketch) Let $D = D_u \cup D_m$. We can guess a near-solution D' such that $D_u \cup Q \subseteq D' \subseteq D_u \cup Q$, along with a state s for D' (to witness consistency), and check $D \models C$ for (i) and (ii) in polynomial time with the help of (i) a Σ_2^P -oracle and (ii) an NP-oracle respectively. This proves membership in (i) Σ_3^P , (ii) Σ_2^P , and (iii) NP, respectively. The hardness results follow from transformations from deciding whether (i) a QBF of the form $\exists X F$ is true, (ii) a QBF of form $\exists Z \forall X F$ is true, and (iii) a QBF of the form $\exists Z \forall X \exists Y F$ is true. $\square$

Similarly, testing arbitrary solution candidates has higher complexity than testing $D \cup Q$.

Theorem 4 *Given an ADU problem (D, Q, C) and an action description D' , deciding whether D' is a solution for it is (i) Π_3^P -complete in general, (ii) Π_2^P -complete if C does not involve dynamic queries, and (iii) coNP-complete if $C = \emptyset$.*

Hence, even recognizing solutions is quite hard. However, recognizing near-solutions is easier (Π_2^P -complete in general, P^{NP} -complete if C has no dynamic queries, and polynomial if $C = \emptyset$). This follows easily from Theorem 2.

6 Computing Solutions

With an oracle for near-solutions, we can incrementally compute a solution to an ADU problem (D, Q, C) where $D = (D_u, D_m)$, in polynomial time as follows:

1. If $((D_u, D_m), Q, C)$ has a near-solution then $D' := \emptyset$ else halt;
2. For each $q_i \in D_m = \{q_1, \dots, q_m\}$ do
 $D_u^i = D_u \cup D' \cup \{q_i\}$; $D_m^i = \{q_{i+1}, \dots, q_m\}$;
 if $((D_u^i, D_m^i), Q, C)$ has a near-solution then
 $D' := D' \cup \{q_i\}$;
3. Output $D_u \cup D'$. $\square$

By virtue of Theorems 3 and 4, no substantially better algorithm of this type is likely to exist, since computing a solution is both Σ_3^P - and Π_3^P -hard, and thus needs the power of a Σ_3^P oracle. If a near-solution D_n for (D, Q, C) is known, we may compute, starting from it, a solution in the manner above (assuming $D_n = \{q_1, \dots, q_k\}$, set $D' = D_n$ and $i = k + 1$).

Near-solutions to a given ADU problem may be nondeterministically computed as in the membership part of Theorem 3, or obtained from a QBF encoding using a QBF solver. We present here a different computation method, using “update descriptions” and “update fluent sets.”

Let $D = (D_u, D_m)$ be an action description with signature $\langle \mathbf{F}, \mathbf{A} \rangle$. The *update description* of D is the action description obtained from D as follows:

1. Extend $\langle \mathbf{F}, \mathbf{A} \rangle$ by a set $\mathbf{H}$ of $k = |D_m|$ new fluents (called *update fluents*) $H_1, \dots, H_k$;
2. label each static law (3) in D_m with a fluent $H_i \in \mathbf{H}$:

$$\text{caused } L \text{ if } G \wedge H_i, \quad (14)$$

and each dynamic law (4) in D_m with a fluent $H_i \in \mathbf{H}$:

$$\text{caused } L \text{ if } G \text{ after } U \wedge H_i, \quad (15)$$

such that no two laws are labeled by the same fluent H_i ;

3. for each H_i labeling a law, add the dynamic law:

$$\text{inertial } H_i, \neg H_i. \quad (16)$$

Let $D = (D_u, D_m)$ be an action description and C be a set of queries with the same signature. Let E be the update description of D , with a set $\mathbf{H}$ of update fluents. Let $\langle S^E, V^E, R^E \rangle$ be the transition diagram described by E . An *update (fluent) set* for E relative to C is a subset $\mathbf{M}$ of $\mathbf{H}$ such that

- (i) $\mathbf{M}$ is a hitting set for S_{-C}^E (i.e., it contains for each state $s \in S_{-C}^E$ some fluent which is true at s);
- (ii) $\mathbf{H} \setminus \mathbf{M}$ is contained in some state in S_C^E .

With the notions above, we can find a near-solution to an ADU problem (D, Q, C) , where $D = (D_u, D_m)$, as follows:

1. If $D \cup Q$ is consistent and $D \cup Q \models C$, output $D \cup Q$.
2. Otherwise, construct the update description E of $D \cup Q = (D_u \cup Q, D_m)$.
3. Take a minimal update set $\mathbf{M}$ for E relative to C , if one exists; otherwise, take $\mathbf{M} = \mathbf{H}$.
4. Identify the set W of causal laws in D_m labeled by the elements of $\mathbf{M}$.
5. Output $D_u \cup (D_m \setminus W) \cup Q$.

Proposition 2 *Let (D, Q, C) be an ADU problem, with $D = (D_u, D_m)$. If there is a nonempty update set for the update description of $D \cup Q = (D_u \cup Q, D_m)$ relative to C , then the method above generates a near-solution to (D, Q, C) .*

Example 3 Consider the ADU problem (D, Q, C) presented in Example 1. Note that $D \cup Q \not\models C$ (as explained in Example 1). We obtain the following update description E of $(D_u \cup Q, D_m)$, which contains $D_u \cup Q$ and the laws:

$$\begin{aligned} &\text{caused } \text{PowerON} \text{ after } \text{PushPB}_{TV} \wedge \neg \text{PowerON}, H_1, \\ &\text{caused } \neg \text{PowerON} \text{ after } \text{PushPB}_{TV} \wedge \text{PowerON}, H_2, \\ &\text{caused } \text{TvON} \text{ if } \text{PowerON}, H_3, \\ &\text{caused } \neg \text{TvON} \text{ if } \neg \text{PowerON}, H_4, \\ &\text{inertial } H_i, \neg H_i \quad (1 \leq i \leq 4). \end{aligned}$$

According to the transition diagram described by E ,

$$\begin{aligned} S_{-C}^E &= \{\text{PowerON}, \text{TvON}, H_3\} \times 2^{\mathbf{H}}, \\ S_C^E &= \{\{H_1, H_2, H_4\}, \dots\}. \end{aligned}$$

The only minimal update set for E relative to C is $\{H_3\}$. We thus remove from $D \cup Q$ the law labeled by H_3 , i.e., (1). The result is a near-solution and also a solution to the problem. $\square$

Example 4 Take D to be the action description of Example 1, with the following causal laws added to D_m :

$$\begin{aligned} &\text{caused } \text{TvON} \text{ after } \text{PushPB}_{TV} \wedge \neg \text{PowerON}, \\ &\text{caused } \neg \text{TvON} \text{ after } \text{PushPB}_{TV} \wedge \text{PowerON}. \end{aligned}$$

Take Q and C as in Example 1. The transition diagram described by $D \cup Q$ is shown in Figure 3. Due to the same reasons mentioned in Example 1, $D \cup Q \not\models C$. The update description E of $(D_u \cup Q, D_m)$ consists of the update description presented in Example 3 and the causal laws:

caused $TvON$ **after** $PushPB_{TV} \wedge \neg PowerON, H_5$,
caused $\neg TvON$ **after** $PushPB_{TV} \wedge PowerON, H_6$,
inertial $H_i, \neg H_i$ ($5 \leq i \leq 6$).

According to the transition diagram described by E ,

$$S_{-C}^E = \{PowerON, TvON, H_3\} \times 2^{H \setminus \{H_1\}} \cup \{H_4, H_5\} \times 2^{H \setminus \{H_1\}},$$

$$S_C^E = \{\{H_1, H_2, H_4, H_5, H_6\}, \dots\}.$$

The minimal update sets for E relative to C are $\{H_3, H_4\}$ and $\{H_3, H_5\}$. We may choose either one and, by removing from $D \cup Q$ the corresponding two causal laws, we get a near-solution to the problem. None of the near-solutions is however a solution, as removing (1) is sufficient. $\square$

7 Discussion

Related work. Despite conceptional similarities to model based diagnosis, for space reasons, we focus our discussion on more closely related approaches. Updating knowledge bases has been studied extensively in the context of both databases and AI, with different approaches, and in various representation frameworks, e.g., [Winslett, 1990; Katsuno and Mendelzon, 1991; Pereira *et al.*, 1991; Eiter *et al.*, 2002; Sakama and Inoue, 2003; Li and Pereira, 1996; Liberatore, 2000]. The last three are related more closely.

[Sakama and Inoue, 2003] is similar to our work in that it also studies update problems in a nonmonotonic framework (yet in logic programming) and considers the same criterion of minimal change. It deals with three kinds of updates to a knowledge base D : theory update of D by some new information Q , inconsistency removal from D , and view update of $D = (D_u, D_m)$ by some new information Q . In the context of reasoning about actions and change, these kinds of updates are expressible as ADU problems $(D, Q, \emptyset)$, $(D, \emptyset, \emptyset)$, and $((D_u, D_m \cup Q), \emptyset, \emptyset)$. Sakama and Inoue show that checking for solution existence is NP-hard for each problem; this complies with Theorem 3 (iii). An important difference to [Sakama and Inoue, 2003] is that in an ADU problem (D, Q, C) , conditions C may not be directly expressed in D . Moreover, the semantics of an action description D in $\mathcal{C}$ is a transition diagram, and only captured by *all* answer sets of a logic program corresponding to D by known transformations.

[Li and Pereira, 1996] and [Liberatore, 2000] study, like we do, theory update problems in the context of reasoning about actions and change, based on an action language (but language $\mathcal{A}$ instead of $\mathcal{C}$). New information, Q , contains facts describing observations over time (e.g., the action $PushPB_{RC}$ occurs at time stamp 0). The action language $\mathcal{C}$ we use is more expressive than $\mathcal{A}$ in that it accommodates nondeterminism and concurrency, and the changes in the world are not only due to direct effects of actions. To formulate temporal observations, we can extend our query language by queries of the forms

$$E \text{ occurs at } t_i, \quad (17)$$

$$P \text{ holds at } t_i, \quad (18)$$

where E is an action name, P is a fluent name, and t_i is a time stamp; a state s satisfies a query (17) resp. (18) if, for some history (8) such that $s = s_0$, E is in A_{i+1} resp. s_i satisfies P .

Our notion of consistency of an action description D (in essence, the existence of a state) is different from that of Zhang *et al.* [2002]. They describe action domains in propositional dynamic logic, and require for consistency the existence of some model of an action description. Different from the setting here, conflicting action effects may prevent any model. With the extension of our query language discussed above, other forms of consistency studied in [Zhang *et al.*, 2002] can be achieved in our framework, by describing possible scenarios or formulas as queries.

Repair of action descriptions. We can sometimes improve solutions (and near-solutions) to an ADU problem (D, Q, C) by considering a slightly different version of the problem. We may take the view that a causal law is not completely wrong, and for instance holds in certain contexts. Suppose that Q is a dynamic law of the form:

caused L' **after** $A' \wedge G'$,

where L' is a literal, G' is a propositional combination of fluents, and A' is an action. We can obtain an action description D^* from D , which describes the same transition diagram as D , by replacing each dynamic law (4) in D_m with:

caused L **if** F **after** $U \wedge G'$,
caused L **if** F **after** $U \wedge \neg G'$.

Here, we can observe that for each solution D' to (D, Q, C) some solution $D^{*'} to (D^*, Q, C) exists which contains D' . Therefore, such a method can be useful to prevent “complete removal” of some laws from the given action description. Similar methods are also useful for repairing an action description, e.g., if some dynamic laws (4) in the action description have missing formulas in U . The study of such techniques is a part of future work.$

References

- [Alferes *et al.*, 2000] J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. C. Przymusinski. Dynamic Updates of Non-Monotonic Knowledge Bases. *Journal of Logic Programming*, 45(1–3):43–70, 2000.
- [Eiter *et al.*, 2002] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. On properties of update sequences based on causal rejection. *TPLP*, 2(6):721–777, 2002.
- [Gelfond and Lifschitz, 1998] M. Gelfond and V. Lifschitz. Action languages. *ETAI*, 3:195–210, 1998.
- [Giunchiglia and Lifschitz, 1998] E. Giunchiglia and V. Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. AAAI*, pp. 623–630, 1998.
- [Katsuno and Mendelzon, 1991] H. Katsuno and A. O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proc. KR*, pp. 387–394, 1991.
- [Li and Pereira, 1996] R. Li and L. M. Pereira. What is believed is what is explained. In *Proc. AAAI*, pp. 550–555.
- [Liberatore, 2000] P. Liberatore. A framework for belief update. In *Proc. JELIA*, pp. 361–375, 2000.
- [Pereira *et al.*, 1991] L. M. Pereira, J. J. Alferes, and J. N. Aparicio. Contradiction removal within well-founded semantics. In *Proc. LPNMR*, pp. 105–119, 1991.
- [Sakama and Inoue, 2003] C. Sakama and K. Inoue. An abductive framework for computing knowledge base updates. *TPLP*, 3(6):671–713, 2003.
- [Winslett, 1990] M. Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.
- [Zhang *et al.*, 2002] D. Zhang, S. Chopra, and N. Foo. Consistency of action descriptions. In *Proc. PRICAI*, 2002.

Quota and Gmin Merging Operators

Patricia Everaere Sébastien Konieczny Pierre Marquis

CRIL - Université d'Artois

62307 Lens, France

{everaere, konieczny, marquis}@cril.univ-artois.fr

Abstract

In this paper, two families of merging operators are considered: quota operators and *Gmin* operators. Quota operators rely on a simple idea: any possible world is viewed as a model of the result of the merging when it satisfies “sufficiently many” bases from the given profile (a multi-set of bases). Different interpretations of the “sufficiently many” give rise to specific operators. Each *Gmin* operator is parameterized by a pseudo-distance and each of them is intended to refine the quota operators (i.e., to preserve more information). Quota and *Gmin* operators are evaluated and compared along four dimensions: rationality, computational complexity, strategy-proofness, and discriminating power. Those two families are shown as interesting alternatives to the formula-based merging operators (which selects some formulas in the union of the bases).

1 Introduction

Merging operators aim at defining the beliefs (resp. goals) of a group of agents from their individual beliefs (resp. goals). The merging problem in the propositional setting has been addressed in many works, both from the artificial intelligence community and the database community [Cholvy, 1993; Revesz, 1997; Liberatore and Schaerf, 1998; Lin and Mendelzon, 1999; Baral *et al.*, 1991; 1992; Konieczny and Pino Pérez, 2002a]. It is also close to important issues considered in Social Choice Theory [Arrow, 1963; Moulin, 1988; Arrow *et al.*, 2002].

Each operator is more or less suited to the many merging scenarios which can be considered. Subsequently, when facing an application for which merging is required, a first difficulty is the choice of a specific merging operator. Among the criteria which can be used to make a clever choice, are the following ones:

Rationality: A main requirement for adhering to a merging method is that it offers the expected properties of what intuitively “merging” means. This calls for sets of rationality postulates and this has been addressed in several papers [Revesz, 1997; Liberatore and Schaerf, 1998; Konieczny and Pino Pérez, 2002a]. In the following, we

focus on the rationality postulates given in [Konieczny and Pino Pérez, 2002a], because they extend other proposals. The more (w.r.t. set inclusion) postulates satisfied the more rational the operator.

Computational Complexity: When one looks for a merging operator for an autonomous multi-agent system, a natural requirement is computational efficiency. In the worst case, merging is not a computationally easy task, and query answering typically lies at the first or even the second level of the polynomial hierarchy. Computationally easier operators can be obviously preferred to more complex ones.

Strategy-proofness: It is usually expected for merging that agents report truthfully their beliefs/goals. For many applications, this assumption can easily be made, in particular when the agents have limited reasoning abilities. However, when rational agents with full inference power are considered, such an assumption must be questioned: agents can be tempted to misreport their beliefs/goals in order to achieve a better merging result from their point of view. Strategy-proof operators must be preferred in such a case.

Discriminating Power: Because information is typically hard to be acquired, another important criterion to compare merging operators is cautiousness: merging operators which preserve only few information from the individual bases cannot be considered as valuable ones. Thus, it is natural to prefer operators leading to consistent merged bases that are as strong as possible from an inferential point of view.

As to rationality, one can look at [Revesz, 1997; Liberatore and Schaerf, 1998; Lin and Mendelzon, 1999; Konieczny, 2000; Konieczny and Pino Pérez, 2002a; Konieczny *et al.*, 2002]. As to computational complexity, see [Konieczny *et al.*, 2002], and for a study of strategy-proofness of many merging operators, a recent reference is [Everaere *et al.*, 2004] (see also [Meyer *et al.*, 2001] for a related study concerning OCF merging operators). In light of those results, it appears that no merging operator is a better performer than any other operator with respect to the four criteria. To be more precise model-based operators¹ are often computationally easier (inference

¹A distinction between model-based operators, which select some interpretations that are the “closest” to the bases, and formula-

is typically Θ_2^p -complete or Δ_2^p -complete) than formula-based ones (inference can be Π_2^p -complete) [Konieczny *et al.*, 2002]. Model-based operators also typically satisfy more rationality postulates (see [Konieczny and Pino Pérez, 2002a; Konieczny, 2000]). The third criterion is much more difficult to satisfy for both families of operators, even in very restricted cases. Actually, most of the merging operators identified so far in the literature are not strategy-proof. Since the four evaluation criteria appear hard to be satisfied altogether, one cannot do better than searching for good trade-offs.

We consider in this paper two families of propositional merging operators. The first one consists of quota merging operators. They rely on a simple idea: any possible world is viewed as a model of the result of the merging when it satisfies “sufficiently many” bases from the given profile. “Sufficiently many” can mean either “at least k ” (any integer, absolute quota), or “at least $k\%$ ” (a relative quota), or finally “as many as possible”, and each interpretation gives rise to a specific merging operator. We show that those operators exhibit good logical properties, have low computational complexity and are strategy-proof. Since this is achieved at the price of a potential lack of discriminating power, we introduce a second family of merging operators: *Gmin* operators. Each *Gmin* operator is parameterized by a pseudo-distance and each of them is intended to refine the quota operators (i.e., to preserve more information). Such operators are both more rational and more discriminating than quota merging operators. Unfortunately, this improvement has to be paid by a higher computational complexity, and more strategic vulnerabilities, but we think they offer an interesting compromise nevertheless.

Note that aggregation functions close to the ones on which quota and *Gmin* operators are based are used to deal with relational structures that are more complex than bipartitions of worlds (which are the structures under consideration in standard propositional logic). For instance, they have been considered in the possibilistic logic setting and for constraint satisfaction problems (see e.g. [Dubois *et al.*, 1999; 1998; Fargier, 1994]). However, as far as we know, no systematic study of quota and *Gmin* operators has been conducted so far in the standard propositional setting. Especially, they have never been evaluated with respect to the four criteria we consider. This is where the main contribution of the paper lies.

The rest of the paper is as follows. The next section gives some notations and definitions. In Section 3, quota operators are defined and their properties are studied. In Section 4, we define $\Delta^{k_{\max}}$, which is the operator obtained when optimizing the value of the quota under the constraint that it does not lead to an inconsistent merged base. In Section 5, Δ^{GMIN} operators are defined and their properties are studied. Finally, we conclude this paper in Section 6.

2 Formal Preliminaries

We consider a propositional language $\mathcal{L}$ defined from a finite set of propositional variables $\mathcal{P}$ and the usual connectives.

An interpretation (or world) is a total function from $\mathcal{P}$ to $\{0, 1\}$, denoted by a bit vector whenever a strict total order on $\mathcal{P}$ is specified. The set of all interpretations is noted $\mathcal{W}$. An

based ones, which pick some formulas in the union of the bases is often made [Konieczny *et al.*, 2002].

interpretation ω is a model of a formula $\phi \in \mathcal{L}$ if and only if it makes it true in the usual truth functional way. $[\phi]$ denotes the set of models of formula ϕ , i.e., $[\phi] = \{\omega \in \mathcal{W} \mid \omega \models \phi\}$.

A base K denotes the set of beliefs/goals of an agent, it is a finite and consistent set of propositional formulas, interpreted conjunctively. Unless stated otherwise, we identify K with the conjunction of its elements.

A profile E denotes the group of agents that is involved in the merging process. It is a multi-set (bag) of belief/goal bases $E = \{K_1, \dots, K_n\}$ (hence two agents are allowed to exhibit identical bases). We denote by $\bigwedge E$ the conjunction of bases of E , i.e., $\bigwedge E = K_1 \wedge \dots \wedge K_n$, and $\bigvee E$ is the disjunction of the bases of E , i.e., $\bigvee E = K_1 \vee \dots \vee K_n$. A profile E is said to be consistent if and only if $\bigwedge E$ is consistent. The multi-set union is noted $\sqcup$, multi-set containment relation is noted $\sqsubseteq$. The cardinal of a finite set (or a finite multi-set) A is noted $\#(A)$. We say that two profiles are equivalent, noted $E_1 \equiv E_2$, if there exists a bijection f from E_1 to E_2 such that for every $\phi \in E_1$, ϕ and $f(\phi)$ are logically equivalent.

The result of the merging of the bases of a profile E , under the integrity constraints μ , is the *merged base* denoted $\Delta_\mu(E)$. The *integrity constraints* consist of a consistent formula the merged base has to satisfy (it may represent some physical laws, some norms, etc.).

3 Quota Operators

Let us first define the quota operators.

Definition 1 Let k be an integer, $E = \{K_1, \dots, K_n\}$ be a profile, and μ be a formula. The k -quota merging operator, denoted Δ^k , is defined in a model-theoretic way as:

$$[\Delta_\mu^k(E)] = \begin{cases} \{\omega \in [\mu] \mid \forall K_i \in E \ \omega \models K_i\} & \text{if non empty,} \\ \{\omega \in [\mu] \mid \#(\{K_i \in E \mid \omega \models K_i\}) \geq k\} & \text{otherwise.} \end{cases}$$

Essentially, this definition states that the models of the result of the k -quota merging of profile E under constraints μ are the models of μ which satisfy at least k bases of E . When there is no conflict for the merging, i.e., $\bigwedge E \wedge \mu$ is consistent, the result of the merging is simply the conjunction of the bases.

Example 1 Let us consider the following example, with a profile $E = \{K_1, K_2, K_3, K_4\}$, such that $[K_1] = \{100, 001, 101\}$, $[K_2] = \{001, 101\}$, $[K_3] = \{100, 000\}$, $[K_4] = \{111\}$, and the integrity constraints $[\mu] = \mathcal{W} \setminus \{010, 011\}$. With quota operators we get as a result $[\Delta_\mu^1(E)] = \{000, 001, 100, 101, 111\}$, $[\Delta_\mu^2(E)] = \{001, 100, 101\}$ and $[\Delta_\mu^3(E)] = \emptyset$.

Here is an equivalent syntactical characterization of $[\Delta_\mu^k(E)]$ that is obtained from preferred consistent subsets of E .² Let us first define the following notation:

$$\lceil n_k \rceil = \{C \subseteq \{1, \dots, n\} \mid \#(C) = k\}.$$

Then the following proposition gives a characterization of quota operators :

²To be more precise, “subsets” stands here for multi-set containment.

Proposition 1 Let k be an integer, $E = \{K_1, \dots, K_n\}$ be a profile, and μ be a formula.

$$\Delta_\mu^k(E) \equiv \begin{cases} \bigwedge E \wedge \mu & \text{if consistent,} \\ \left(\bigvee_{C \in \Gamma_{n_k}^\top} \left(\bigwedge_{j \in C} K_j \right) \right) \wedge \mu & \text{otherwise.} \end{cases}$$

Interestingly, the size of the formula equivalent to $[\Delta_\mu^k(E)]$ given by Proposition 1 is polynomial in $|E| + |\mu|$. Hence, merged bases can be easily compiled as propositional formulas.

3.1 Logical Properties

Since we aim at investigating the logical properties of our family of merging operators, a set of properties must first be considered as a base line. The following set of postulates was pointed out in [Konieczny and Pino Pérez, 1999; 2002a]:

Definition 2 (IC merging operators) Δ is an IC merging operator if and only if it satisfies the following postulates:

- (IC0) $\Delta_\mu(E) \models \mu$
- (IC1) If μ is consistent, then $\Delta_\mu(E)$ is consistent
- (IC2) If $\bigwedge E$ is consistent with μ , then $\Delta_\mu(E) \equiv \bigwedge E \wedge \mu$
- (IC3) If $E_1 \equiv E_2$ and $\mu_1 \equiv \mu_2$, then $\Delta_{\mu_1}(E_1) \equiv \Delta_{\mu_2}(E_2)$
- (IC4) If $K_1 \models \mu$ and $K_2 \models \mu$, then $\Delta_\mu(\{K_1, K_2\}) \wedge K_1$ is consistent if and only if $\Delta_\mu(\{K_1, K_2\}) \wedge K_2$ is consistent
- (IC5) $\Delta_\mu(E_1) \wedge \Delta_\mu(E_2) \models \Delta_\mu(E_1 \sqcup E_2)$
- (IC6) If $\Delta_\mu(E_1) \wedge \Delta_\mu(E_2)$ is consistent, then $\Delta_\mu(E_1 \sqcup E_2) \models \Delta_\mu(E_1) \wedge \Delta_\mu(E_2)$
- (IC7) $\Delta_{\mu_1}(E) \wedge \mu_2 \models \Delta_{\mu_1 \wedge \mu_2}(E)$
- (IC8) If $\Delta_{\mu_1}(E) \wedge \mu_2$ is consistent, then $\Delta_{\mu_1 \wedge \mu_2}(E) \models \Delta_{\mu_1}(E)$

An IC merging operator is said to be an IC majority operator if it satisfies (Maj)

$$(Maj) \quad \exists n \quad \Delta_\mu(E_1 \sqcup \underbrace{E_2 \sqcup \dots \sqcup E_2}_n) \models \Delta_\mu(E_2)$$

Quota merging operators exhibit good logical properties.

Proposition 2 Δ^k operators satisfy properties (IC0), (IC2), (IC3), (IC4), (IC5), (IC7) and (IC8). They do not satisfy (IC1), (IC6) and (Maj) in general.

Only two properties of IC merging operators are not satisfied: (IC1)³ since the result of the quota merging can be inconsistent (see Example 1), and (IC6).

Beside those general properties, some specific additional properties, are satisfied by quota operators.

(Disj) If $(\bigvee E) \wedge \mu$ is consistent, then $\Delta_\mu(E) \models (\bigvee E) \wedge \mu$

³It is possible to make (IC1) satisfied by requiring that, when no interpretation reaches the quota (i.e., satisfies at least k bases), the merged base is equivalent to the integrity constraints formula. But this definition leads to operators which do not satisfy (IC5). This last property is very important from an aggregation point of view. It corresponds to a Pareto condition, that is considered as a minimal rationality requirement for aggregation in Social Choice Theory [Arrow, 1963; Moulin, 1988; Arrow et al., 2002]. This is why we do not consider such an additional family of operators any longer in the following.

Interestingly, the disjunction property (Disj) is not shared by every IC majority merging operator [Konieczny and Pino Pérez, 2002a], since most of them allow for “generating” some new beliefs/goals from the ones in the bases of the profile (some interpretations that do not satisfy any of the bases can be chosen as models of the merged base). When this behaviour is unexpected, formula-based merging operators – which satisfy (Disj) – can be used, but such operators do not satisfy many rationality postulates [Konieczny, 2000] (especially (IC3) is not satisfied) and are often hard from a computational point of view. Quota operators (as well as the other operators studied in this paper) which also ensure (Disj) offer interesting alternatives to formula-based operators in this respect.

Two other interesting postulates can be defined for characterizing more precisely quota operators; the first one is a weakening of (Maj), which is not satisfied by every IC merging operator:

(Wmaj) If $\Delta_\mu(E_2)$ is consistent, then $\exists n \quad \Delta_\mu(E_1 \sqcup \underbrace{E_2 \sqcup \dots \sqcup E_2}_n) \wedge \Delta_\mu(E_2)$ is consistent

The second one shows the prominence of the largest maximal consistent subsets of the profile; let us define $Maxcons_\mu(E)$ as $\{M \mid M \sqsubseteq E, \bigwedge M \wedge \mu \text{ is consistent, and } \forall M' M \sqsubset M' \sqsubseteq E, \bigwedge M' \wedge \mu \text{ is not consistent}\}$:

(Card) If $E_1, E_2 \in Maxcons_\mu(E)$, $\#(E_1) \leq \#(E_2)$, and $\Delta_\mu(E) \wedge E_1$ is consistent, then $\Delta_\mu(E) \wedge E_2$ is consistent

Proposition 3 Δ^k operators satisfy properties (Disj), (Card) and (Wmaj).

Note that it is not the case that every IC majority merging operator satisfies (Card) (see Section 5).

3.2 Computational Complexity

Let Δ be a merging operator. We consider the following decision problem $MERGE(\Delta)$:

- **Input :** a triple $\langle E, \mu, \alpha \rangle$ where $E = \{K_1, \dots, K_n\}$ is a profile, $\mu \in \mathcal{L}$ is a formula, and $\alpha \in \mathcal{L}$ is a formula.
- **Question :** Does $\Delta_\mu(E) \models \alpha$ hold?

For quota merging operators, we can prove that:

Proposition 4 $MERGE(\Delta^k)$ is $BH(2)$ -complete.

This $BH(2)$ -completeness result is obtained even in the restricted case the query is a propositional symbol and there is no integrity constraints ($\mu = \top$). Note that this complexity class is located at a low level of the boolean hierarchy. And that, obviously, the complexity of $MERGE(\Delta^k)$ decreases to $coNP$ whenever k is not lower than the number of bases of E (or under the restriction $\bigwedge E \wedge \mu$ is inconsistent).

3.3 Strategy-Proofness

Let us now investigate how robust quota operators are with respect to manipulation. Intuitively, a merging operator is strategy-proof if and only if, given the beliefs/goals of the other agents, reporting untruthful beliefs/goals does not enable an agent to improve her satisfaction. A formal counterpart of it is given in [Everaere et al., 2004]:

Definition 3 Let i be a satisfaction index, i.e., a total function from $\mathcal{L} \times \mathcal{L}$ to $\mathbb{R}$. A merging operator Δ is strategy-proof for i if and only if there is no integrity constraint μ , no profile $E = \{K_1, \dots, K_n\}$, no base K and no base K' such that $i(K, \Delta_\mu(E \sqcup \{K'\})) > i(K, \Delta_\mu(E \sqcup \{K\}))$.

Clearly, there are numerous different ways to define the satisfaction of an agent given a merged base. While many *ad hoc* definitions can be considered, the following three indexes from [Everaere *et al.*, 2004] are meaningful when no additional information are available:

Definition 4

- $i_{d_w}(K, K_\Delta) = \begin{cases} 1 & \text{if } K \wedge K_\Delta \text{ is consistent,} \\ 0 & \text{otherwise.} \end{cases}$
- $i_{d_s}(K, K_\Delta) = \begin{cases} 1 & \text{if } K_\Delta \models K, \\ 0 & \text{otherwise.} \end{cases}$
- $i_p(K, K_\Delta) = \begin{cases} \frac{\#([K] \cap [K_\Delta])}{\#[K_\Delta]} & \text{if } \#([K_\Delta]) \neq 0, \\ 0 & \text{otherwise.} \end{cases}$

For the weak drastic index (i_{d_w}), the agent is considered satisfied as soon as its beliefs/goals are consistent with the merged base. For the strong drastic index (i_{d_s}), in order to be satisfied, the agent must impose her beliefs/goals to the whole group. The last index (“probabilistic index” i_p) is not a boolean one, leading to a more gradual notion of satisfaction. The more compatible the merged base with the agent’s base the more satisfied the agent. The compatibility degree of K with K_Δ is the (normalized) number of models of K that are models of K_Δ as well.

Proposition 5 Quota merging operators are strategy-proof for i_p , i_{d_w} and i_{d_s} .

This is quite noticeable since strategy-proof merging operators are not numerous [Everaere *et al.*, 2004]. Strategy-proofness is hard to achieve, as illustrated in Social Choice Theory, for aggregation of preference relations, by the Gibbard-Satterthwaite impossibility theorem [Gibbard, 1973; Satterthwaite, 1975; Moulin, 1988].

3.4 Absolute and Relative Quotas

In the definition of quota merging operators, an absolute threshold, i.e., a fixed integer not depending on the number of bases in the profile, has been used. On the other hand, it can prove also sensible to express quota in a relative manner, and to define the models of the merged base as the interpretations satisfying at least half (or the two third, or the wanted ratio) of the initial bases. This technique is close to a well-known voting method used in Social Choice Theory, namely voting in committees [Barberà *et al.*, 1991]. Let us call such operators k -ratio merging operators (with $0 \leq k \leq 1$), and let us note them $\Delta^{\bar{k}}$.

Example 1 (continued) $[\Delta_{\mu}^{\bar{0.2}}(E)] = \{001, 100, 101\}$, $[\Delta_{\mu}^{\bar{0.3}}(E)] = [\Delta_{\mu}^{\bar{0.5}}(E)] = \{001, 100, 101\}$.

One can quickly figure out the close connections between the two families of quota merging operators (the one based on absolute quota and the other one on relative quota, or ratio). Each ratio merging operator corresponds to a family of quota merging operators (one for each possible cardinal of

the profile). And given a fixed cardinal, each (absolute) quota merging operator corresponds to a family of ratio merging operators.

Although the intuitive motivations of the two definitions of those families look different, it turns out that ratio merging operators have exactly the same properties w.r.t. computational complexity and strategy-proofness as (absolute) quota merging operators (though the proofs of some results are different). Only some logical properties are different.

Proposition 6 $\Delta^{\bar{k}}$ operators satisfy properties (IC0), (IC2), (IC3), (IC4), (IC5), (IC7), (IC8), (Maj), (Disj) and (Card). They do not satisfy (IC1) and (IC6) in general.

4 $\Delta^{k_{\max}}$ Operator

Now, whatever the chosen quota is absolute or not, an important point is the choice of its value. Let us first observe that quota merging operators lead to a sequence of merged bases that is monotonic w.r.t. logical entailment:

Proposition 7 Let E be a profile, μ be a formula. We have $\Delta_{\mu}^{k+1}(E) \models \Delta_{\mu}^k(E)$ for all integers k .

Among the elements of this sequence, some of them are of special interest. Thus, Δ^0 gives the conjunction of the bases (with the constraints) when consistent and μ otherwise. It is called *full meet merging operator* in [Konieczny and Pino Pérez, 1999]. Δ^1 gives the conjunction of the bases (with the constraints) when consistent and the disjunction of the bases (with the constraints) otherwise; it is closed to the *basic merging operator* [Konieczny and Pino Pérez, 1999], and is also definable as a model-based merging operator obtained using the drastic distance and max as aggregation function [Konieczny *et al.*, 2002]. The only difference is that Δ^1 gives an inconsistent result when the disjunction of the bases is not consistent with the constraints, whilst the basic merging operator gives μ in this case.

Each time k is increased, the result of the merging is either the same as for the previous value of k or is logically stronger. In our finite propositional framework, the sequence $(\Delta_{\mu}^k(E)) (k > 0)$ is obviously stationary from some stage. The value for which it becomes stationary is not interesting in itself, since the corresponding merged base is either equivalent to the conjunction of the bases of the profile (with the constraints), or to the inconsistent base. But an interesting value of k is the one leading to the last nontrivial merged base.

Definition 5 Let $E = \{K_1, \dots, K_n\}$ be a profile, μ be a formula. Let $k_{\max} = \max(\{i \leq \#(E) \mid \Delta_{\mu}^i(E) \not\models \perp\})$. $\Delta^{k_{\max}}$ is defined in a model-theoretic way as:

$$[\Delta_{\mu}^{k_{\max}}(E)] = \begin{cases} \{\omega \in [\mu] \mid \forall K_i \in E \omega \models K_i\} & \text{if non empty,} \\ \{\omega \in [\mu] \mid \#(\{K_i \in E \mid \omega \models K_i\}) = k_{\max}\} & \text{otherwise.} \end{cases}$$

While very close to quota operators, the resulting operator $\Delta^{k_{\max}}$ is not a true quota operator since the value of $k_{\max}$ is not given a priori, but depends on E and μ .

Example 1 (continued) $[\Delta_{\mu}^{k_{\max}}(E)] = \{001, 100, 101\}$.

At a first glance, $\Delta^{k_{\max}}$ looks similar to the formula-based operator Δ^{C^4} which selects cardinality-maximal subbases in

the union of the bases from the profile [Konieczny, 2000; Baral *et al.*, 1991; 1992]; however, $\Delta^{k_{\max}}$ and Δ^{C^4} are distinct; thus, while both operators satisfy **(Disj)**, $\Delta^{k_{\max}}$ satisfies **(IC3)** and **(Maj)** while Δ^{C^4} satisfies none of them. Contrastingly, $\Delta^{k_{\max}}$ belongs to two important families of model-based merging operators, namely the Δ^Σ family and the Δ^{GMax} family when the drastic distance is used [Konieczny and Pino Pérez, 2002b]. Accordingly, $\Delta^{k_{\max}}$ has very good logical properties:

Proposition 8 $\Delta^{k_{\max}}$ satisfies **(IC0 - IC8)**, **(Maj)**, **(Disj)** and **(Card)**.

$\Delta^{k_{\max}}$ is obtained by considering the problem of optimizing the quota (for “pure” quota operators, k is given, so it does not need to be computed). Unsurprisingly, the corresponding inference problem is computationally harder than the inference problem for quota operators (under the standard assumptions of complexity theory):

Proposition 9 $\text{MERGE}(\Delta^{k_{\max}})$ is Θ_2^P -complete.

Clearly enough, if $k_{\max}$ is computed during an off-line pre-processing stage and becomes part of the input afterwards, the complexity falls down to coNP.

Now, as to strategy-proofness, the $k_{\max}$ operator exhibits all the good properties of quota operators.

Proposition 10 $\Delta^{k_{\max}}$ is strategy-proof for the three indexes i_p , i_{d_w} and i_{d_s} .

5 Δ^{GMIN} Operators

Starting from $\Delta^{k_{\max}}$, one could wonder whether it is possible to constrain further the quota operators so as to get operators with a higher discriminating power, i.e., allowing more inferences to be drawn. In this section we provide a full family of such operators.

In order to define a Δ^{GMIN} operator, the definition of a pseudo-distance between interpretations is first needed:

Definition 6 A pseudo-distance between interpretations is a total function d from $\mathcal{W} \times \mathcal{W}$ to $\mathbb{N}$ such that for every $\omega_1, \omega_2 \in \mathcal{W}$

- $d(\omega_1, \omega_2) = d(\omega_2, \omega_1)$, and
- $d(\omega_1, \omega_2) = 0$ if and only if $\omega_1 = \omega_2$.

Any pseudo-distance between interpretations d induces a “distance” between an interpretation ω and a formula K given by $d(\omega, K) = \min_{\omega' \models K} d(\omega, \omega')$.

Examples of some such distances are the drastic distance, noted d_D , that gives 0 when $\omega_1 = \omega_2$ and 1 otherwise, or the Dalal distance [Dalal, 1988], noted d_H , that is the Hamming distance between interpretations.

Then $\Delta_\mu^{d, \text{GMIN}}$ operators are defined as:

Definition 7 Let d be a pseudo-distance, μ an integrity constraint, $E = \{K_1, \dots, K_n\}$ a profile and let ω be an interpretation. The “distance” $d_{d, \text{GMIN}}(\omega, E)$ is defined as the list of numbers $(d_1, \dots, d_n)$ obtained by sorting in increasing order the set $\{d(\omega, K_i) \mid K_i \in E\}$. The models of $\Delta_\mu^{d, \text{GMIN}}(E)$ are the models of μ that are minimal w.r.t. the lexicographic order induced by the natural order.

ω	K_1	K_2	K_3	K_4	$d_{d_H, \text{GMIN}}(\omega, E)$
000	1	1	0	3	(0,1,1,3)
001	0	0	1	2	(0,0,1,2)
100	0	1	0	2	(0,0,1,2)
101	0	0	1	1	(0,0,1,1)
110	1	2	1	1	(1,1,1,2)
111	1	1	2	0	(0,1,1,2)

Table 1: $\Delta^{d_H, \text{GMIN}}$ operator.

Example 1 (continued) $[\Delta_\mu^{d_D, \text{GMIN}}(E)] = \{001, 100, 101\}$. $[\Delta_\mu^{d_H, \text{GMIN}}(E)] = \{101\}$. The computations are reported in Table 1. Each column K_i gives the “distance” $d_H(\omega, K_i)$ between the models of the integrity constraints and K_i .

Clearly enough, $\Delta^{k_{\max}}$ is a specific $G_{\min}$ operator:

Proposition 11 $\Delta^{d_D, \text{GMIN}} = \Delta^{k_{\max}}$.

As far as discriminating power is concerned, Δ^{GMIN} operators are interesting operators, since they refine the operator $\Delta^{k_{\max}}$ (so they refine also every quota merging operator), as stated by the following property:

Proposition 12 For any pseudo-distance d , any integrity constraint μ and any profile E , we have

$$\Delta_\mu^{d, \text{GMIN}}(E) \models \Delta_\mu^{k_{\max}}(E)$$

Furthermore, $G_{\min}$ operators exhibit very good logical properties:

Proposition 13 Let d be any pseudo-distance. $\Delta^{d, \text{GMIN}}$ satisfies **(IC0 - IC8)**, **(Maj)** and **(Disj)**. It does not satisfy **(Card)** in general.

Thus, like formula-based merging operators, Δ^{GMIN} operators satisfy **(Disj)**, but contrariwise to formula-based merging operators, Δ^{GMIN} operators are IC merging operators.

Let us now investigate the strategy-proofness issue for the Δ^{GMIN} operators. In the general case, strategy-proofness of quota merging operators is lost:

Proposition 14 Let d be a pseudo-distance. $\Delta^{d, \text{GMIN}}$ is not strategy-proof for any index among the three indexes i_{d_w} , i_p and i_{d_s} .

We can guarantee strategy-proofness, but only in some very specific cases:

Proposition 15

- $\Delta^{d, \text{GMIN}}$ is strategy-proof for i_p , i_{d_w} and i_{d_s} if the bases are complete (i.e. each base has a unique model),
- $\Delta^{d, \text{GMIN}}$ is strategy-proof for the indexes i_{d_w} and i_{d_s} when $\#(E) = 2$ and $\mu = \top$.

Finally, let us turn to the computational complexity criterion.

Proposition 16 Assume that the pseudo-distance d of any pair of interpretations ω_1 and ω_2 can be computed in time polynomial in $|\omega_1| + |\omega_2|$. Then $\text{MERGE}(\Delta^{d, \text{GMIN}})$ is in Δ_2^P .

Proposition 17

- $\text{MERGE}(\Delta^{d_D, \text{GMIN}})$ is Θ_2^P -complete.

- $\text{MERGE}(\triangle^{d_H, \text{GMIN}})$ is Δ_2^p -complete.

As expected, the complexity of inference for $\triangle^{d, \text{GMIN}}$ operators is higher than the complexity of inference for quota operators (under the usual assumptions of complexity theory). However, it remains at the first level of the polynomial hierarchy under reasonable requirements on the pseudo-distance.

6 Conclusion

We have considered two families of merging operators, and investigated the properties of their operators with respect to four criteria: rationality, computational complexity, strategy-proofness and discriminating power. We claim that those four criteria are the main dimensions along with propositional merging operators have to be evaluated.

While no merging operators optimizing every criteria exist, we claim that both quota and *Gmin* operators are interesting trade-offs; even if they are not fully rational and discriminating, quota operators exhibit “low complexity” and are strategy-proof; on the other hand, *Gmin* operators are slightly more complex and not strategy-proof in the general case, but they are fully rational and much less cautious. They also lead to merged bases implying the disjunction of the bases from the considered profile, thus offering an interesting alternative to formula-based merging operators [Baral *et al.*, 1991; 1992; Konieczny, 2000; Konieczny *et al.*, 2002], which are typically at least as hard from the complexity point of view and satisfy less rationality postulates.

Acknowledgements

The authors would like to thank the anonymous reviewers. This work has been supported by the IUT de Lens, the Université d’Artois, the Région Nord/Pas-de-Calais, the IRCICA consortium, and by the European Community FEDER Program.

References

- [Arrow *et al.*, 2002] K.J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*, volume 1. North-Holland, 2002.
- [Arrow, 1963] K. J. Arrow. *Social Choice and Individual Values*. Wiley, New York, second edition, 1963.
- [Baral *et al.*, 1991] C. Baral, S. Kraus, and J. Minker. Combining multiple knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 3(2):208–220, 1991.
- [Baral *et al.*, 1992] C. Baral, S. Kraus, J. Minker, and V. S. Subrahmanian. Combining knowledge bases consisting of first-order theories. *Computational Intelligence*, 8(1):45–71, 1992.
- [Barberà *et al.*, 1991] S. Barberà, H. Sonnenschein, and L. Zhou. Voting by committees. *Econometrica*, 59(3):595–609, May 1991.
- [Cholvy, 1993] L. Cholvy. Proving theorems in a multi-sources environment. In *Proceedings of IJCAI’93*, pages 66–71, 1993.
- [Dalal, 1988] M. Dalal. Investigations into a theory of knowledge base revision: preliminary report. In *Proceedings of AAAI’88*, pages 475–479, 1988.
- [Dubois *et al.*, 1998] D. Dubois, H. Fargier, and H. Prade. Possibilistic likelihood relations. In *Proceedings of IPMU’98*, pages 1196–1203, 1998.
- [Dubois *et al.*, 1999] D. Dubois, H. Prade, and R. Yager. *Fuzzy Sets in Approximate Reasoning and Information Systems*, chapter Merging fuzzy information, pages 335–401. Handbook of Fuzzy Sets Series. Kluwer Academic, 1999.
- [Everaere *et al.*, 2004] P. Everaere, S. Konieczny, and P. Marquis. On merging strategy-proofness. In *Proceedings of KR’04*, pages 357–367, 2004.
- [Fargier, 1994] H. Fargier. *Problèmes de satisfaction de contraintes flexibles : application à l’ordonnancement de production*. PhD thesis, Université Paul Sabatier - Toulouse, 1994.
- [Gibbard, 1973] A. Gibbard. Manipulation of voting schemes. *Econometrica*, 41:587–602, 1973.
- [Konieczny and Pino Pérez, 1999] S. Konieczny and R. Pino Pérez. Merging with integrity constraints. In *Proceedings of ECSQARU’99*, LNAI 1638, pages 233–244, 1999.
- [Konieczny and Pino Pérez, 2002a] S. Konieczny and R. Pino Pérez. Merging information under constraints: a qualitative framework. *Journal of Logic and Computation*, 12(5):773–808, 2002.
- [Konieczny and Pino Pérez, 2002b] S. Konieczny and R. Pino Pérez. On the frontier between arbitration and majority. In *Proceedings of KR’02*, pages 109–118, 2002.
- [Konieczny *et al.*, 2002] S. Konieczny, J. Lang, and P. Marquis. Distance-based merging: a general framework and some complexity results. In *Proceedings of KR’02*, pages 97–108, 2002.
- [Konieczny, 2000] S. Konieczny. On the difference between merging knowledge bases and combining them. In *Proceedings of KR’00*, pages 135–144, 2000.
- [Liberatore and Schaerf, 1998] P. Liberatore and M. Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering*, 10(1):76–90, 1998.
- [Lin and Mendelzon, 1999] J. Lin and A. O. Mendelzon. Knowledge base merging by majority. In *Dynamic Worlds: From the Frame Problem to Knowledge Management*. Kluwer, 1999.
- [Meyer *et al.*, 2001] T. Meyer, A. Ghose, and S. Chopra. Social choice, merging and elections. In *Proceedings of ECSQARU’01*, pages 466–477, 2001.
- [Moulin, 1988] H. Moulin. *Axioms of Cooperative Decision Making*. Econometric society monographs. Cambridge University Press, 1988.
- [Revesz, 1997] P. Z. Revesz. On the semantics of arbitration. *International Journal of Algebra and Computation*, 7(2):133–160, 1997.
- [Satterthwaite, 1975] M.A. Satterthwaite. Strategy-proofness and Arrow’s conditions. *Journal of Economic Theory*, 10:187–217, 1975.

Strong Equivalence for Logic Programs with Preferences*

Wolfgang Faber[†]

Department of Mathematics
University of Calabria
I-87030 Rende (CS), Italy
Email: faber@mat.unical.it

Kathrin Konczak

Institut für Informatik
Universität Potsdam
PF 900327, D-14439 Potsdam, Germany
Email: konczak@cs.uni-potsdam.de

Abstract

Recently, strong equivalence for Answer Set Programming has been studied intensively, and was shown to be beneficial for modular programming and automated optimization. In this paper we define the novel notion of strong equivalence for logic programs with preferences. Based on this definition we give, for several semantics for preference handling, necessary and sufficient conditions for programs to be strongly equivalent. These results provide a clear picture of the relationship of these semantics with respect to strong equivalence, which differs considerably from their relationship with respect to answer sets. Finally, based on these results, we present for the first time simplification methods for logic programs with preferences.

1 Introduction

During the last decade, Answer Set Programming (ASP) [Gelfond and Lifschitz, 1988] has become an increasingly acknowledged tool for knowledge representation and reasoning. A main advantage of ASP is that it is based on solid theoretical foundations, while being able to model commonsense reasoning in an arguably satisfactory way. The availability of efficient solvers has furthermore stimulated its use in practical applications in recent years. This development had quite some implications on ASP research. For example, increasingly large applications require features for modular programming. Another requirement is the fact that in applications, ASP code is often generated automatically by so-called frontends, calling for optimization methods which remove redundancies, as also found in database query optimizers. For these purposes the rather recently suggested notion of strong equivalence for ASP [Lifschitz *et al.*, 2001; Turner, 2003] can be used. Indeed, if two ASP programs are strongly equivalent, they can be used interchangeably in

any context. This gives a handle on showing the equivalence of ASP modules. If a program is strongly equivalent to a subprogram of itself, then one can always use the subprogram instead of the original program, a technique which serves as an effective optimization method.

On a different line of ASP research, many extensions of the basic formalism have been proposed. Perhaps the most intensively studied one is the modeling of preferences in ASP, cf. [Delgrande *et al.*, 2004]. Strongly rooted in the research of nonmonotonic formalisms, the ability to specify preferences is acknowledged to be particularly beneficial to ASP, since they constitute a very natural and effective way of resolving indeterminate solutions. For example, preferences have been successfully used for timetabling, auctioning, configuration, and other domains. A sophisticated application for information site selection has been described in [Eiter *et al.*, 2002]. The emergence of such applications thus also calls for optimization methods, as for standard ASP.

In this paper we tackle this issue and generalize the notion of strong equivalence to ASP with preferences. Since a plethora of formalisms and semantics has been introduced for extending ASP by preferences, we have to limit ourselves in this paper to a few concrete approaches. A basic classification of preference handling is (i) over which elements the preference is defined and (ii) the way how preferences are interpreted, and in particular how they are aggregated. In this work we focus on ordered programs, where preferences are defined among rules, which semantically act as filters over answer sets. In particular, we consider the three semantics defined in [Delgrande *et al.*, 2003; Wang *et al.*, 2000; Brewka and Eiter, 1999], referred to as *D*-, *W*-, and *B*-semantics, respectively. The reason for this choice is that these semantics seem to be widely accepted and their properties and interrelationships are fairly well-understood.

The contributions of this work are as follows. We define the notion of strong equivalence for ordered programs. It turns out that the definition for standard ASP cannot be straightforwardly extended for two main reasons: (1) the “union” of ordered programs needs to be defined properly, and (2) not any “union” of ordered programs yields a valid ordered program. To overcome these problems, we define the notion of *admissible extension*, which provides a general method for defining strong equivalence for arbitrary structures. We show several properties of strong order equivalence, providing necessary

*The first author was partially supported by the EC under project IST-2002-33570 (INFOMIX). The second author was partially supported by the German Science Foundation (DFG) under grant SCHA 550/6-4, TP C. Both authors were partially supported by the EC under project IST-2001-37004 WASP.

[†]Funded by APART of the Austrian Academy of Sciences.

and sufficient conditions for programs to be strongly order equivalent. We also explore the relationship of strong order equivalence between the three semantics studied in this paper, and the relationship to strong equivalence of standard logic programs. Based on these results, we study the applicability of program simplification rules, known from standard ASP, in the presence of preferences. We obtain that most of these simplifications can be applied only in particular contexts. The main results of this work are that the relationship between the three preference semantics is changing under strong equivalence and that preferences cannot be “compiled away” in a modular fashion. We omit all proofs due to the page limit.¹

2 Background

Answer Sets. A logic program (LP) is a finite set of rules such as $p_0 \leftarrow p_1, \dots, p_m, \text{not } p_{m+1}, \dots, \text{not } p_n$, where $n \geq m \geq 0$, and each p_i ($0 \leq i \leq n$) is an atom. For such a rule r , we let $\text{head}(r)$ denote the head, p_0 , of r and $\text{body}(r)$ the body, $\{p_1, \dots, p_m, \text{not } p_{m+1}, \dots, \text{not } p_n\}$, of r . Let $\text{body}^+(r) = \{p_1, \dots, p_m\}$ and $\text{body}^-(r) = \{p_{m+1}, \dots, p_n\}$. For a set of rules Π , we write $\text{head}(\Pi) = \{\text{head}(r) \mid r \in \Pi\}$. A program is *basic* if $\text{body}^-(r) = \emptyset$ for all its rules. The *reduct*, Π^X , of a program Π relative to a set X of atoms is defined by $\Pi^X = \{\text{head}(r) \leftarrow \text{body}^+(r) \mid r \in \Pi, \text{body}^-(r) \cap X = \emptyset\}$. A set of atoms X is *closed* under a basic program Π if for any $r \in \Pi$, $\text{head}(r) \in X$ if $\text{body}^+(r) \subseteq X$. The smallest set of atoms being closed under a basic program Π is denoted by $\text{Cn}(\Pi)$. Then, a set X of atoms is an *answer set* of a program Π if $\text{Cn}(\Pi^X) = X$. We use $\text{AS}(\Pi)$ for denoting the set of all answer sets of Π . In what follows, an important concept is that of the *generating rules* of an answer set. Given a set X of atoms from Π , define $R_\Pi(X) = \{r \in \Pi \mid \text{body}^+(r) \subseteq X, \text{body}^-(r) \cap X = \emptyset\}$.

Strong Equivalence. Under the answer set semantics, two LPs Π_1 and Π_2 are regarded as equivalent, denoted $\Pi_1 \equiv \Pi_2$, iff $\text{AS}(\Pi_1) = \text{AS}(\Pi_2)$. Two LPs Π_1 and Π_2 are *strongly equivalent*, denoted $\Pi_1 \equiv_s \Pi_2$, iff, for any LP Π , the programs $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ are equivalent, i.e. $\Pi_1 \cup \Pi \equiv \Pi_2 \cup \Pi$. As shown in [Lifschitz et al., 2001], strong equivalence is closely related to the non-classical logic of here-and-there, which was adapted to logic-programming terms by Turner [Turner, 2003]: Let Π be an LP, and let $X, Y \subseteq A$ such that $X \subseteq Y$. The pair (X, Y) is an *SE-model* of Π (over A), if $Y \models \Pi$ and $X \models \Pi^Y$. By $\text{SE}^A(\Pi)$ we denote the set of all SE-models over A of Π . Usually the set A is left implicit and one just writes $\text{SE}(\Pi)$. Then, for any LPs Π_1 and Π_2 , $\Pi_1 \equiv_s \Pi_2$ iff $\text{SE}(\Pi_1) = \text{SE}(\Pi_2)$ [Turner, 2003].

Preferences. A logic program Π is said to be *ordered* if we have a set $\mathcal{N}$ of terms serving as *names* for rules and a *preference relation* $< \subseteq \mathcal{N} \times \mathcal{N}$ as a strict partial order among rules, where we write $s < t$ for the names $s, t \in \mathcal{N}$ of rules in Π . Furthermore, we assume a bijective function $n(\cdot)$ assigning to each rule $r \in \Pi$ a name $n(r) \in \mathcal{N}$. To simplify our notation, we usually write r_i instead of $n(r_i)$ for some $r_i \in \Pi$. Given $r_i, r_j \in \Pi$, $r_i < r_j$ states that r_j has

higher priority than r_i . Formally, an ordered (logic) program (OLP) can be understood as a quadruple $(\Pi, \mathcal{N}, n, <)$, where Π is a logic program, n is a bijective function between Π and the set of names $\mathcal{N}$, and $<$ is a set of preference relations over Π . Whenever possible, we leave $\mathcal{N}$ and n implicit and just write $(\Pi, <)$ for OLPs. Moreover, we write $< = \emptyset$ if no partial order is given that is $(\Pi, \emptyset)$ denotes a logic program. The interpretation that one rule has higher priority than another rule can be made precise in different ways. In what follows, we consider three such interpretations: *D*– [Delgrande et al., 2003], *B*– [Brewka and Eiter, 1999], and *W*– preference [Schaub and Wang, 2003]. Given $(\Pi, <)$, all of them use $<$ for selecting preferred answer sets among the standard answer sets of Π . We now recall the definitions of these semantics.²

Definition 2.1 Let $(\Pi, <)$ be an OLP and X be an answer set of Π . Then, X is $<^D$ –preserving, if an enumeration $\langle r_i \rangle_{i \in I}$ of Π exists s.t. for every $i, j \in I$ we have

1. if $r_i < r_j$, then $j < i$, and
2. if $r_i \in R_\Pi(X)$ then $\text{body}^+(r_i) \subseteq \{\text{head}(r_j) \mid r_j \in R_\Pi(X), j < i\}$, and
3. if $r_i \in \Pi \setminus R_\Pi(X)$ then
 - (a) $\text{body}^+(r_i) \not\subseteq X$ or
 - (b) $\text{body}^-(r_i) \cap \{\text{head}(r_j) \mid r_j \in R_\Pi(X), j < i\} \neq \emptyset$.

Cond.1 stipulates that the enumeration of Π is compatible with $<$. Cond.2 makes the property of supportedness explicit. Although any answer set is generated by a supported sequence of rules, in *D*–preferences, rules cannot be supported by lower-ranked ones. Cond.3a separates the handling of unsupported rules from preference handling. Cond.3b guarantees that rules can never be blocked by lower-ranked ones.

Definition 2.2 Let $(\Pi, <)$ be an OLP and X be an answer set of Π . Then, X is $<^W$ –preserving, if an enumeration $\langle r_i \rangle_{i \in I}$ of Π exists s.t. for every $i, j \in I$ we have

1. if $r_i < r_j$, then $j < i$, and
2. if $r_i \in R_\Pi(X)$ then
 - (a) $\text{body}^+(r_i) \subseteq \{\text{head}(r_j) \mid r_j \in R_\Pi(X), j < i\}$ or
 - (b) $\text{head}(r_i) \in \{\text{head}(r_j) \mid r_j \in R_\Pi(X), j < i\}$, and
3. if $r_i \in \Pi \setminus R_\Pi(X)$ then
 - (a) $\text{body}^+(r_i) \not\subseteq X$ or
 - (b) $\text{body}^-(r_i) \cap \{\text{head}(r_j) \mid r_j \in R_\Pi(X), j < i\} \neq \emptyset$ or
 - (c) $\text{head}(r_i) \in \{\text{head}(r_j) \mid r_j \in R_\Pi(X), j < i\}$.

For *W*–preference, the concept of order preservation from *D*–preferences is weakened in Condition 2 and 3 for suspending both conditions, whenever the head of a preferred rule is derivable in an alternative way.

Definition 2.3 Let $(\Pi, <)$ be an OLP and X be an answer set of Π . Then, X is $<^B$ –preserving, if an enumeration $\langle r_i \rangle_{i \in I}$ of Π exists s.t. for every $i, j \in I$ we have

1. if $r_i < r_j$, then $j < i$, and

¹A version of the paper including proofs can be found at <http://www.wfaber.com/research/#publications>

²In the following, I is an index set $I = \{1, \dots, n\}$ for $|\Pi| = n$.

2. if $r_i \in \Pi \setminus R_\Pi(X)$ then

- (a) $\text{body}^+(r_i) \not\subseteq X$ or
- (b) $\text{body}^-(r_i) \cap \{\text{head}(r_j) \mid r_j \in R_\Pi(X), j < i\} \neq \emptyset$ or
- (c) $\text{head}(r_i) \in X$.

B -preference drops Cond.2; thus decoupling preference handling from the order induced by consecutive rule applications.

For $\sigma \in \{D, W, B\}$, we define $AS^\sigma((\Pi, <))$ as the set of all $<^\sigma$ -preserving answer sets of OLP $(\Pi, <)$. As shown in [Schaub and Wang, 2003], the three strategies yield an increasing number of preferred answer sets. That is, D -preference is stronger than W -preference, which is stronger than B -preference, which is stronger than no preference. More precisely, we have $AS^D((\Pi, <)) \subseteq AS^W((\Pi, <)) \subseteq AS^B((\Pi, <)) \subseteq AS(\Pi)$ and, in addition, $AS^\sigma((\Pi, \emptyset)) = AS(\Pi)$ for $\sigma \in \{D, W, B\}$.

3 Strong Order Equivalence

In this section we define variants of equivalences for OLPs. Defining “standard” equivalence (two programs permit precisely the same answer sets) is straightforward by considering the respective preferred answer sets.

However, concerning the notion of strong equivalence, there are some caveats. For standard programs, strong equivalence for Π_1 and Π_2 holds if for any program Π , $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ are equivalent. If we now take preferences into account, for any $(\Pi, <)$, $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ would be strongly equivalent if $(\Pi_1 \cup \Pi, <_1 \cup <)$ and $(\Pi_2 \cup \Pi, <_2 \cup <)$ are equivalent. The problem is that $(\Pi_1 \cup \Pi, <_1 \cup <)$ is not necessarily an OLP, because of two issues: 1. Rule label clashes, and 2. partial order clashes.

Rule label clashes, i.e. two different rules have the same name or one rule has two different names, can be resolved by a *renaming*. We say that $(\Pi, \mathcal{N}^*, n^*, <^*)$ is a *renaming* of $(\Pi, \mathcal{N}, n, <)$ if $\mathcal{N}^*$ is a new set of names of rules, $n^*(\cdot)$ is a bijective function assigning each rule of Π an unique name of $\mathcal{N}^*$, and $n^*(r) <^* n^*(r')$ holds iff $n(r) < n(r')$, for all $r, r' \in \Pi$. The union of two OLPs is then defined as follows.

Definition 3.1 Let $(\Pi, \mathcal{N}, n, <)$ and $(\Pi', \mathcal{N}', n', <')$ be OLPs where $\mathcal{N}, \mathcal{N}'$ are disjoint sets of rule names.

We define the union of $(\Pi, \mathcal{N}, n, <)$ and $(\Pi', \mathcal{N}', n', <')$ as $(\Pi^*, \mathcal{N}^*, n^*, <^*)$, where

- $\Pi^* = \Pi \cup \Pi'$
- $\mathcal{N}^*$ as new set of names of rules,
- $n^*(r)$ for $r \in \Pi \cup \Pi'$ as a bijective function assigning each rule of $\Pi \cup \Pi'$ an unique name of $\mathcal{N}^*$, and
- $<^*$ as $< \cup <'$, where for $r, r' \in \Pi \cup \Pi'$, $n^*(r) <^* n^*(r')$ holds whenever $n(r) < n(r')$ or $n'(r) < n'(r')$ holds.

Again, we usually leave $\mathcal{N}^*$ and n^* implicit. Sometimes we write $(\Pi \cup \Pi', < \cup <')$ for the union of $(\Pi, <)$ and $(\Pi', <')$.

We next observe that the union of OLPs is not necessarily an OLP. For this, consider

$$\begin{aligned} (\Pi_1, <_1) &= \{r_1 : a \leftarrow, r_2 : b \leftarrow a, r_2 <_1 r_1\} \\ (\Pi_2, <_2) &= \{r_3 : a \leftarrow, r_4 : b \leftarrow a, r_3 <_2 r_4\} \end{aligned}$$

Then, we have

$$(\Pi^*, <^*) = \{r_5 : a \leftarrow, r_6 : b \leftarrow a, r_5 <^* r_6, r_6 <^* r_5\}$$

where $<^*$ is not a strict partial order. Hence, $(\Pi^*, <^*)$ is not an OLP. To avoid this, we define the notion of admissible extensions of an OLP.

Definition 3.2 Let $(\Pi, <)$ and $(\Pi', <')$ be OLPs.

Then, $(\Pi', <')$ is an *admissible extension* of $(\Pi, <)$ if $(\Pi \cup \Pi', < \cup <')$ is an OLP.

We remark that the notion of compatibility for orderings, as defined in [Brewka and Eiter, 1999], is similar in spirit to admissible extensions, but concerns only the order relation and not the rules in the program.

Analogously to (normal) logic programs, we can define the notion of equivalence of OLPs.

Definition 3.3 (Order equivalence) Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be OLPs and $\sigma \in \{D, B, W\}$. Then, $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ are $<^\sigma$ -equivalent, denoted $(\Pi_1, <_1) \equiv^\sigma (\Pi_2, <_2)$, iff $AS^\sigma((\Pi_1, <_1)) = AS^\sigma((\Pi_2, <_2))$.

Since the definition of strong equivalence for logic programs can be interpreted as having the same answer sets in any extension, we employ the notion of admissible extensions for defining the analogon for OLPs.

Definition 3.4 (Strong order equivalence) Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be OLPs and $\sigma \in \{D, B, W\}$.

Then, $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ are strongly $<^\sigma$ -equivalent iff for all admissible extensions $(\Pi, <)$ of $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ it holds that $(\Pi_1 \cup \Pi, <_1 \cup <)$ and $(\Pi_2 \cup \Pi, <_2 \cup <)$ are $<^\sigma$ -equivalent.

Abbreviatory, we write $\equiv_s^\sigma$ whenever two OLPs are strongly $<^\sigma$ -equivalent.

4 Properties and Relationships

Next, we want to characterize which conditions two OLPs must fulfill s.t. they are strongly order equivalent.

The next theorem states that strong order equivalence implies strong equivalence of the underlying logic programs.

Theorem 4.1 Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be OLPs and $\sigma \in \{D, W, B\}$. If $(\Pi_1, <_1) \equiv_s^\sigma (\Pi_2, <_2)$ then $\Pi_1 \equiv_s \Pi_2$.

Intuitively, if the logic programs are not strongly equivalent, one extension of them admits an answer set A , which is no answer set of the extension of the other program. If the corresponding ordered extensions have the same answer sets, this means that A is inhibited by the preferences. But then one can always construct a further extension, which regenerates A for one program, but not for the other. For example, programs

$$\begin{aligned} (\Pi_1, \emptyset) &= \{r_1 : a \leftarrow\} \\ (\Pi_2, <_2) &= \{r_2 : a \leftarrow \text{not } b, r_3 : b \leftarrow \text{not } a, r_3 <_2 r_2\} \end{aligned}$$

have both the preferred answer set $\{a\}$, whereas Π_2 has additionally the answer set $\{b\}$. Taking $(\Pi, \emptyset) = \{r_4 : b \leftarrow y, r_5 : y \leftarrow\}$ yields $\{a, b, y\}$ as preferred answer set of $(\Pi_1 \cup \Pi, \emptyset)$, but $\{b, y\}$ as preferred answer set of $(\Pi_2 \cup \Pi, <_2)$.

Note that one can create such a “compensating” program for any answer set which is inhibited by the preference relation. For this, we use new symbols and new rules, which are

not involved in the preference relation, to “repair” this answer set. Therefore strong equivalence is a necessary condition for strong order equivalence.

Next, we turn our attention to the set of generation rules. For this, consider the following OLPs:

$$\begin{aligned}(\Pi_1, \emptyset) &= \{r_1 : a \leftarrow, r_3 : b \leftarrow\} \\ (\Pi_2, \emptyset) &= \{r_1 : a \leftarrow, r_2 : b \leftarrow a\}\end{aligned}$$

We observe that $\Pi_1 \equiv_s \Pi_2$, but taking

$$(\Pi^*, <^*) = \{r_2 : b \leftarrow a, r_4 : y \leftarrow \text{not } b, r_2 <^* r_4\}$$

yields $\{a, b\} \in AS^\sigma((\Pi_1 \cup \Pi^*, <^*))$ but $AS^\sigma((\Pi_2 \cup \Pi^*, <^*)) = \emptyset$ for $\sigma \in \{D, W, B\}$. They are not strongly order equivalent because they do not coincide on their set of generating rules. $(\Pi_1, \emptyset)$ contains r_3 and $(\Pi_2, \emptyset)$ rule r_2 to derive the atom b . Adding r_4 and the preference that r_4 is higher preferred to r_2 entails that $(\Pi_2 \cup \Pi^*, <^*)$ has no preferred answer set since r_4 is blocked by the lower preferred rule r_2 , while $(\Pi_1 \cup \Pi^*, <^*)$ has a preferred answer set since r_3 allows to block r_4 in an alternative way. This allows us to suppose that strong order equivalence enforces that the OLPs must coincide on all rules which can become applicable. For this, we say that a rule $r \in \Pi$ contributes to an answer set X , if there exists an Π^* such that $X \in AS(\Pi \cup \Pi^*)$ and $r \in R_{\Pi \cup \Pi^*}(X)$. Rules which are not contributing to answer sets, are never applicable, e.g. $body^+(r) \cap body^-(r) \neq \emptyset$. For B -preferences, we obtain that two strongly order equivalent programs must coincide on their rules contributing to answer sets.

Theorem 4.2 *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be OLPs. If $(\Pi_1, <_1) \equiv_s^B (\Pi_2, <_2)$, then for all Π and for all $X \in AS(\Pi_1 \cup \Pi) = AS(\Pi_2 \cup \Pi)$ we have $R_{\Pi_1 \cup \Pi}(X) = R_{\Pi_2 \cup \Pi}(X)$.*

Interestingly, for D - and W -preferences, these conditions have to be weakened.

Theorem 4.3 *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be OLPs and $\sigma \in \{D, W\}$. If $(\Pi_1, <_1) \equiv_s^\sigma (\Pi_2, <_2)$, then for all Π and for all $X \in AS(\Pi_1 \cup \Pi) = AS(\Pi_2 \cup \Pi)$ we have $R_{\Pi_1 \cup \Pi}(X) \setminus \{r \in \Pi_1 \cup \Pi \mid head(r) \in body^+(r)\} = R_{\Pi_2 \cup \Pi}(X) \setminus \{r \in \Pi_2 \cup \Pi \mid head(r) \in body^+(r)\}$.*

Since D - and W - preferences couple rule application with preference handling, answer sets can be inhibited by preferences because applied rules are not enumerable in an order preserving way (see Condition 2 in Definition 2.1 and 2.2). But one can always find an extension such that those answer sets are regenerated. For applied “loop” rules, i.e. $head(r) \in body^+(r)$, another rule must always exist which derives the head in an order preserving way. Hence, such loop rules are redundant wrt. D - and W - preferences as long as they are not involved in the preference relations. In contrast, B - preferences decouple preference handling from rule application and hence, one can always find an extension, where those loop rules regenerate a non-preferred answer set to a preferred one in one program but not in the other one. For example,

$$(\Pi_1, \emptyset) = \{r_1 : a \leftarrow\} \quad (\Pi_2, \emptyset) = \{r_1 : a \leftarrow, r_2 : a \leftarrow a\}$$

are not strongly $<^B$ - equivalent. Taking

$$(\Pi, <) = \{r_1 : a \leftarrow, r_3 : y \leftarrow \text{not } a, r_1 < r_3\}$$

yields $AS^B((\Pi_1 \cup \Pi, <)) = \emptyset$ but $AS^B((\Pi_2 \cup \Pi, <)) = \{a\}$. Here, rule r_2 can be used to block r_3 . For D - and W -preferences, however, r_2 has to be ordered after r_1 , hence it can not be used to block r_3 .

In the following we will concentrate on the preference relations. For this consider the program

$$\Pi = \{r_1 : a \leftarrow, r_2 : b \leftarrow a\}$$

and the preference relation $r_1 > r_2$. We obtain that $(\Pi, \emptyset) \equiv^\sigma (\Pi, <)$ holds for all $\sigma \in \{D, W, B\}$ and $(\Pi, \emptyset)$ and $(\Pi, <)$ coincide on their set of rules. For $(\Pi, \emptyset)$, rule r_1 must be considered before rule r_2 in D - and W -preference, since the derivation of the positive body of r_2 depends from r_1 . Thus, adding the preference relation $r_1 > r_2$ to $(\Pi, \emptyset)$ has no effect on $<^D$ - and W - preferred answer sets. Surprisingly, this does not hold when considering strong order equivalence. Taking

$$(\Pi', <') = \left\{ \begin{array}{lll} r_1 : a \leftarrow & r_3 : z \leftarrow & r_5 : a \leftarrow x \\ r_2 : b \leftarrow a & r_4 : y \leftarrow \text{not } z & r_6 : x \leftarrow \\ r_3 < r_2 & r_1 < r_4 & \end{array} \right\}$$

yields $AS^\sigma((\Pi \cup \Pi', <')) = \{\{a, b, x, z\}\}$ while $AS^\sigma((\Pi \cup \Pi', < \cup <')) = \emptyset$. For $(\Pi, <) \cup (\Pi', <')$ we get the transitive preference $r_4 > r_3$, implying that r_4 is blocked by a lower preferred rule, which we don’t obtain for $(\Pi, \emptyset) \cup (\Pi', <')$. Since one can always construct such a program (observe that the relevant rules contain new symbols), it follows that two OLPs can never be strongly equivalent if the preference relations do not coincide.

Theorem 4.4 *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be OLPs and $\sigma \in \{D, W, B\}$. If $(\Pi_1, <_1) \equiv_s^\sigma (\Pi_2, <_2)$ then $<_1 = <_2$.*

As a special case we obtain that $(\Pi, \emptyset) \not\equiv_s^\sigma (\Pi, <)$ holds for any program Π , any non-empty preference relation on Π , and any $\sigma \in \{D, W, B\}$. Moreover, for every OLP $(\Pi, <)$, $< \neq \emptyset$, there exists no strongly order equivalent program $(\Pi', \emptyset)$.

Corollary 4.5 *Let $(\Pi, <)$ be an OLP such that $< \neq \emptyset$ and $\sigma \in \{D, W, B\}$. Then, there exists no logic program Π' such that $(\Pi, <) \equiv_s^\sigma (\Pi', \emptyset)$.*

Interestingly, this tells us that no transformation from an OLP with a non-empty preference relation to a logic program exists such that the transformed program is strongly order equivalent to the original one. The results given above are sufficient for characterizing strong order equivalence.

Theorem 4.6 *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be OLPs. If $\Pi_1 \equiv_s \Pi_2$, $<_1 = <_2$, and for all Π and for all $X \in AS(\Pi_1 \cup \Pi) = AS(\Pi_2 \cup \Pi)$ we have $R_{\Pi_1 \cup \Pi}(X) = R_{\Pi_2 \cup \Pi}(X)$, then $(\Pi_1, <_1) \equiv_s^B (\Pi_2, <_2)$.*

Theorem 4.7 *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be OLPs and $\sigma \in \{D, W\}$. If $\Pi_1 \equiv_s \Pi_2$, $<_1 = <_2$, and for all Π and for all $X \in AS(\Pi_1 \cup \Pi) = AS(\Pi_2 \cup \Pi)$ we have $R_{\Pi_1 \cup \Pi}(X) \setminus \{r \in \Pi_1 \cup \Pi \mid head(r) \in body^+(r)\} = R_{\Pi_2 \cup \Pi}(X) \setminus \{r \in \Pi_2 \cup \Pi \mid head(r) \in body^+(r)\}$, then $(\Pi_1, <_1) \equiv_s^\sigma (\Pi_2, <_2)$.*

For example,

$$\begin{aligned}(\Pi_1, \emptyset) &= \{r_1 : a \leftarrow, r_2 : b \leftarrow \text{not } a\} \\ (\Pi_2, \emptyset) &= \{r_1 : a \leftarrow, r_3 : c \leftarrow \text{not } a\}\end{aligned}$$

are strongly order equivalent in all three semantics, whereas

$$(\Pi_3, \emptyset) = \{r_1 : a \leftarrow\} \quad (\Pi_4, \emptyset) = \{r_1 : a \leftarrow, r_2 : a \leftarrow a\}$$

are only strongly $<^D$ - (and $<^W$ -) equivalent, but not strongly $<^B$ - equivalent. With these results we can formalize the following characterization of strong order equivalence.

Corollary 4.8 *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be OLPs. Then, $(\Pi_1, <_1) \equiv_s^B (\Pi_2, <_2)$ iff $\Pi_1 \equiv_s \Pi_2$, $<_1 = <_2$, and for all Π and for all $X \in AS(\Pi_1 \cup \Pi) = AS(\Pi_2 \cup \Pi)$ we have $R_{\Pi_1 \cup \Pi}(X) = R_{\Pi_2 \cup \Pi}(X)$.*

Corollary 4.9 *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be OLPs and $\sigma \in \{D, W\}$. Then, $(\Pi_1, <_1) \equiv_s^\sigma (\Pi_2, <_2)$, iff $\Pi_1 \equiv_s \Pi_2$, $<_1 = <_2$, and for all Π and for all $X \in AS(\Pi_1 \cup \Pi) = AS(\Pi_2 \cup \Pi)$ we have $R_{\Pi_1 \cup \Pi}(X) \setminus \{r \in \Pi_1 \cup \Pi \mid head(r) \in body^+(r)\} = R_{\Pi_2 \cup \Pi}(X) \setminus \{r \in \Pi_2 \cup \Pi \mid head(r) \in body^+(r)\}$.*

Regarding preferred answer sets, the different strategies for preference handling yield an increasing number of preferred answer sets. Interestingly, the above given theorems show that there is no difference in the behavior of D - and W -preferences when considering strong order equivalence. In contrast, fewer programs are strongly order equivalent in the B -semantics, since preference handling is decoupled from rule application.

Theorem 4.10 *Let $(\Pi_1, <_1)$ and $(\Pi_2, <_2)$ be OLPs. Then (a) if $(\Pi_1, <_1) \equiv_s^B (\Pi_2, <_2)$ then $(\Pi_1, <_1) \equiv_s^W (\Pi_2, <_2)$, (b) $(\Pi_1, <_1) \equiv_s^W (\Pi_2, <_2)$ iff $(\Pi_1, <_1) \equiv_s^D (\Pi_2, <_2)$.*

Considering answer sets, each $<^D$ -preferred answer set is $<^W$ -preferred and each $<^W$ -preferred is $<^B$ -preferred. Interestingly, regarding strong order equivalence we obtain that $\equiv_s^D$ iff $\equiv_s^W$, so the differences between the D - and W -semantics can be compensated by extending the program in a suitable way. Moreover, since $\equiv_s^B$ implies $\equiv_s^W$ and $\equiv_s^W$ implies $\equiv_s^D$, the B -semantics imposes a strictly stronger criterion for programs to be strongly order equivalent. The reason for this is that B -preference decouples preference handling from rule application, and in addition, these differences cannot be compensated by adding suitable extensions.

5 Program Simplifications

Corollaries 4.9 and 4.8 show that two OLPs which are strongly order equivalent, differ in rules which are not involved in the preference relation and which are never applicable. Additionally, for D - and W - semantics, strongly order equivalent logic programs may differ in loop rules, in which the head is contained in the positive body. These results allow us to formalize the following results concerning transformations on OLPs. In the following, we say that a rule r is involved in a preference relation $<$ if there exists an r' such that $r < r'$ or $r' < r$ holds.

Corollary 5.1 *Let $(\Pi, <)$ be an OLP such that for $r \in \Pi$, which is not involved in $<$, for all Π' and $X \in AS(\Pi \cup \Pi')$ we have $r \notin R_{\Pi \cup \Pi'}(X)$. Then, $(\Pi, <) \equiv_s^\sigma (\Pi \setminus \{r\}, <)$ holds for $\sigma \in \{D, W, B\}$.*

This corollary expresses that a rule, which is never applicable and not involved in $<$, is redundant regarding strong order equivalence. The next corollary concentrates on loop rules.

Corollary 5.2 *Let $(\Pi, <)$ be an OLP and $\sigma \in \{D, W\}$. Furthermore, let $r \in \Pi$ s.t. $head(r) \in body^+(r)$ and r is not involved in $<$. Then, $(\Pi, <) \equiv_s^\sigma (\Pi \setminus \{r\}, <)$.*

In [Eiter et al., 2004; Osorio et al., 2001] transformations on logic programs are reported, which can be used for simplifying a program. For those modular transformations a program must be strongly equivalent to the transformed one. Considering OLPs, we observe that these transformations in general do not guarantee strong order equivalence. Transformation *TAUT* expresses that for all logic programs Π and all $r \in \Pi$ where $head(r) \in body^+(r)$ we have $\Pi \equiv_s \Pi \setminus \{r\}$. Corollary 5.2 implies that the analogon for OLPs only holds for the D - and W - semantics as long as r is not involved in $<$. Transformation *RED*⁻ expresses that for a logic program Π and for rules r_1, r_2 , where $head(r_2) \in body^-(r_1)$ and $body(r_2) = \emptyset$, we have $\Pi \equiv_s \Pi \setminus \{r_1\}$. Since r_2 is always a generating rule, regardless which rules we add to Π , r_1 never contributes to an answer set. Hence, *RED*⁻ is feasible for OLPs unless r_1 is involved in $<$. For $r \in \Pi$, where $body^+(r) \cap body^-(r) \neq \emptyset$ we have that $\Pi \equiv_s \Pi \setminus \{r\}$ (transformation *CONTRA*). Since such a rule is never applicable, this result can be carried over to OLPs whenever this rule is not involved in $<$. Transformation *NONMIN* states that for $r_1, r_2 \in \Pi$, where $head(r_2) = head(r_1)$ and $body(r_2) \subseteq body(r_1)$, we have that $\Pi \equiv_s \Pi \setminus \{r_1\}$ holds. Since there is no information whether r_1 can become applicable or not, this transformation can only be made on OLPs whenever r_1 is not involved in $<$ and never becomes applicable. For example, consider the program:

$$(\Pi_1, \emptyset) = \{r_1 : a \leftarrow, r_2 : b \leftarrow, r_3 : c \leftarrow a, r_4 : c \leftarrow a, b\}$$

Here we have that $head(r_3) = head(r_4)$ and $body(r_3) \subseteq body(r_4)$, hence, $\Pi_1 \equiv_s \Pi_1 \setminus \{r_4\}$. Since r_4 is applicable, the simplified program is not strongly order equivalent. Taking

$$(\Pi^*, <^*) = \{r_3 : c \leftarrow a, r_5 : y \leftarrow not\ c, r_3 <^* r_5\}$$

yields $\{a, b, c\} \in AS^\sigma((\Pi_1 \cup \Pi^*, <^*))$ but $AS^\sigma((\Pi_1 \setminus \{r_4\} \cup \Pi^*, <^*)) = \emptyset$, for $\sigma \in \{D, W, B\}$. Thus, $(\Pi_1, \emptyset) \not\equiv_s^\sigma (\Pi_1 \setminus \{r_4\}, \emptyset)$. Similar considerations apply for the transformation *S-IMP*, where for all $r, r' \in \Pi$ such that there exists an $A \subseteq body^-(r')$ such that $head(r) \in head(r') \cup A$, $body^-(r) \subseteq body^-(r') \setminus A$ and $body^+(r) \subseteq body^+(r')$ we have $\Pi \equiv_s \Pi \setminus \{r'\}$. This transformation can only be carried over to OLPs whenever r' is not involved in $<$ and never becomes applicable. For example, taking

$$\Pi = \{r' : a \leftarrow b, not\ c, r : a \leftarrow\}$$

and $A = \{c\}$ yields $\Pi \equiv_s \Pi \setminus \{r'\}$, where r' is applicable wrt $\{a, b\} \in AS(\Pi \cup \{b \leftarrow\})$. But $(\Pi, \emptyset) \not\equiv_s^\sigma (\Pi \setminus \{r'\}, \emptyset)$, since for

$$(\Pi^*, <^*) = \{r : a \leftarrow, r^y : y \leftarrow not\ a, r^b : b \leftarrow, r <^* r^y\}$$

we get $\{a, b\} \in AS^\sigma((\Pi \cup \Pi^*, <^*))$ but $\emptyset = AS^\sigma((\Pi \setminus \{r'\} \cup \Pi^*, <^*))$ for $\sigma \in \{D, W, B\}$. Transformation *WGPPE* states for a rule $r_1 \in \Pi$, where $a \in body^+(r_1)$, $G_a = \{r_2 \in \Pi \mid$

$head(r_2) = a\}$ and $G_a \neq \emptyset$ that $\Pi \equiv_s \Pi \cup G'_a$ holds where $G'_a = \{head(r_1) \leftarrow (body^+(r_1) \setminus \{a\}) \cup not\ body^-(r_1) \mid body(r_2) \mid r_2 \in G_a\}$. As with the other transformations obtained for strong equivalence, this one is transferable to strong order equivalence if the rules from G_a become never applicable and are not involved in the preference relation. E.g., for

$$\Pi = \{r_1 : b \leftarrow a, r_2 : a \leftarrow\}$$

we have that $\Pi \equiv_s \Pi \cup \{b \leftarrow\}$ holds. By taking

$$(\Pi^*, <^*) = \{r_1 : b \leftarrow a, r^y : y \leftarrow not\ b, r^y >^* r_1\}$$

we obtain $(\Pi, \emptyset) \not\equiv_s^\sigma (\Pi \cup \{b \leftarrow\}, \emptyset)$ since $(\Pi \cup \Pi^*, <^*)$ has no $<^\sigma$ -preferred answer set while $(\Pi \cup \{b \leftarrow\} \cup \Pi^*, <^*)$ has $<^\sigma$ -preferred answer set $\{a, b\}$, for $\sigma \in \{D, W, B\}$.

6 Conclusions and Future Work

We have presented the notion of strong order equivalence for ASP with preferences. We have provided an extensive analysis of this novel notion, and studied the relationship between different preference semantics for it. We have also analyzed the conditions for certain simplification methods to guarantee strong order equivalence (hence applicability to modules).

We could show in Corollaries 4.8 and 4.9 that two programs are strongly W - and D -equivalent, if (and only if) the preference relation is equal, their standard ASP programs are strongly equivalent, and their “non-looping” generating rules are identical for all answer sets of any extension of them. For B -preference, also the “looping” generating rules must be identical in any answer set.

From Theorem 4.10 we obtain that exactly the same pairs of programs are strongly $<^D$ - and $<^W$ -equivalent, while not all of them are strongly $<^B$ -equivalent. That is, the semantic differences between D - and W -preference have no longer any effect when considering strong order equivalence. On the contrary, for B - semantics the differences to D - and W -preferences are strengthened under strong order equivalence, since it decouples preference handling from rule application, which can not be compensated by suitable extensions.

We have furthermore studied possibilities to simplify ordered programs. In Corollaries 5.1 and 5.2 we have given abstract conditions for simplifications, and have assessed under which circumstances simplification rules from standard ASP can be applied on ASP modules with preferences.

The reader may have noted that we did not define a notion which would correspond to SE-models in standard ASP strong equivalence. Indeed there seem to be considerable difficulties: SE-models can only express statements on atoms, and not directly on rules. However, as demonstrated e.g. in Theorem 4.3, strong order equivalence depends heavily on the rules involved in the program. We therefore conjecture that “SE-models” for D -, W -, and B - preferred answer sets must include a means for stating relationships among rules, and leave this topic for future work.

Another way of analyzing strong order equivalence could be offered by a compilation of the programs, as done e.g. in the plp-System [Delgrande *et al.*, 2003; plp, 2002]. There, ordered programs are compiled into logic programs such that their answer sets directly correspond to the preferred answer

sets of the original ordered program. Since the compilation is a logic program, one could suppose that two ordered programs are strongly order equivalent if the compilation of them possess the same SE-models. However, the compilation introduces additional symbols. Hence, the answer sets of two compiled programs, which represent the same preferred answer set, are in general incomparable because of these additional symbols. One would therefore have to consider strong equivalence under a projection of the answer sets. Furthermore, for an exact notion one would also have to take into account that compiled programs have a particular, rather than arbitrary, structure. Such a notion of strong equivalence has (to the best of our knowledge) not been studied so far. Therefore, we leave this kind of analysis for future work.

Corollary 4.5 shows that no transformation from an ordered program $(\Pi, <)$ into a logic program Π' exists such that $(\Pi, <)$ is strongly order equivalent to Π' . Also, Corollaries 4.8 and 4.9 describe rigorous conditions on ordered programs being strongly order equivalent under the considered preference semantics. Hence, further work could be to define and study weaker notions of order equivalence.

References

- [Brewka and Eiter, 1999] G. Brewka and T. Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2):297–356, 1999.
- [Delgrande *et al.*, 2003] J. Delgrande, T. Schaub, and H. Tompits. A framework for compiling preferences in logic programs. *TPLP*, 3(2):129–187, 2003.
- [Delgrande *et al.*, 2004] J. Delgrande, T. Schaub, H. Tompits, and K. Wang. A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence*, 20(2):308–334, 2004.
- [Eiter *et al.*, 2004] T. Eiter, M. Fink, H. Tompits, and S. Woltran. Simplifying logic programs under uniform and strong equivalence. In *LPNMR'04*, pages 87–99. Springer, 2004.
- [Eiter *et al.*, 2002] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. A Generic Approach for Knowledge-Based Information Site Selection. In *KR-02*, pages 459–469. Morgan Kaufmann, 2002.
- [Gelfond and Lifschitz, 1988] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *ICLP*, pages 1070–1080. The MIT Press, 1988.
- [Lifschitz *et al.*, 2001] V. Lifschitz, D. Pearce, and A. Valverde. Strongly Equivalent Logic Programs. *ACM TOCL*, 2(4):526–541, 2001.
- [Osorio *et al.*, 2001] M. Osorio, J.A. Navarro, and J. Arrazola. Equivalence in answer set programming. In *LOPSTR*, pages 57–75. Springer, 2001.
- [plp, 2002] <http://www.cs.uni-potsdam.de/~torsten/plp>, 2002.
- [Schaub and Wang, 2003] T. Schaub and K. Wang. A semantic framework for preference handling in answer set programming. *TPLP*, 3(4-5):569–607, 2003.
- [Turner, 2003] H. Turner. Strong equivalence made easy: nested expressions and weight constraints. *TPLP*, 3(4-5):609–622, 2003.
- [Wang *et al.*, 2000] K. Wang, L. Zhou, and F. Lin. Alternating Fixpoint Theory for Logic Programs with Priority. In *CL 2000*, pages 164–178. Springer.

Representing Flexible Temporal Behaviors in the Situation Calculus

Alberto Finzi and Fiora Pirri
DIS- Università di Roma “La Sapienza”
Via Salaria 113, I-00198 Rome, Italy
{finzi,pirri}@dis.uniroma1.it

Abstract

In this paper we present an approach to representing and managing temporally-flexible behaviors in the Situation Calculus based on a model of time and concurrent situations. We define a new hybrid framework combining temporal constraint reasoning and reasoning about actions. We show that the Constraint Based Interval Planning approach can be imported into the Situation Calculus by defining a temporal and concurrent extension of the basic action theory. Finally, we provide a version of the Golog interpreter suitable for managing flexible plans on multiple timelines.

1 Introduction

In real-world domains robots have to perform multiple tasks either simultaneously or in rapid alternation, employing diverse sensing and actuating tools, like motion, navigation, visual exploration, mapping, and several modes of perception. To ensure a suitable multiple-task performance, some approaches (e.g. [14; 23]) have recommended that executive control processes supervise the selection, initiation, execution, and termination of actions. From these ideas, a new paradigm has been proposed, called the Constraint Based Interval Planning (CBI), which essentially amalgamates planning, scheduling and resources optimization for reasoning about all the competing activities involved in a flexible concurrent plan (see [10; 4; 7]). The CBI approach, and similar approaches emerged from the planning community, have shown a strong practical impact when it comes to real world applications (see e.g. RAX [10], IxTeT [7], INOVA [21], and RMPL [23]). However, from the standpoint of cognitive robotics it is important to ensure both optimal performance, in practical applications, and to provide a logical framework for ensuring coherence of actions preconditions and effects. The system coherence emerges as a core issue also when control processes negotiate resources allocation with individual perceptual-motor and cognitive processes; indeed, the executive has to establish priorities among individual processes to allot resources for multiple-task performance (see the discussion in [8; 11; 3]). Therefore, different, concurrent, and interleaving behaviors, subject to switching-time criteria and current situation needs, lead to a new integration paradigm. In this paper we suggest that the reactive aspects that have

to cope with flexible behaviors and the cognitive capabilities enabling reasoning about these processes, can be combined in the Temporal Flexible Situation Calculus. We present a new approach to flexible behaviors, that exploits the full expressiveness of the Situation Calculus (SC) [19; 12], where computational concerns related to time can be monitored by a temporal network, obtained via a transformation of added constraints. To embed many concepts elaborated in the CBI framework we extend the SC with concurrency and time (extensions of SC with time was already explored in [15; 17; 16]), deploying Allen’s interval relations [1], and further constraining the language to represent concurrent timelines. In this framework we can conjugate the advantages of the SC with the expressive power of the CBI paradigm. Our aim here is twofold: on the one hand, it is made possible to introduce a separated timeline for each component of the dynamic system (i.e. each entity, which is part of an autonomous system, such as a robot), so that concurrency and flexibility can be clearly addressed; on the other hand the causal relationships among processes can be dealt with in the SC language, which provides a clear framework for preconditions and postconditions of actions and a simple solution to the frame problem. We show that the CBI perspective, with all its arsenal of specifications in terms of flexible time, alternation constraints, resources optimization, failure recovering, and tasks scheduling, can be imported into the SC ([19; 12]), defining a temporal and concurrent extension of the basic action theory (related approaches are [15; 17; 16; 8; 20; 6]). Finally, we provide a version of the Golog language and interpreter for manipulating flexible plans on multiple timelines.

2 Preliminaries

2.1 Situation Calculus and Golog

The Situation Calculus [12; 19] is a *sorted first order language* for representing dynamic domains by means of *actions*, *situations*, and *fluents*. *Actions* and *situations* are first order terms, and *situation*-terms stand for history of actions, compound with the binary symbol *do*: *do(a, s)* is the situation obtained by executing the action *a* after the sequence *s*. The dynamic domain is described by a *Basic Action Theory BAT* $= (\Sigma, \mathcal{D}_{S_0}, \mathcal{D}_{ssa}, \mathcal{D}_{una}, \mathcal{D}_{ap})$. We refer the reader to [19] for a complete introduction to the SC. Temporal Concurrent Situation Calculus (*TCSC*) has been earlier introduced in [15; 18; 17]; actions are instantaneous, and their time is selected

by the function $time(\cdot)$. Durative actions are considered as *processes* [15; 19], represented by fluents, and durationless actions are to start and terminate these processes. For example, $going(hill, s)$ is started by the action $startGo(hill, t)$ and it is ended by $endGo(hill, t')$.

Golog. Golog is a situation calculus-based programming language for denoting complex actions composed of the primitive (simple or concurrent) actions defined in the *BAT*. Golog programs are defined by means of standard (and not so-standard) Algol-like control constructs: i. action sequence: $p_1; p_2$, ii. test: $\phi?$, iii. nondeterministic action choice $p_1 | p_2$, iv. conditionals, while loops, and procedure calls. An example of a Golog program is:

```
while  $\neg at(hill, 3)$  do
  if  $\neg (\exists x) going(x)$  do  $\pi(t, (t < 3)?; startGo(hill, t))$ 
```

The semantics of a Golog program δ is a SC formula $Do(\delta, s, s')$ meaning that s' is a possible situation reached by δ once executed from s . For example, $Do(a; p, s, s') \doteq Do(p, do(a, s), s') \wedge Poss(a, s)$ defines the execution of an action a followed by the program p . For more details on the SC and Golog we refer the reader to [19].

2.2 Constraint Based Interval Paradigm

The constraint based interval framework (CBI) [10; 4], is a well known framework for temporal planning systems combining temporal reasoning and scheduling, including, e.g., RAX [10], IxTeT [7], and INOVA[21]. The CBI paradigm accounts for concurrency and time, and action instances and states are described in terms of temporal intervals linked by constraints. We refer to the timeline-based version of the CBI [10] where a domain behavior is seen as the continuous interaction of different components, and each component is represented by *state variables*; a single state variable is a relevant feature of the components and represents a concurrent thread. Both states and activities are uniformly treated as temporal intervals. The history of states for a state variable over a period of time is called a *timeline*. Figure 1 illustrates two timelines (state variables) repr. the engine and the navigation processes of a mobile robot: initially, the robot is $at(p_1)$ and $stop$; when it starts $go(p_1, p_2)$ the engine is $running$; the engine is $stop$ again once the rover arrives $at(p_2)$. Some temporal constraints among the activities are: $at(x)$ holds only if $stop$ holds, $go(x, y)$ is followed by $at(y)$.

Domain Constraints. A *CBI model* [10] $\mathcal{M}_{cbi} = (X, J, \mathcal{R})$ is defined by: i. a set $X = \{x_1, \dots, x_n\}$ of state variables, one for each component (e.g. x_{loc} and x_{eng} in Fig. 1); ii. a set $J = \{J_1, \dots, J_n\}$ s.t. for each x_i the set J_i collects the associated temporal fluents $p_{i,j}(\vec{y})$, e.g. $at(x)$ and $go(x, y)$ for *location* state variable in Fig. 1; iii. a set of temporal constraints $\mathcal{R} = \{r_{i,j}\}$, usually called *compatibilities*: for each temporal fluent property $p_{i,j}$ there is a *compatibility* relation $r_{i,j}$ representing all its possible legal relations with the other temporal fluents, i.e. which temporal property must proceed, follow, be co-temporal, etc. to $p_{i,j}$ in a legal plan. The latter are specified in terms of metric version of temporal relations *a la Allen* [1]. E.g. the arrows in Fig. 1 illustrate the compatibility associated to the fluent at : $at(x)$ *meets* $go(x, y)$, and $at(x)$ *during* $stop$.

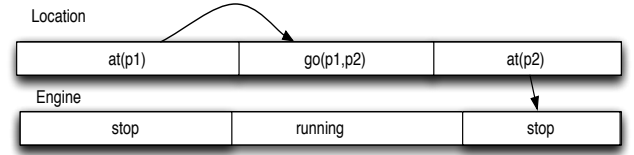


Figure 1: Timelines and constraints

Planning Problem in CBI. Given the *CBI model* $\mathcal{M}_{cbi}$, a *planning problem* is defined by $\mathcal{P}_{cbi} = (\mathcal{M}_{cbi}, P_c)$, where P_c is a *candidate plan*, representing an incomplete instance of a plan. The *candidate plan* defines both the initial situation and the goals. In particular, P_c consists of: i. a *planning horizon* h ; ii. a set of temporal properties to be satisfied for each state variable $x_i \in X$ (e.g. $at(p_1)$ ends before 10 and after 20 in the timeline of Figure 1); iii. the set of precedence constraints among fluents $p_{k,1} \prec p_{k,2}$, which are to hold on a timeline, e.g. $at(p_1) \prec at(p_2)$; iv. the set of constraints $\mathcal{R} = \{r_{i,j}\}$ associated with the temporal properties $p_{i,j}$ mentioned in the timelines. A fluent $p_{i,j}(\vec{x})$ mentioned in a *candidate plan* is *fully supported* [10] if all its associated constraints $r_{i,j}$ are satisfied. E.g. in Figure 1 $at(p_1)$ and $at(p_2)$ are fully supported. A *candidate plan* is said to be a *complete plan* if: i. each temporal property on each timeline is fully supported; ii. all timelines fully cover the planning horizon. Given the *planning problem* $(\mathcal{M}_{cbi}, P_c)$, the planning task is to provide a *sufficient plan* [10], i.e. a *complete plan* with the maximum flexibility: the planner is to minimally ground the (temporal) variables to allow for on-line binding of the values.

3 Temporal Flexible Situation Calculus

In this section we present the Temporal Flexible Situation Calculus framework (*TFSC*), which integrates the CBI paradigm, introduced above, into the language of SC.

Actions. We define a partition $\{\alpha_v; v \in \mathcal{C}\}$ of the set of actions according to the different components $\mathcal{C}$ the system has to care of. For example, the component *Pan-Tilt-Unit* manages actions like start-scan $s_{pts}(\cdot)$, or end-scan $e_{pts}(\cdot)$. To induce the partition we introduce a type operator ν , and extend the SC foundational axioms with the following definitions. Let $\mathcal{H} = \bigcup_{i=1}^n H_i$ be a set specifying the types of actions, we assume this set finite. For each name of action the following holds:

$H_i(a) \wedge H_j(a') \rightarrow \neg(a = a')$ for $i \neq j$, and $\nu(A(\vec{x})) = \nu(A'(\vec{y})) \equiv \bigvee_{i=1}^n H_i(A(\vec{x})) \wedge H_i(A'(\vec{y}))$, together with the unique name for actions, here the indices stay only for different components names, a denotes an action variable, while A denotes the name of an action function.

Situations. Typization of actions induces also a partition on the set of situations whence on the set of histories. Histories become streams of situations over timelines. Typization is inherited by situations as follows:

$$\begin{aligned} \forall a. \quad & \nu(a) = \nu(do(a, S_0)). \\ \forall a, s. \quad & \nu(a) = \nu(s) \equiv \\ & \forall a', s'. s = do(a', s') \rightarrow \nu(a) = \nu(a') \wedge \nu(a') = \nu(s'). \end{aligned} \quad (1)$$

For each component $v \in \mathcal{C}$, a timeline is specified by a history of actions concerning v . The evolution of the set of timelines is a set of situations $sc = \{s_v | v \in \mathcal{C}\}$, called *situation class*, spanning different types. In other words, $\exists s.s \in sc$ abbreviates: $\exists s_1 \dots s_n \exists a_1 \dots a_n. \bigvee_{i=1}^n [s = do(a_i, s_i) \wedge \nu(a_i) = \nu(s_i) \wedge \bigwedge_{j=1, i \neq j}^n \nu(a_i) \neq \nu(s_j)]$. Where $\nu(s_i)$ is the type of situation s_i corresponding to some component $i \in \mathcal{C}$, $n=|\mathcal{C}|$. The sc class is equipped with the following $\leq_c$ relation:

$$sc' \leq_c sc \equiv \forall s'. s.s \in sc \wedge s' \in sc' \rightarrow s' \leq s. \quad (2)$$

$do(a_v, sc)$ denotes the sequence resulting from adding the action a_v to a type-compatible situation mentioned in sc .

Time. Time is part of the sorted domain of $TFSC$, as noted in Section 2; we extend the time selection function from actions to situations (see [16] for a different notation), and situation classes as follows:

$$\begin{aligned} time(S_0) &= t_0. \quad time(do(a, s)) = time(a). \\ time(s) &\leq time(s') \equiv \exists a \ a' \ s'' . s = do(a, s') \wedge \\ &\quad s' = do(a', s'') \wedge time(a) \leq time(a'). \end{aligned} \quad (3)$$

The time of a situation class sc , depends on the time of the situations in sc , so $time(sc) = t$ abbreviates:

$$\exists s \forall s'. s \in sc \wedge time(s) = t \wedge s' \in sc \wedge time(s') = t' \rightarrow t \geq t'. \quad (4)$$

Consistency of the extension. The above axioms concerning types, time and situation classes, are added to the foundational axioms in Σ and are conservative (i.e. obtained by extending the language), therefore the extended set (which we still denote with Σ) must be consistent.

Processes. Processes span the subtree of situations, over a single interval specified by a start and end action. They are implicitly typed by the actions process: for each process there are two actions, starting and ending the process, abbreviated by s_P , meaning *starts process P* and e_P , meaning *ends process P*. A process is denoted by a fluent $P(\vec{x}, t^-, s)$, (here t^- is its start time). Successor state axioms (SSP) for processes extend the set of SSA for fluents and are defined as follows:

$$\begin{aligned} P(\vec{x}, t^-, do(a, s)) &\equiv a = s_P(\vec{x}, t^-) \vee \\ &\quad P(\vec{x}, t^-, s) \wedge \forall t'. a \neq e_P(\vec{x}, t'). \end{aligned} \quad (5)$$

For example, moving towards θ , can be defined as:

$$\begin{aligned} move(\theta, t^-, do(a, s)) &\equiv a = s_{move}(\theta, t^-) \vee \\ &\quad move(\theta, t^-, s) \wedge \forall t'. a \neq e_{move}(\theta, t'). \end{aligned}$$

Action preconditions axioms define the conditions for the *start* and *end* actions to be executed. Let P be a process, we say that it is *linear* if it does not mention any other process in the right-hand side of the definition. The set $\mathcal{D}_{ssp}$ of successor state axioms for processes is *linear* if all processes mentioned in it are linear. Let $\mathcal{D}_{ssp}$ be a set of linear successor state axioms for processes, let the BAT be $\mathcal{D} \cup \mathcal{D}_{ssp} = \Sigma \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{AP} \cup \mathcal{D}_{ssp}$ then:

Lemma 1 $\mathcal{D} \cup \mathcal{D}_{ssp}$ is consistent iff $\mathcal{D}_{S_0}$ is.

Proof (sketch). The proof relies on the *relative consistency* of the SC. Furthermore, linearity of $\mathcal{D}_{ssp}$ ensures that there are no loops w.r.t. time, hence for each start and end action, as defined in the processes, it is not possible to introduce dependencies that could lead, for instance, to $s_P(\vec{x}, t) < s_Q(\vec{y}, t')$ and $s_P(\vec{x}, t) > s_Q(\vec{y}, t')$.

Intervals. Intervals mention temporal variables, which are always free, and get assigned only according to a specified set of behaviors, in so being very well suited for representing *flexible behaviors*. In this paragraph we show how embedding intervals in the SC leads to the construction of a temporal network. The following formula Ψ , specifies the conditions for a process P_i to hold on its induced timeline, from t_i^- to t_i^+ with arg. $\vec{x}$:

$$\begin{aligned} \Psi_i(\vec{x}, t_i^-, t_i^+, sc) &= \exists s_i s'_i s''_i . s_i \in sc \wedge \\ &\quad s_i \geq do(e_{P_i}(\vec{x}, t_i^+), s'_i) > do(s_{P_i}(\vec{x}, t_i^-), s''_i) \wedge \\ &\quad \neg \exists t \ s.s'_i \geq do(e_{P_i}(\vec{x}, t), s) \geq s_i \end{aligned} \quad (6)$$

Let $\mathbf{p}, \mathbf{m}, \mathbf{o}, \mathbf{d}, \mathbf{s}, \mathbf{f}, \mathbf{e}$ abbreviate the usual *precedes*, *meets*, *overlaps*, *during*, *starts*, *finishes*, *equals*, as defined in the *temporal intervals* literature, started in [1], with op ranging over all relations, and let γ_{op} denote:

$$\begin{aligned} \gamma_{\mathbf{p}} &= (t_i^+ < t_j^-); \quad \gamma_{\mathbf{m}} = (t_i^+ = t_j^-); \quad \gamma_{\mathbf{s}} = (t_i^- = t_j^-); \\ \gamma_{\mathbf{f}} &= (t_i^+ = t_j^+); \quad \gamma_{\mathbf{e}} = (t_i^+ = t_j^+ \wedge t_i^- = t_j^-); \\ \gamma_{\mathbf{o}} &= (t_i^- < t_j^- \wedge t_j^- < t_i^+ \wedge t_i^+ < t_j^+); \\ \gamma_{\mathbf{d}} &= (t_i^- > t_j^- \wedge t_j^- \geq t_i^+ \vee t_i^- \leq t_j^- \wedge t_j^- < t_i^+). \end{aligned}$$

We represent the interval relations between processes, given in situation-suppressed form (i.e not mentioning situations), and taking the “end-time” t^+ as argument, according to the following macro-definition, where the free variables mentioned in Ψ_i (see (6) above) are just hidden:

$$P_i(\vec{x}, t_i^-, t_i^+) \mathbf{op} P_j(\vec{x}, t_j^-, t_j^+) \triangleq \Psi_i \rightarrow [\Psi_j \wedge \gamma_{op}]. \quad (7)$$

Note that the metrical version (like in [10]) including durative relations can be analogously defined. Letting each operator in the macro being not commutative, inverse can be defined as $P_i \mathbf{op}^{-1} P_j = P_j \mathbf{op} P_i$.

We assume that the domain theory $\mathcal{D}_T$ is associated with temporal relations, as those given in (7), specifying the temporal interactions between processes as constraint patterns. For example, wanting to say that *going* has always to be preceded by *beingAt* we would associate to $\mathcal{D}_T$ the set $\{going([t_g^-, t_g^+]) \mathbf{p} beingAt([t_b^-, t_b^+])\}$. We call this set the *temporal compatibilities* abbreviated by T_c .

We denote $\mathbb{I}(\tau, sc)$ the set of the (7) interval relations associated with the situation class sc , where τ is the set of time variables, varying over the intervals $[t^+, t^-]$, related to the set of timelines sc . The $\mathbb{I}(\tau, sc)$ is obtained by abducing the time constraints on τ , according to a construction illustrated in the paragraphs below. For instance, for a suitable sc and $\tau = \{[t_1^-, t_1^+], [t_2^-, t_2^+], [t_3^-, t_3^+], [t_4^-, t_4^+]\}$ we can get the set:

$$\begin{aligned} \mathbb{I}(\tau, sc) = \{ &P_1(t_1^-, t_1^+) \mathbf{s} P_2(t_2^-, t_2^+), P_1(t_1^-, t_1^+) \mathbf{o} P_2(t_2^-, t_2^+), \\ &P_1(t_1^-, t_1^+) \mathbf{d} P_2(t_2^-, t_2^+), P_3(t_3^-, t_3^+) \mathbf{d} P_2(t_2^-, t_2^+), \\ &P_3(t_3^-, t_3^+) \mathbf{s} P_4(t_4^-, t_4^+), P_4(t_4^-, t_4^+) \mathbf{d} P_2(t_2^-, t_2^+)\}, \end{aligned}$$

whose constraints are depicted in the network of Fig. 2.

Temporal Constraint Network. The satisfiability problem we are concerned with is the following. Let $\mathcal{D}_T = \mathcal{D} \cup \mathcal{D}_{ssp} \cup \mathbb{I}(\tau, sc)$ be a basic theory of actions, extended with time specifications (see eq. (3) above), successor state axioms for processes, time interval constraints, in which time variables are free, and with an associated set of compatibilities T_c . We seek an assignment set for time variables, which is a feasible solution for the constraints, and s.t. a substitution of the

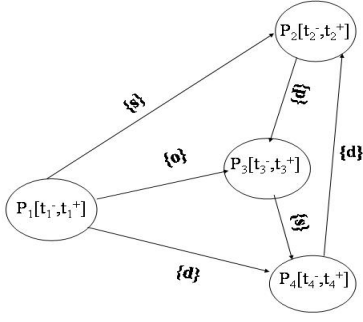


Figure 2: A TCN representing a set of interval constraints on processes P_1, P_2, P_3, P_4

free variables for this set induces an interpretation, which is a model of $\mathcal{D}_T \cup \mathbb{I}(\tau, sc)$. To find such an assignment we appeal to the concept of *general temporal constraint networks* (TCN) introduced in Meiri [13] following the one developed for discrete constraint networks [5]. A TCN involves variables denoting both intervals and points, together with unary and binary constraints. A TCN is associated with a *direct constraint graph* where each node is labeled by a temporal variable and labeled-directed edges represent the binary relations between them (e.g. p, e etc.). According to the underlying temporal algebra, TCNs express different forms of reasoning (Allen's interval algebra [1], the Point Algebra [22], metric point algebra [5], and so on) see in particular [2]. Let $\tau = \{X_1, \dots, X_n\} = \{[t_1^-, t_1^+], \dots, [t_n^-, t_n^+]\}$ be a set of time variables, denoting intervals (i.e. each variable X_i denotes an interval $[t_i^-, t_i^+]$), and let R be a set of binary relations; the assignment set $V = \{[s_1, e_1], \dots, [s_n, e_n] : s_i, e_i \in \mathbb{R}, s_i < e_i\}$ is called a solution set for R , if $\{X_1 = [s_1, e_1], \dots, X_n = [s_n, e_n]\}$ satisfies all the constraints defining R . The network is consistent if at least one solution set exists. In the sequel we shall identify a TCN with its labeled direct graph and denote with V a solution set. We also say that V satisfies the constraints.

Mapping SC to TCN. Let $h : \mathbb{I}(\tau, sc) \mapsto TCN$, $h(\mathbb{I}(\tau, sc))$ is a temporal network Net , with cardinality m , specified by the nodes labeled by the defined processes $\{P_i, P_j | P_i \text{ op } P_j\}$, mentioned in $\mathbb{I}(\tau, sc)$.

By the above definitions the network Net is consistent if there exists an assignment V to the temporal variables τ , which is a feasible solution for $\mathbb{I}(\tau, sc)$. For example the set $V = \{[10, 160], [10, 70], [75, 95], [95, 150]\}$ is a solution set for the variables $\tau = \{[t_1^-, t_1^+], [t_2^-, t_2^+], [t_3^-, t_3^+], [t_4^-, t_4^+]\}$, of the network depicted in Figure 2, given the relations $\{m, p, d, s, f\}$. Note that, in the macro definition (7), there are two parts, the left-hand component is the *definiendum* $P_i(\bar{x}, t_i^-, t_i^+) \text{ op } P_j(\bar{x}, t_j^-, t_j^+)$, which is mapped into a temporal network, while the right-hand component is the *definiens* $\Psi_i \rightarrow (\Psi_j \wedge \gamma_{op})[\tau]$, which is interpreted into a structure of the SC, where the assignments for the free temporal variables τ are obtained by the TCN. A semantic correspondence, between the network and the formulae of SC, can be established as follows. Let $\mathcal{M} = \langle D, I \rangle$ be a struc-

ture of SC (where D is a sorted domain and I an interpretation) we extend $\mathcal{M}$ with the above defined mapping h , i.e. $\mathcal{M}_h = \langle D, I, h \rangle$. Then $\mathcal{M}_h$ is a model for $\mathcal{D} \cup \mathbb{I}(\tau, sc)$, iff there is a consistent temporal network $h(\mathbb{I}(\tau, sc))$, under some assignment V to the free time-variables, satisfying $\mathbb{I}(\tau, sc)$. Given the above definitions we can state the following:

Lemma 2 Let $\mathcal{D}_T$ be a consistent action theory. $\mathcal{D}_T \cup \mathbb{I}(\tau, sc)$ is consistent iff there exists a consistent temporal constraint network $h(\mathbb{I}(\tau, sc))$, under the assignment V , such that $\mathcal{M}_h, V \models \bigwedge \mathcal{D} \wedge \Psi[\tau]$.

Proof(sketch). Let $h(\mathbb{I}(\tau, sc))$ be a consistent temporal network, under the assignment V . Consider the formula $\Psi[\tau] = \bigwedge_{i,j \leq m} \Psi_i \rightarrow (\Psi_j \wedge \gamma_{op_{ij}})$, with τ the set of free time-variables in Ψ , and we let $\mathcal{M}$ be a model for $\mathcal{D}_T$. $\mathcal{M}$ can be extended to a model for all the $\gamma_{op_{ij}}$ according to V , as follows. For each process P_i build a chain of situations of the kind $\Gamma_i = \{do(s_{P_i}(t_i'), \sigma') \leq \sigma \leq do(e_{P_i}(t_i''), \sigma'') \leq \sigma_i\}$, with σ a sequence such that e_{P_i} is not mentioned in σ , and the free-temporal variables are from τ according to the P_i timeline in sc . Extend the structure $\mathcal{M}$ to one for $\mathcal{D}_T \cup \bigcup_i \Gamma_i$ by choosing from V a suitable assignment to the free variables τ , appearing in each Γ_i , and according to the constraints in the $\gamma_{op_{ij}}$, corresponding to those in $\mathbb{I}(\tau, sc)$, and such that each sequence is satisfied. This is always possible, because each process is made true of a situation by a start action, and false by an end action, by the sequence construction, on a timeline. Then V is an assignment to τ s.t. $(\mathcal{M}_h, V)$ is a model for $\mathcal{D}_T \cup \bigcup_i \Gamma_i[\tau]$. Finally it is enough to show that any model for $\mathcal{D}_T \cup \bigcup_i \Gamma_i[\tau]$, is a model for $\Psi[\tau]$. $\square$

By Lemma 1 $\mathcal{D}_T \cup \mathbb{I}(\tau, sc)$ is consistent iff $\mathcal{D}_{S_0} \cup \mathbb{I}(\tau, sc)$ is consistent. Logical implication can be now defined as follows. Let $\mathcal{M}_h = \langle D, I, h \rangle$, and $\beta(\tau)$ be a formula with all its free temporal variables among τ :

$$\begin{aligned} \mathcal{D}_T \cup \mathbb{I}(\tau, sc) \models \beta(\tau) \text{ iff for any } h, \text{ for any } V, \\ \mathcal{M}_h, V \models \mathcal{D}_T \cup \mathbb{I}(\tau, sc) \text{ implies } \mathcal{M}_h, V \models \beta(\tau). \end{aligned} \quad (8)$$

TCN construction. The TCN construction involves the following inference mechanisms: i. abduce $\mathbb{I}(\tau, sc)$ to obtain the temporal network topology $h(\mathbb{I}(\tau, sc))$; ii. implement the temporal constraints into situations, according to the network. We show the inference process through an example. Let $T_c = \{P_1 \text{ m } P_2\}$, be the compatibilities specified in the domain theory. Let $sc = do([s_{P_1}(t_1^-), e_{P_1}(t_1^+), s_{P_2}(t_2^-), e_{P_2}(t_2^+), s_{P_1}(t_1'^-), e_{P_1}(t_1'^+)], S_0)$, then the abduced constraints, according to T_c , sc and τ , are:

$$\mathbb{I}(\tau, sc) = \{P_1[t_1^-, t_1^+] \text{ m } P_2[t_2^-, t_2^+], P_1[t_1'^-, t_1'^+] \text{ m } P_2[t_2'^-, t_2'^+]\} \quad (9)$$

From the above constraints a three nodes network $h(\mathbb{I}(\tau, sc))$ would be constructed. The time range of the solutions to the network – given the free temporal variables and the situation class sc – is a *flexible situation class*, which is a triple of the kind $\langle sc, \tau, i \rangle$, where $i(\tau)$ is the constraint over the free-variables, e.g. $i(\tau) = 7 \leq t_1^- \wedge t_2^+ > 10$.

Progression. We want to determine if, according to the timing and advancements of the current situation class, we can forget about past scheduled processes and think of future resources allocation. This is the well known progression problem [19] in the SC. We face here a simplified version of the

progression problem, and with regressable formulas, which is all we need. Let us state the problem as follows. Given a sentence φ , one needs first to determine the set of situations in sc to which it can be progressed, given the timelines, and a candidate interval $t = [t^-, t^+]$. To this end one has to order the situation terms in φ according to the relation $\leq_{sc}$ introduced in (2), and get the smallest situations set, w.r.t. $\leq_{sc}$, mentioning t ; let it be min . We now consider the domain theory $\mathcal{D}_T^{min}$, which is equal to $\mathcal{D}_T$ but with $\mathcal{D}_{S_0}$ replaced by a set $W^{min} = \{\varphi | \mathcal{D}_T \models \varphi\}$, where φ mentions only ground situations σ , with $\{\sigma_i\} =_{sc} min$, but no free variables of sort time. Let $\Phi(t_1, \dots, t_n)$ be a SC formula mentioning situations $\{\sigma_i\} \geq_{sc} min$, which are ground but for possible free variable of sort time all among $t_1, \dots, t_n$. Then

Lemma 3 $\mathcal{D}_T \cup \mathbb{I}(\tau, sc) \models \bigwedge (\mathcal{D}_T^{min} \cup \mathbb{I}(\tau, sc) \cup W^{min})$.

Theorem 1 $\mathcal{D}_T \cup \mathbb{I}(\tau, sc) \models \Phi(t_1, \dots, t_n)$ iff $\mathcal{D}_T^{min} \cup W^{min} \cup \mathbb{I}(\tau, sc) \models \Phi(t_1, \dots, t_n)$

Proof. (sketch) One direction follows by cut. For the other direction, suppose $\mathcal{D}_T \cup \mathbb{I}(\tau, sc) \models \Phi(t_1, \dots, t_n)$, let us regress $\Phi(t_1, \dots, t_n)$ along the different timelines, this can be done by separating Φ into a DNF, and considering each conjunct separately, we regress it with respect to min , and let G be the regressed formula – note that no change is needed in regression to account for regressed sentences as time variables will be kept as they are, without turning to quantified variables through successor state axioms (as they are universally quantified in linear $\mathcal{D}_{ssp}$). Then by definition of W^{min} , $G \in W^{min}$ hence, by the previous lemma, the claim follows.

4 Flexible High Level Programming in Golog

Golog Syntax. Given the extended action theory presented above, the following constructs inductively build Golog programs:

1. *Primitive action:* α .
2. *Nondeterministic choice:* $\alpha | \beta$. Do α or β .
3. *Test action:* $\phi?$. Test if ϕ is true in the current sit. class.
4. *Nondet. arg. choice:* choose $\vec{x}$ for $prog(\vec{x})$.
5. *Action sequence:* $p_1; p_2$. Do p_1 followed by p_2 .
6. *Partial order act. choice:* $p_1 \prec p_2$. Do p_1 before p_2 .
7. *Parallel execution:* $p_1 \parallel p_2$. Do p_1 concurrently with p_2 .
8. *Conditionals:* **if** ϕ **then** p_1 **else** p_2 .
9. *While loops:* **while** ϕ **do** p_1 .
10. *Procedures, including recursion.*

Golog Semantics. The macro $Do(p, sc, sc', h)$ gives the semantics for the above constructs; where p is a program, sc and sc' are situation classes, and h specifies the finite horizon.

- Null program:

$$Do(Nil, sc, sc', h) \triangleq time(sc) \leq h \wedge \forall s. [s \in sc \equiv s \in sc']$$

- Primitive first program action with horizon:

$$Do(a_i; prog, sc, sc', h) \triangleq \exists s_i. s_i \in sc \wedge Poss(a_i, s_i) \wedge time(s_i) \leq h \wedge [time(a_i) \leq h \wedge Do(prog, do(a_i, sc), sc', h) \vee time(a_i) > h \wedge Do(prog, sc, sc', h)]$$

Here, if the first program action is applicable to $s_i \in sc$, and a_i can be scheduled after the horizon then it is neglected (i.e. each action, which can be started after the horizon can be postponed).

- Partial order action choice:

$$Do(prog_1 \prec prog_2, sc, sc', h) \triangleq Do(prog_1 : prog_2, sc, sc', h) \vee \exists a. select(a, sc) \wedge Do(prog_1 \prec a \prec prog_2, sc, sc', h)$$

Here, either the second program can be directly executed, or it is possible to insert a primitive action a , selected by a suitable fluent predicate $select(a, s)$ representing the selection criterion (set to true if no selection holds).

- Parallel execution:

$$Do(prog_1 \parallel prog_2, sc, sc', h) \triangleq Do(prog_1, sc, sc', h) \wedge Do(prog_2, sc, sc', h)$$

- Test action:

$$Do(\phi?; prog, sc, sc', h) \triangleq \phi[sc] \wedge Do(prog, sc, sc', h)$$

Here $\phi[sc]$ stands for generalization of the standard $\phi[s]$ in the SC extended to situation classes, e.g. $P_1[sc] \wedge P_2[sc]$ is for $P_1(s_1) \wedge P_2(s_2)$ with $s_1, s_2 \in sc$, i.e. each fluent is evaluated w.r.t. its specific timeline.

- The semantics of nondet. action choice, nondet. argument selection, conditionals, while loops, and procedures is defined in the usual way.

Flexible Temporal Behaviors in Golog. The CBI planning problem $(\mathcal{M}_{cbi}, P_c)$ introduced in Section 2.2 can be easily coded and solved in the *TFSC* framework. Given a $\mathcal{D}_T$ representing the timelines and processes in $\mathcal{M}_{cbi}$, a candidate plan P_c can be encoded by a Golog program $prog_c$. This is possible once we introduce, for each interval constraint for $p_{i,j}(\vec{r})$ in P_c , a Golog procedure of the kind:

$$\text{proc}(\pi_{i,j}, (\psi_i^j(\vec{r}, t^-, t^+) \wedge \gamma(t^-, t^+))?),$$

where ψ_i^j is the macro (6) associated with the process P_i^j (i.e. i -th process in the j -th timeline), and γ any temporal constraint. For example, $go(d, e)$ ends in $[6, 10]$ can be represented as

$$\text{proc}(\pi_{1,1}, (\psi_{go}^{nav}(d, e, t^-, t^+) \wedge 6 \leq t^+ \leq 10)?).$$

Given the $\pi_{i,j}$ procedures, a partial specification of a single timeline $\mathcal{T}_j$ can be easily defined using the $\prec$ operator:

$$\text{proc}(plan_T_j, \pi_{0,j} \prec \pi_{1,j} \prec \pi_{2,j} \prec \dots \prec \pi_{k,j}),$$

Given a set of timelines $\{\mathcal{T}_i\}$, a candidate plan P_c can be represented as a parallel execution of the $plan_T_i$:

$$\text{proc}(prog_c, plan_T_1 : Nil \parallel \dots \parallel plan_T_k : Nil).$$

Since a CBI complete plan P is associated with a fully supported set of timelines $\{\mathcal{T}_i\}$ and a set of constraints $i(\tau)$ (see Section 2.2), we can introduce a mapping g transforming a CBI plan P into a flexible situation $\langle sc, \tau, i \rangle$. The following holds.

Proposition 1 *Given a CBI planning problem $(\mathcal{M}_{cbi}, P_c)$, where P_c is a candidate plan [10], for any complete plan P of $(\mathcal{M}_{cbi}, P_c)$, maximally flexible in the time variables τ , there exists a $\mathcal{D}_T$, a Golog program $prog_c$, where*

$$\mathcal{D}_T \cup \mathbb{I}(\tau, sc) \models Do(prog_c, ini, sc[\tau], h).$$

with $\langle sc, \tau, i \rangle$ a flexible situation, and such that:

$$g(P) = \langle sc, \tau, i \rangle$$

The proof is omitted.

Example. We assume an autonomous rover, which has to explore the environment and perform observations. While the robot is moving the pant-tilt unit must be pointing ahead *ptIdle*. To point to a location x the rover must be stopped there, $at(x)$, while the pan-tilt scans in direction z : $ptScan(x, z)$. Hence, we consider two components: *pant-tilt*, *navigation*. Each component has a set of processes: $J_{pt} = \{ptIdle, ptScan(z, x)\}$; $J_{nav} = \{at(x), go(x, y)\}$. Each of these is to be encoded in the $\mathcal{D}_T$ as in (5), e.g. $ptScan(t, z, x, do(a, s)) \equiv a = s_{pts}(z, x, t) \vee ptScan(t, z, x, s) \wedge \neg \exists t'. a = e_{pts}(z, x, t')$. The temporal compatibilities T_c among the activities are defined by a set of temporal constraints patterns, defined by macros (7), e.g.:

$$\begin{aligned} go(x, y, t_1, t_2) \text{ m } at(y, t_3, t_4); at(x, t_1, t_2) \text{ m } go(x, y, t_3, t_4); \\ go(x, y, t_1, t_2) \text{ d } ptScan(z, x, t_3, t_4); \\ ptScan(z, x, t_1, t_2) \text{ d } at(x, t_3, t_4). \end{aligned}$$

Consider a partial specification for the rover behavior: from time 0 to 3 it must remain $at(p_1)$, and at time 0 the pant-tilt is *ptIdle*; the rover mission is to observe p_3 , in direction θ before 30 and after 20, and be back at p_1 before 50. This *candidate plan*, can be encoded in the following Golog program:

$$\begin{aligned} \text{proc}(\text{mission}, \\ \psi_{at}^{nav}(p_1, 0, 3)? \prec (\psi_{at}^{nav}(p_1, t_1^1, t_2^1) \wedge t_2^1 \leq 50)? : Nil \parallel \\ \psi_{pti}^{pt}(0, t_1^2)? \prec (\psi_{pts}^{pt}(\theta, p_3, t_2^2, t_3^2) \wedge [t_2^2, t_3^2] \subseteq [20, 30])? \prec Nil). \end{aligned}$$

Given the horizon $h = 50$, and an initial situation class *ini*, a possible complete plan is a flexible situation $\langle sc, \tau, i \rangle$, with $sc[\tau] = \{\sigma_{nav}[\tau_1], \sigma_{pt}[\tau_2]\}$, s.t. $\mathcal{D}_T \cup \mathbb{I}(\tau, sc) \models Do(\text{mission}, ini, sc[\tau], 50)$, e.g.

$$\begin{aligned} \sigma_{nav} = do([s_{at}(p_1, 0), e_{at}(p_1, 3), s_{go}(p_3, t_1^1), e_{go}(p_3, t_2^1), \\ s_{at}(p_3, t_3^1), e_{at}(p_3, t_4^1), s_{go}(p_1, t_5^1), e_{go}(p_1, t_6^1)], S_0); \\ \sigma_{pt} = do([s_{pti}(0), e_{pti}(t_1^2), s_{pts}(t_2^2), e_{pts}(t_3^2), s_{pti}(t_4^2), e_{pti}(t_5^2), s], S_0), \end{aligned}$$

and $i(\tau)$ is the minimal set of constraints solving the $h(\mathbb{I}(sc, \tau))$ temporal network, i.e. $\{t_2^1 = t_3^1 < t_4^1 = t_5^1 < t_6^1 < 50; t_1^2 = t_2^2 < t_3^2 = t_4^2 < t_5^2; t_2^2 > t_3^2, t_3^2 < t_4^2\}$.

Implementation. We provided a constraint logic programming (CLP) [9] implementation of the Golog interpreter. Since the Golog interpreter is to generate flexible temporal plans, it must be endowed with a constraint problem solver. Analogous to [17] we rely on a logic programming language with a built-in solver for linear constraints over the reals (CLP($\mathcal{R}$)). We appeal to the ECRC Common Logic Programming System ECLIPSE 5.7. Currently, we are deploying the Golog interpreter as the control engine of a monitoring systems for robotics applications.

References

- [1] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.
- [2] Federico Barber. Reasoning on interval and point-based disjunctive metric constraints in temporal contexts. *J. Artif. Intell. Res. (JAIR)*, 12:35–86, 2000.
- [3] Giuseppe de Giacomo, Yves Lesperance, and Hector J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artif. Intell.*, 121(1-2):109–169, 2000.
- [4] A.K. Jonsson D.E. Smith, J. Frank. Bridging the gap between planning and scheduling. *Knowledge Engineering Review*, 15(1), 2000.
- [5] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artif. Intell.*, 49(1-3):61–95, 1991.
- [6] Alfredo Gabaldon. Programming hierarchical task networks in the situation calculus. In *AIPS'02*, 2002.
- [7] Malik Ghallab and Herv Laruelle. Representation and control in ixtet, a temporal planner. In *AIPS 1994*, pages 61–67.
- [8] H. Grosskreutz and G. Lakemeyer. ccgolog – a logical language dealing with continuous change. *Logic Journal of the IGPL*, 11(2):179–221, 2003.
- [9] Joxan Jaffar and Michael J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–581, 1994.
- [10] Ari K. Jonsson, Paul H. Morris, Nicola Muscettola, Kanna Rajan, and Benjamin D. Smith. Planning in interplanetary space: Theory and practice. In *Artificial Intelligence Planning Systems*, pages 177–186, 2000.
- [11] J. Kvarnstrom, P. Doherty, and P. Haslum. Extending talplanner with concurrency and resources. In *Proc. ECAI-00*, pages 501–505, 2000.
- [12] 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, pp. 410–417.
- [13] Itay Meiri. Combining qualitative and quantitative constraints in temporal reasoning. *Artif. Intell.*, 87(1-2):343–385, 1996.
- [14] Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian C. Williams. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1-2):5–47, 1998.
- [15] J.A. Pinto and R. Reiter. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, 14(2-4):251–268, September 1995.
- [16] Javier Pinto. Occurrences and narratives as constraints in the branching structure of the situation calculus. *Journal of Logic and Computation*, 8(6):777–808, 1998.
- [17] Fiora Pirri and Raymond Reiter. Planning with natural actions in the situation calculus. pages 213–231, 2000.
- [18] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Proceedings of KR'96*, pages 2–13, 1996.
- [19] Raymond Reiter. *Knowledge in action : logical foundations for specifying and implementing dynamical systems*. MIT Press, 2001.
- [20] Tran Cao Son, Chitta Baral, and Le-Chi Tuan. Adding time and intervals to procedural and hierarchical control specifications. In *AAAI 2004*, pages 92–97, 2004.
- [21] Austin Tate. I-n-ova and i-n-ca - representing plans and other synthesised artifacts as a set of constraints.
- [22] Marc B. Vilain and Henry A. Kautz. Constraint propagation algorithms for temporal reasoning. In *AAAI*, pages 377–382, 1986.
- [23] B. Williams, M. Ingham, S. Chung, P. Elliott, M. Hofbaur, and G. Sullivan. Model-based programming of fault-aware systems. *AI Magazine*, Winter 2003.

Cohesion, coupling and the meta-theory of actions

Andreas Herzig and Ivan Varzinczak*

IRIT – Université Paul Sabatier

118 route de Narbonne – 31062 Toulouse Cedex 04 France

e-mail: {herzig,ivan}@irit.fr

Abstract

In this work we recast some design principles commonly used in software engineering and adapt them to the design and analysis of domain descriptions in reasoning about actions. We show how the informal requirements of cohesion and coupling can be turned into consistency tests of several different arrangements of modules. This gives us new criteria for domain description evaluation and clarifies the link between software and knowledge engineering in what concerns the meta-theory of actions.

1 Introduction

Among the principles of the object-oriented paradigm are the following:

1. Work with modules (or components, functions, etc.).
2. Minimize interactions between such modules.
3. Organize the modules into well-defined layers to help minimize interactions. The goal is to have components of one layer using only components from immediate neighbors, wherever possible.
4. Anticipate what kind of extensions or modifications might be made in the future, and support this at design time so that one can extend the system with minimal disruption later.

There seems to be an agreement that such principles for object-oriented programming or design are the same as for knowledge representation. To witness, the design of domain descriptions in reasoning about actions has much more in common with software engineering than one might think: in the same way as for software projects, one can talk about consistency, evolution and correctness of domain descriptions.

All the principles above can be applied to the design of domain descriptions, too. We argue that a good domain description should be one whose consistency check and maintenance complexities are minimized, so that any further modification is localized, with a bounded scope.

* Supported by a fellowship from the government of the FEDERATIVE REPUBLIC OF BRAZIL. Grant: CAPES BEX 1389/01-7.

With this in mind, one can see the specification of domain descriptions as a task similar to project development in software engineering: Item 4 above is what has been called *elaboration tolerance* [McCarthy, 1988]. In this way a representation is elaboration tolerant to the extent that the effort required to add new information (a new action or effect) to the representation is proportional to the complexity of that information [Shanahan, 1997]. Items 1, 2 and 3 reflect the concept of *modularity*, which means that different modules have no elements in common. Such a notion of modularity is going to lead us along the present work.

This paper is an elaboration of the results we have presented in [Herzig and Varzinczak, 2004]. Here we pursue the following plan: in Section 2 we recall some important concepts from software engineering; after discussing the ontology of dynamic domains (Section 3) we apply the concepts of Section 2 to the design of domain descriptions (Sections 4 and 5) making a step towards formal criteria for domain description evaluation. In Section 6 we present the main results that follow from our approach, and before concluding we address related work found in the literature on this subject.

2 Some principles of software engineering

One of the first steps in software development is that of *abstraction*. Abstraction consists mainly in rendering lower-level details invisible to upper levels in order to facilitate the understanding and design of complex systems. As an example, a specification of a data or knowledge base query does not need to take into account the algorithmic process that will be carried out in order to answer the query.

In parallel to abstraction, one of the most important guidelines in project design is that of *modularity*: dividing the software into modules, based on their functionality or on the similarity of the information they handle. This means that instead of having a “jack of all trades” program, it is preferable to split it up into specialized subprograms. For instance, a program made of a module for querying a database and a module for checking its integrity is more modular than a single module that does these two tasks at the same time.

Among the major benefits of modular systems are reusability, scalability and better management of complexity.

There is more than one way to split up a program. One of the most used techniques is that of forcing functional in-

dependence of its modules. One ensures functional independence in a project by defining modules with only one purpose and “aversion” to excessive interaction with other modules [Pressman, 1992].

Among the criteria commonly used for evaluating functional independence of modules (and thus how modular a piece of software is) are the informal notions of cohesion and coupling.

Cohesion is how closely related pieces of a single component are to each other. A module is cohesive when at the high level of abstraction it does only a single task. The more a module is focused on a precise goal the more it is cohesive.

A highly cohesive module will be simpler to understand, having to do only a single task, while a lowly cohesive module, performing so many tasks, will be difficult to understand.

It is difficult to reuse a task-overloaded module, while a highly cohesive module is simpler to reuse and to extend.

Coupling is the interdependency between a method and the environment (other methods, objects and classes). Low coupling means to keep dependencies (communication, information sharing) between components at a minimum.

A design that has low coupling is more amenable to change, since it reduces the probability of changes cascading and affecting a larger part of the system.

Unanimously in object-oriented development, the best way to design a software is to have low coupling and high cohesion. We sum this up in two informal design principles:

- P1. **Maximal cohesion:** *Every module should be conceived in such a way that it is maximally cohesive.*
- P2. **Minimal coupling:** *All modules should be conceived in such a way that they minimize coupling.*

3 Natural modules in domain descriptions

Like in object-oriented programming, in describing a domain different entities should be separated in different modules. Moreover, each module should be conceived in such a way that it has no direct access to the contents of the others. In reasoning about actions, accessing a module means using it to perform reasoning tasks like prediction, postdiction, planning and others. This amounts to using its logical formulas in inferences. In this section we establish the ontology of domain descriptions and present the way we arrange in different modules the axioms commonly used to describe them.

Every domain description contains a representation of action effects. We call *effect laws* formulas relating an action to its effects. Statements of conditions under which an action cannot be executed are called *inexecutability laws*. *Executability laws* in turn stipulate the context where an action is guaranteed to be executable. Finally, *state constraints* are formulas that do not mention actions and express constraints that must hold in every possible state. These are our four ingredients that we introduce more formally in the sequel.

If we think of a domain description as a software application, we can imagine its organization in an object-oriented view and attempt to have a kind of class diagram for it. This

is illustrated by Figure 1, where we can see the relationship among the different types of entities.

A domain description consists of a description of effects of actions, their non-effects, executabilities, inexecutabilities and also state constraints that do not depend on any particular action.

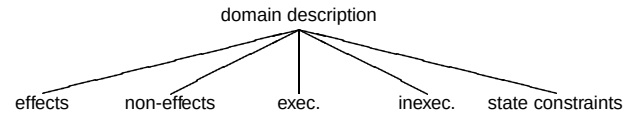


Figure 1: “Class diagram” of modules in designing domain descriptions. Edges represent *has-a* relations.

Among the effects of actions, we can distinguish *direct effects* and *indirect effects* (ramifications).

Non-effects of actions are related with the frame problem [McCarthy and Hayes, 1969], and indirect effects with the ramification problem [Finger, 1987]. In this work we abstract from these problems and assume we have a consequence relation powerful enough to derive the intended conclusions. We suppose given a ‘doped’ consequence relation $\approx$, which encapsulates some traditional approach in the literature (e.g., [Schubert, 1990; Lin, 1995; McCain and Turner, 1995]), with which all intended frame axioms and indirect effects can be derived, and we use it henceforth. As examples we have

$$\{loaded(s)\} \approx loaded(do(wait, s))$$

(i.e., waiting does not change the status of *loaded*) and

$$\left\{ \begin{array}{l} walking(S_0), \\ walking(s) \rightarrow alive(s), \\ \neg alive(do(shoot, s)) \end{array} \right\} \approx \neg walking(do(shoot, S_0))$$

Hence shooting has the indirect effect that the victim will no longer be walking.

We use small letters to denote variables, and capital letters to denote constant symbols. Free variables are supposed to be universally quantified.

To sum it up, our main concern here will be with direct effects (henceforth effects), inexecutabilities, executabilities and state constraints. We introduce this in what follows.

Effect laws Logical frameworks for reasoning about actions contain expressions linking actions and their effects. We suppose that such effects might be conditional, and thus get a third component of such axioms. An *effect law* for action *a* is of the form

$$Poss(a, s) \rightarrow (\Phi(s) \rightarrow \Psi(do(a, s)))$$

where $\Phi(s)$ is a *simple state formula about situation s*, and $\Psi(do(a, s))$ is a *simple state formula about situation do(a, s)*.

A simple state formula about a situation term *t* contains no *Poss*-predicate and no situation terms other than *t* [Lin, 1995].

An example of effect law is

$$Poss(shoot, s) \rightarrow (loaded(s) \rightarrow \neg alive(do(shoot, s))),$$

saying that whenever *shoot* is executable and the gun is loaded then after shooting the turkey is dead. Another one is $Poss(tease, s) \rightarrow walking(do(tease, s))$: the result of teasing is that the turkey starts walking.

Inexecutability laws The design of domain descriptions must also provide a way to express qualifications of actions, i.e., conditions under which an action cannot be executed at all. An *inexecutability law* for action a is of the form

$$\Phi(s) \rightarrow \neg \text{Poss}(a, s)$$

where $\Phi(s)$ is a simple state formula about s .

For example, $\neg \text{HasGun}(s) \rightarrow \neg \text{Poss}(\text{shoot}, s)$ states that *shoot* cannot be executed if the agent has no gun.

State constraints (alias domain constraints) Frameworks allowing for indirect effects make use of formulas that link invariant propositions about the world. Such formulas characterize the set of possible states. A *state constraint* is a simple state formula about the situation term s that is consistent. An example is $\text{walking}(s) \rightarrow \text{alive}(s)$, saying that if a turkey is walking, then it must be alive [Thielscher, 1995].

Executability laws With only state constraints and effect laws one cannot guarantee that action *shoot* is executable if the agent has a gun. An *executability law* for action a is of the form $\Phi(s) \rightarrow \text{Poss}(a, s)$, where $\Phi(s)$ is a simple state formula about s . For instance $\text{HasGun}(s) \rightarrow \text{Poss}(\text{shoot}, s)$ says that shooting can be executed whenever the agent has a gun, and $\text{Poss}(\text{tease}, s)$ that the turkey can always be teased.

Whereas all the extant approaches in the literature that allow for indirect effects of actions contain state constraints and effect laws, the status of executability laws is less consensual: some authors [Schubert, 1990; Doherty *et al.*, 1996; McCain and Turner, 1995; Thielscher, 1995] more or less tacitly consider that executability laws should not be made explicit but rather inferred by the reasoning mechanism. Others [Lin, 1995; Zhang *et al.*, 2002] have executability laws as first class objects one can reason about.

We nevertheless would like to point out that maximizing executability, as usually done in the literature, is not always intuitive: suppose we know that if we have the ignition key, the tank is full, . . . , and the battery tension is beyond 10V, then the car (necessarily) will start. Suppose we also know that if the tension is below 8V, then the car will not start. What should we conclude in situations where we know that the tension is 9V? Maximizing executabilities makes us infer that it will start, but such reasoning is not what we want if we would like to be sure that all possible executions lead to the goal.

It seems a matter of debate whether one can always do without executabilities. We think that in several domains one wants to explicitly state under which conditions a given action is guaranteed to be executable, such as that a robot should never get stuck and should always be able to execute a move action. In any case, allowing for executability laws gives us more flexibility and expressive power.

Domain descriptions Given the four types of entities defined above, we arrange them in the following way: for a given action a , $\mathbf{Eff}_a$ is the set of its effect laws, $\mathbf{Inex}_a$ is the set of its inexecutability laws, and $\mathbf{Exe}_a$ is the set of its executability laws. $\mathbf{Stat}$ denotes the set of all state constraints of a given domain. Thus, $\mathbf{Eff}_a$, $\mathbf{Exe}_a$ and $\mathbf{Inex}_a$, for each action a , and $\mathbf{Stat}$ are the natural modules we consider here in designing a domain description.

For parsimony's sake, we define $\mathbf{Eff} = \bigcup \mathbf{Eff}_a$, $\mathbf{Inex} = \bigcup \mathbf{Inex}_a$, and $\mathbf{Exe} = \bigcup \mathbf{Exe}_a$. We suppose all these sets are consistent.

A *domain description* $\mathbf{D}$ is a tuple of the form $\langle \mathbf{Eff}, \mathbf{Inex}, \mathbf{Exe}, \mathbf{Stat} \rangle$.

Once the information contained in a module is not mixed with others', it can be expected that undesirable side effects due to further modifications are less likely to propagate to other parts of the domain description. The same thing can be obtained for the consistency check if beyond of being separated the modules are designed in such a way that their interaction is minimized. This is what we address in this section.

As we have seen, in software engineering functional independence is evaluated by means of two criteria: cohesion, a criterion for evaluating the relative functional strength of a module, and coupling, an assessment of relative interdependence among different modules. Both these notions are quite informal, even in software engineering, and cannot be measured in an objective way.

Here we explore these concepts when applied to domain descriptions and show how the informal requirements of software engineering can be turned into tests of consistency of several different arrangements of modules.

4 Cohesion

Normally cohesion comes with modularization, and its evaluation depends mainly on the entities that one takes into account when describing a domain.

In talking about sets of logical formulas we take cohesion as how simple or well-defined a logical module is, considering the different types of formulas that can be derived from it. We thus refine our first design principle:

P1'. *The less types of laws a given module entails alone, the more cohesive it is.*

As an example consider the following module:

$$\left\{ \begin{array}{l} \neg \text{HasGun}(s) \rightarrow \neg \text{Poss}(\text{shoot}, s), \\ \text{HasGun}(s) \rightarrow \text{Poss}(\text{shoot}, s) \end{array} \right\}$$

From such a set alone one can derive both $\neg \text{HasGun}(s) \rightarrow \neg \text{Poss}(\text{shoot}, s)$ and $\text{HasGun}(s) \rightarrow \text{Poss}(\text{shoot}, s)$, which are formulas of two different kinds. In this case we say that such a set is a lowly cohesive module, for alone it functions to derive executabilities and inexecutabilities. A better approach would be to decompose such a module into the following ones:

$$\mathbf{Inex}_{\text{shoot}} = \{ \neg \text{HasGun}(s) \rightarrow \neg \text{Poss}(\text{shoot}, s) \}$$

$$\mathbf{Exe}_{\text{shoot}} = \{ \text{HasGun}(s) \rightarrow \text{Poss}(\text{shoot}, s) \}$$

Total cohesion is not always easy to achieve. Suppose, for instance, a hypothetical situation in which we reason about the effects of drinking a cup of coffee:

$$\mathbf{Eff}_{\text{drink}} = \left\{ \begin{array}{l} \text{Poss}(\text{drink}, s) \rightarrow \\ (\text{sugar}(s) \rightarrow \text{happy}(\text{do}(\text{drink}, s))), \\ \text{Poss}(\text{drink}, s) \rightarrow \\ (\text{salt}(s) \rightarrow \neg \text{happy}(\text{do}(\text{drink}, s))) \end{array} \right\}$$

Then, $\mathbf{Eff}_{\text{drink}}$ entails $(\text{sugar}(s) \wedge \text{salt}(s)) \rightarrow \neg \text{Poss}(\text{drink}, s)$. This means that from $\mathbf{Eff}_{\text{drink}}$ alone we do not get only effect

laws but also inexecutability laws. Therefore $\mathbf{Eff}_{drink}$ is not as cohesive as one might have expected.

One step towards augmenting cohesion of a module of effect laws can be by completely specifying the preconditions of effects of actions. For example, the weaker effect laws

$$\mathbf{Eff}'_{drink} = \left\{ \begin{array}{l} \text{Poss}(\text{drink}, s) \rightarrow \\ (\text{sugar}(s) \wedge \neg \text{salt}(s)) \rightarrow \text{happy}(\text{do}(\text{drink}, s)), \\ \text{Poss}(\text{drink}, s) \rightarrow \\ (\text{salt}(s) \wedge \neg \text{sugar}(s)) \rightarrow \neg \text{happy}(\text{do}(\text{drink}, s)) \end{array} \right\}$$

guarantee a higher cohesion of module $\mathbf{Eff}'_{drink}$ in comparison to that of $\mathbf{Eff}_{drink}$.

By the definition of $\mathbf{Stat}$, it is easy to see that from the state constraints we can derive formulas of any type, so $\mathbf{Stat}$ is by nature a lowly cohesive module.

We are thus interested in refining even more our principle of high cohesion P1' by the following ones:

- P1'-1. If $\mathbf{Inex} \approx \Phi(s)$, then $\emptyset \approx \Phi(s)$.
- P1'-2. If $\mathbf{Inex} \approx \Phi(s) \rightarrow \text{Poss}(a, s)$, then $\emptyset \approx \Phi(s) \rightarrow \text{Poss}(a, s)$.
- P1'-3. If $\mathbf{Exe} \approx \Phi(s)$, then $\emptyset \approx \Phi(s)$.
- P1'-4. If $\mathbf{Exe} \approx \Phi(s) \rightarrow \neg \text{Poss}(a, s)$, then $\emptyset \approx \Phi(s) \rightarrow \neg \text{Poss}(a, s)$.
- P1'-5. If $\mathbf{Exe} \approx \text{Poss}(a, s) \rightarrow (\Phi(s) \rightarrow \Psi(\text{do}(a, s)))$, then $\emptyset \approx \text{Poss}(a, s) \rightarrow (\Phi(s) \rightarrow \Psi(\text{do}(a, s)))$.
- P1'-6. If $\mathbf{Eff} \approx \Phi(s)$, then $\emptyset \approx \Phi(s)$.
- P1'-7. If $\mathbf{Eff} \approx \Phi(s) \rightarrow \text{Poss}(a, s)$, then $\emptyset \approx \Phi(s) \rightarrow \text{Poss}(a, s)$.
- P1'-8. If $\mathbf{Eff} \approx \Phi(s) \rightarrow \neg \text{Poss}(a, s)$, then $\emptyset \approx \Phi(s) \rightarrow \neg \text{Poss}(a, s)$.

All these principles say is that a formula of a given type entailed by a module of a different type must be a theorem of the logic.

5 Coupling

As we have seen, coupling evaluates how much a module is tied to or dependent upon other modules. We take as coupling of two or more sets of different types of action laws how much interaction among them is needed to derive a formula of a given type. Interaction here means sharing logical formulas. Now we refine our second design principle:

P2'. *The less new consequences two or several modules have, the less coupled they are.*

(The new consequences of modules $\mathbf{M}_1$ and $\mathbf{M}_2$ are those consequences of $\mathbf{M}_1 \cup \mathbf{M}_2$ that are not consequences neither of $\mathbf{M}_1$ nor of $\mathbf{M}_2$ alone.)

For instance, consider the domain description $\mathbf{D}_1$:

$$\mathbf{Eff}_1 = \left\{ \begin{array}{l} \text{Poss}(\text{tease}, s) \rightarrow \text{walking}(\text{do}(\text{tease}, s)), \\ \text{Poss}(\text{shoot}, s) \rightarrow \\ (\text{loaded}(s) \rightarrow \neg \text{alive}(\text{do}(\text{shoot}, s))) \end{array} \right\}$$

$$\mathbf{Inex}_1 = \{\neg \text{alive}(s) \rightarrow \neg \text{Poss}(\text{tease}, s)\}, \mathbf{Exe}_1 = \emptyset$$

$$\mathbf{Stat}_1 = \left\{ \begin{array}{l} \text{walking}(s) \rightarrow \text{alive}(s), \\ \text{dead}(s) \leftrightarrow \neg \text{alive}(s) \end{array} \right\}$$

Observe that to derive the domain constraint $\text{walking}(s) \rightarrow \neg \text{dead}(s)$ one only needs $\mathbf{Stat}_1$, i.e., no other module is required for that. On the other hand, to conclude $\text{dead}(s) \rightarrow \neg \text{Poss}(\text{tease}, s)$ one needs both $\mathbf{Stat}_1$ and $\mathbf{Inex}_1$.

Totally decoupled descriptions are not common in applications of real interest. For the example above, it seems to be impossible to diminish the interaction between $\mathbf{Stat}_1$ and $\mathbf{Inex}_1$ without abandoning the concept of state constraints.

On the other hand, if $\mathbf{Exe}_1$ in our example contained $\text{Poss}(\text{tease}, s)$, things would be different: in this case, with $\mathbf{Inex}_1$ one would be able to infer the state constraint $\text{alive}(s)$, but such a law cannot be derived from $\mathbf{Stat}_1$ alone. A higher degree of interaction between this set and the others is necessary in order to do that. In such a case one would say that there is a high coupling among $\mathbf{D}_1$'s modules.

The principle of minimal coupling P2' can be refined in two more specific design principles:

P2'-1. **No implicit inexecutability laws:**

$$\text{if } \mathbf{D} \approx \Phi(s) \rightarrow \neg \text{Poss}(a, s), \text{ then} \\ \mathbf{Inex}, \mathbf{Stat} \approx \Phi(s) \rightarrow \neg \text{Poss}(a, s)$$

P2'-2. **No implicit state constraints:**

$$\text{if } \mathbf{D} \approx \Phi(s), \text{ then } \mathbf{Stat} \models \Phi(s).$$

P2'-2 is a useful feature of descriptions: beyond being a reasonable principle of design that helps avoiding mistakes, it clearly restricts the search space, and thus makes reasoning easier. To witness, if $\mathbf{D}$ satisfies P2'-2, then its consistency amounts to that of $\mathbf{Stat}$:

Theorem 5.1 If $\mathbf{D}$ has no implicit state constraints, then

$$\mathbf{D} \approx \perp \text{ iff } \mathbf{Stat} \models \perp.$$

5.1 No implicit inexecutability laws

Consider the following domain description $\mathbf{D}_2$:

$$\mathbf{Eff}_2 = \left\{ \begin{array}{l} \text{Poss}(\text{tease}, s) \rightarrow \text{walking}(\text{do}(\text{tease}, s)), \\ \text{Poss}(\text{shoot}, s) \rightarrow \\ (\text{loaded}(s) \rightarrow \neg \text{alive}(\text{do}(\text{shoot}, s))) \end{array} \right\}$$

$$\mathbf{Inex}_2 = \mathbf{Exe}_2 = \emptyset, \mathbf{Stat}_2 = \{\text{walking}(s) \rightarrow \text{alive}(s)\}$$

From $\text{Poss}(\text{tease}, s) \rightarrow \text{walking}(\text{do}(\text{tease}, s))$ it follows with $\mathbf{Stat}_2$ that $\text{Poss}(\text{tease}, s) \rightarrow \text{alive}(\text{do}(\text{tease}, s))$, i.e., in every situation, after teasing the turkey is alive:

$$\mathbf{Eff}_2, \mathbf{Stat}_2 \approx \text{Poss}(\text{tease}, s) \rightarrow \text{alive}(\text{do}(\text{tease}, s))$$

Now as $\mathbf{D}_2 \approx \neg \text{alive}(s) \rightarrow \neg \text{alive}(\text{do}(\text{tease}, s))$, the status of fluent *alive* is not modified by the *tease* action, and we have $\mathbf{Eff}_2, \mathbf{Stat}_2 \approx (\text{Poss}(\text{tease}, s) \wedge \neg \text{alive}(s)) \rightarrow (\text{alive}(\text{do}(\text{tease}, s)) \wedge \neg \text{alive}(\text{do}(\text{tease}, s)))$. From this it follows $\mathbf{D}_2 \approx \neg \text{alive}(s) \rightarrow \neg \text{Poss}(\text{tease}, s)$, i.e., the turkey cannot be teased if it is dead. But $\mathbf{Inex}_2, \mathbf{Stat}_2 \not\approx \neg \text{alive}(s) \rightarrow \neg \text{Poss}(\text{tease}, s)$, hence Principle P2'-1 is violated. The formula $\neg \text{alive}(s) \rightarrow \neg \text{Poss}(\text{tease}, s)$ is an example of what we call an *implicit inexecutability law*.

In the literature, such laws are also known as *implicit qualifications* [Ginsberg and Smith, 1988], and it has been argued that it is a positive feature of reasoning about actions frameworks to leave them implicit and provide mechanisms for inferring them [Lin, 1995; Thielscher, 1995]. The other way round, one might argue as well that implicit qualifications indicate that the domain has not been described in an adequate manner: inexecutability laws have a form simpler than that of effect laws, and it might be reasonably expected that it is easier to exhaustively describe them. (Note that nevertheless this is not related to the qualification problem, which basically says that it is difficult to state all the executability laws of a domain.) Thus, all the inexecutabilities should be explicitly stated, and this is what Principle P2'-1 says.

5.2 No implicit state constraints

Executability laws increase expressive power, but might conflict with inexecutability laws. For instance, let $\mathbf{D}_3$ be such that $\mathbf{Eff}_3 = \mathbf{Eff}_2$, $\mathbf{Inex}_3 = \{\neg \text{alive}(s) \rightarrow \neg \text{Poss}(\text{tease}, s)\}$, $\mathbf{Exe}_3 = \{\text{Poss}(\text{tease}, s)\}$, and $\mathbf{Stat}_3 = \mathbf{Stat}_2$. (Note that Principle P2'-1 is satisfied.) We have the unintuitive $\mathbf{Inex}_3, \mathbf{Exe}_3 \approx \text{alive}(s)$: the turkey is immortal! This is an *implicit state constraint* because $\text{alive}(s)$ does not follow from $\mathbf{Stat}_3$ alone: P2'-2 is violated.

The existence of implicit state constraints may thus indicate too strong executability laws: in our example, one wrongly assumed that *tease* is always executable. It may also indicate that the inexecutability laws are too strong, or that the state constraints are too weak.

6 Results for a dependence based solution to the frame problem

Given an axiomatic theory of actions with a solution to the frame and the ramification problems, we are interested in knowing whether domain descriptions encoded in it satisfy or not our set of design principles. Here we chose to use the modal framework of $\mathcal{LAP}_{\sim}$ [Castilho *et al.*, 1999], which has been shown to support Reiter's solution to the frame problem [Demolombe *et al.*, 2003] and also proposes an assessment of the ramification problem.

Let $tr_{SitCalc}$ be a translation of a domain description in $\mathcal{LAP}_{\sim}$ into the Situation Calculus. Dependences $a \rightsquigarrow l$ are translated into predicates $dep(a, l)$, meaning that action a may cause literal l to be true. The extension of dep is then circumscribed (cf. Schubert's explanation closure assumption). As examples, $dep(\text{shoot}, \neg \text{walking})$ means that *shoot* may cause *walking* to be false, and the absence of $dep(\text{tease}, \text{alive})$ induces the frame axiom $\neg \text{alive}(s) \rightarrow \neg \text{alive}(\text{do}(\text{tease}, s))$.

Theorem 6.1 If $\mathbf{D}_{\mathcal{LAP}_{\sim}}$ is a domain description in $\mathcal{LAP}_{\sim}$, then $tr_{SitCalc}(\mathbf{D}_{\mathcal{LAP}_{\sim}})$ satisfies Principles P1'-1—P1'-7.

Even in $\mathcal{LAP}_{\sim}$, however, it is possible to derive inexecutabilities from $\mathbf{Eff}$ (see the example in Section 4), which violates Principle P1'-8. Establishing maximal cohesion of $\mathbf{Eff}$ in this case involves weakening of preconditions of action effects. Anyway, conceiving an algorithm to accomplish this task is not difficult (due to space limitations we omit its presentation here).

Checking whether a domain description satisfies Principle P2'-2 can be made with little adaptation of the material on the subject present in the literature [Zhang *et al.*, 2002; Lang *et al.*, 2003; Herzig and Varzinczak, 2004]. We do not deepen into further details here, and just present the main results that we obtain when considering descriptions that satisfy the design principles that have been proposed (due to space limits no proof is given).

Theorem 6.2 Let $\mathbf{D}$ be the translation into Situation Calculus of a domain description in $\mathcal{LAP}_{\sim}$. If $\mathbf{D}$ has no implicit state constraints, then $\mathbf{D} \approx \text{Poss}(a, s) \rightarrow (\Phi(s) \rightarrow \Psi(\text{do}(a, s)))$ iff $\mathbf{Eff}_a, \mathbf{Inex}_a, \mathbf{Stat} \approx \text{Poss}(a, s) \rightarrow (\Phi(s) \rightarrow \Psi(\text{do}(a, s)))$.

This means that under P2'-2 one has modularity inside $\mathbf{Eff}$, too: when deducing the effects of action a we need not consider the action laws for the other actions. Versions for executability and inexecutability can be stated as well.

Theorem 6.3 There exist descriptions $\mathbf{D}$ not satisfying P2'-2 such that $\mathbf{D} \approx \text{Poss}(a, s) \rightarrow (\Phi(s) \rightarrow \Psi(\text{do}(a, s)))$ and $\mathbf{Eff}_a, \mathbf{Inex}_a, \mathbf{Stat} \not\approx \text{Poss}(a, s) \rightarrow (\Phi(s) \rightarrow \Psi(\text{do}(a, s)))$.

For example, just take $\mathbf{D}_3$ as before:

$\mathbf{D}_3 \approx \text{Poss}(\text{shoot}, s) \rightarrow (\neg \text{alive}(s) \rightarrow \text{alive}(\text{do}(\text{shoot}, s)))$, however $\mathbf{Eff}_{3\text{shoot}}, \mathbf{Inex}_{3\text{shoot}}, \mathbf{Stat}_3 \not\approx \text{Poss}(\text{shoot}, s) \rightarrow (\neg \text{alive}(s) \rightarrow \text{alive}(\text{do}(\text{shoot}, s)))$.

7 Related work

Pirri and Reiter [1999] have investigated the metatheory of the Situation Calculus. In a spirit similar to ours, they use executability laws and effect laws. Contrarily to us, their executability laws are equivalences and are thus at the same time inexecutability laws. There are no state constraints, i.e., $\mathbf{Stat} = \emptyset$. For this setting they give a syntactical condition on effect laws forcing them not to interact with executability laws, which precludes implicit state constraints. Basically, the condition says that when there are effect laws $\text{Poss}(a, s) \rightarrow (\Phi(s) \rightarrow \Phi(\text{do}(a, s)))$ and $\text{Poss}(a, s) \rightarrow (\Phi'(s) \rightarrow \Phi'(\text{do}(a, s)))$, then $\Phi(s)$ and $\Phi'(s)$ are inconsistent (which essentially amounts to having in their domain descriptions a kind of “implicit state constraint schema” of the form $\neg(\Phi(s) \wedge \Phi'(s))$).

This then allows them to show that such descriptions are always consistent. Moreover they thus simplify the entailment problem for this calculus, and show for several problems such as consistency or regression that only some of the modules of a domain description are necessary.

Amir [2000] focuses on design and maintenance of action descriptions applying concepts of the object-oriented paradigm in the Situation Calculus. In that work, guidelines for a partitioned representation of a given description are presented, with which the inference task can also be optimized, as it is restricted to the part of the domain description that is really relevant to a given query. This is observed specially when different agents are involved: the design of an agent's description can be done with no regard to others', and after the integration of multiple agents, queries about an agent's beliefs do not take into account the belief state of other agents.

In that work, executabilities are as in [Pirri and Reiter, 1999] and the same condition on effect laws is assumed, which syntactically avoids implicit state constraints.

Despite of using many of the object-oriented paradigm tools and techniques, no mention is made to the concepts of cohesion and coupling. In the approach presented in [Amir, 2000], even if modules are highly cohesive, they are not lowly coupled, due to the dependence between objects in the reasoning process defined there. We do not investigate this further here, but conjecture that this could be done there by, during the reasoning process, avoiding passing to a module a formula of a type different from those it contains.

The present work generalizes and extends Pirri and Reiter's result to the case where $\text{Stat} \neq \emptyset$ and both [Pirri and Reiter, 1999; Amir, 2000] where the syntactical restriction on effect laws is not made. This gives us more expressive power, as we can reason about inexecutabilities, and a better modularity in the sense that we do not combine formulas that are conceptually different (viz. executabilities and inexecutabilities).

8 Conclusion

We have established a link between knowledge engineering and software engineering showing that many of the concepts and techniques developed for the latter are useful in the design and maintenance of domain descriptions. In particular, with the concepts of cohesion and coupling we get better criteria for domain description evaluation.

Our central hypothesis is that the different types of axioms should be neatly separated and only interfere in one sense: state constraints together with action laws may have consequences that do not follow from the action laws alone. The other way round, action laws should not allow to infer new state constraints, effect laws should not allow to infer inexecutability laws, etc.

At first glance, because of **Stat**'s interaction with other modules, it could be said that domain descriptions described in our way do not completely minimize coupling. However, given the intrinsic nature of **Stat**, observe that we cannot do otherwise: in the same way it is not possible to write completely decoupled methods (or the program will not work!), we cannot have totally decoupled domain descriptions (unless, of course, we constrain ourselves to domains without ramifications like in [Pirri and Reiter, 1999]).

It could be argued that unintuitive consequences in domain descriptions are mainly due to badly written axioms and not to the lack of modularity. True enough, but what we have presented here is the case that making a domain description modular gives us a tool to detect at least some of such problems and correct it. (But note that we do not claim to correct badly written axioms automatically and once for all). Besides this, having separate entities in the ontology and controlling their interaction help us to localize where the problems are, which can be crucial for real world applications.

References

- [Amir, 2000] E. Amir. (De)composition of situation calculus theories. In *Proc. AAAI'2000*, pages 456–463, 2000.
- [Castilho et al., 1999] M. A. Castilho, O. Gasquet, and A. Herzig. Formalizing action and change in modal logic I: the frame problem. *J. of Logic and Computation*, 9(5):701–735, 1999.
- [Demolombe et al., 2003] R. Demolombe, A. Herzig, and I. Varzinczak. Regression in modal logic. *J. of Applied Non-classical Logics (JANCL)*, 13(2):165–185, 2003.
- [Doherty et al., 1996] P. Doherty, W. Łukaszewicz, and A. Szalas. Explaining explanation closure. In *Proc. Intl. Symp. on Methodologies for Int. Systems*, 1996.
- [Finger, 1987] J. J. Finger. *Exploiting constraints in design synthesis*. PhD thesis, Stanford University, 1987.
- [Ginsberg and Smith, 1988] M. L. Ginsberg and D. E. Smith. Reasoning about actions II: The qualification problem. *Artificial Intelligence*, 35(3):311–342, 1988.
- [Herzig and Varzinczak, 2004] A. Herzig and I. Varzinczak. Domain descriptions should be modular. In *Proc. ECAI'04*, pages 348–352, 2004.
- [Lang et al., 2003] J. Lang, F. Lin, and P. Marquis. Causal theories of action – a computational core. In *Proc. IJCAI'03*, pages 1073–1078, 2003.
- [Lin, 1995] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. IJCAI'95*, pages 1985–1991, 1995.
- [McCain and Turner, 1995] N. McCain and H. Turner. A causal theory of ramifications and qualifications. In *Proc. IJCAI'95*, pages 1978–1984, 1995.
- [McCarthy and Hayes, 1969] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, volume 4, pages 463–502. 1969.
- [McCarthy, 1988] J. McCarthy. *Mathematical logic in artificial intelligence*. Daedalus, 1988.
- [Pirri and Reiter, 1999] F. Pirri and R. Reiter. Some contributions to the metatheory of the situation calculus. *Journal of the ACM*, 46(3):325–361, 1999.
- [Pressman, 1992] R. S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 1992.
- [Schubert, 1990] L. K. Schubert. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer, 1990.
- [Shanahan, 1997] M. Shanahan. *Solving the frame problem: a mathematical investigation of the common sense law of inertia*. MIT Press, Cambridge, MA, 1997.
- [Thielscher, 1995] M. Thielscher. Computing ramifications by postprocessing. In *Proc. IJCAI'95*, pages 1994–2000, 1995.
- [Zhang et al., 2002] D. Zhang, S. Chopra, and N. Y. Foo. Consistency of action descriptions. In *PRICAI'02, Topics in Artificial Intelligence*. Springer-Verlag, 2002.

A Tableaux Decision Procedure for $\mathcal{SHOIQ}$

Ian Horrocks and Ulrike Sattler

School of Computer Science, University of Manchester, UK
horrocks|sattler@cs.man.ac.uk

Abstract

This paper presents a tableaux decision procedure for $\mathcal{SHOIQ}$, the DL underlying OWL DL. To the best of our knowledge, this is the first goal-directed decision procedure for $\mathcal{SHOIQ}$.

1 Introduction

Description Logics (DLs) are a family of logic based knowledge representation formalisms. Although they have a range of applications (e.g., configuration [McGuinness & Wright, 1998], and information integration [Calvanese *et al.*, 1998]), they are perhaps best known as the basis for widely used ontology languages such as OIL, DAML+OIL and OWL [Horrocks *et al.*, 2003], the last of which is now a W3C recommendation [Bechhofer *et al.*, 2004]. Two of the three language “species” described in the OWL specification are based on expressive DLs: OWL Lite is based on $\mathcal{SHIF}$, and OWL DL is based on $\mathcal{SHOIN}$. This decision was motivated by a requirement that key inference problems (such as ontology consistency) be decidable, and that it should be possible to provide reasoning services to support ontology design and deployment [Horrocks *et al.*, 2003].

Although the ontology consistency problem for $\mathcal{SHOIN}$ is known to be decidable,¹ to the best of our knowledge no “practical” decision procedure is known for it, i.e., no goal directed procedure that is likely to perform well with realistic ontology derived problems [Tobies, 2001; Horrocks & Sattler, 2001]. In this paper we present such a decision procedure for $\mathcal{SHOIQ}$, i.e., $\mathcal{SHOIN}$ extended with *qualified* number restrictions [Baader & Hollunder, 1991]. The algorithm extends the well-known tableaux algorithm for $\mathcal{SHIQ}$ [Horrocks *et al.*, 1999], which is the basis for several highly successful implementations [Horrocks & Patel-Schneider, 1998; Haarslev & Möller, 2001; Pellet, 2003].

The $\mathcal{O}$ in $\mathcal{SHOIQ}$ denotes *nominals*, i.e., classes with a singleton extension. Nominals are a prominent feature of hybrid logics [Blackburn & Seligman, 1995], and can also

be viewed as a powerful generalisation of *ABox individuals* [Schaerf, 1994; Horrocks & Sattler, 2001]. They occur naturally in ontologies, e.g., when describing a class such as *EUCountries* by enumerating its members, i.e., $\{\text{Austria}, \dots, \text{UnitedKingdom}\}$ (such an enumeration is equivalent to a disjunction of nominals). This allows applications to infer, e.g., that persons who only visit *EUCountries* can visit at most 15 countries.

One reason why DLs (and propositional modal and dynamic logics) enjoy good computational properties, such as being robustly decidable, is that they have some form of tree model property [Vardi, 1997], i.e., if an ontology is consistent, then it has a (form of) tree model. This feature is crucial in the design of tableaux algorithms, allowing them to search only for tree like models. More precisely, tableaux algorithms decide consistency of an ontology by trying to construct an abstraction of a model for it, a so-called “completion graph”. For logics with the tree model property, we can restrict our search/construction to tree-shaped completion graphs.

Tableaux algorithms for expressive DLs employ a cycle detection technique called *blocking* to ensure termination. This is of special interest for $\mathcal{SHIQ}$, where the interaction between inverse roles and number restrictions results in the loss of the *finite model property*, i.e., there are consistent ontologies that only admit infinite models. On such an input, the $\mathcal{SHIQ}$ tableaux algorithm generates a finite tree-shaped completion graph that can be *unravalled* into an infinite tree model, and where a node in the completion graph may stand for infinitely many elements of the model. Even when the language includes nominals, but *excludes* one of number restrictions or inverse roles [Horrocks & Sattler, 2001; Hladik & Model, 2004], or if nominals are restricted to *ABox individuals* [Horrocks *et al.*, 2000], we can work on forest-shaped completion graphs, with each nominal (individual) being the root of a tree-like section; this causes no inherent difficulty as the size of the non-tree part of the graph is restricted by the number of individuals/nominals in the input.

The difficulty in extending the $\mathcal{SHOQ}$ or $\mathcal{SHIQ}$ algorithms to $\mathcal{SHOIQ}$ is due to the interaction between nominals, number restrictions, and inverse roles, which leads to the *almost* complete loss of the tree model property, and causes the complexity of the ontology consistency problem to jump from ExpTime to NExpTime [Tobies, 2000]. To see this, consider an ontology containing the following two axioms that

¹This is an immediate consequence of a reduction of DLs with transitive roles to DLs without such roles [Tobies, 2001] and the fact that applying this reduction to $\mathcal{SHOIN}$ yields a fragment of the two variable fragment of first order logic with counting quantifiers [Pacholski *et al.*, 1997].

use a nominal o to impose an upper bound, say 17, on the number of instances of the concept F :

$$\top \sqsubseteq \exists R^-.o, \quad o \sqsubseteq (\leq 17 R.F)$$

The first statement requires that, in a model of this ontology, every element has an incoming R -edge from o ; the second statement restricts the number of R -edges going from o to instances of F to at most 17. In this case, we might need to consider arbitrarily complex relational structures amongst instances of F , and thus cannot restrict our attention to completion trees or forests. Let us assume that our ontology also forces the existence of an infinite number of instances of another concept, say N , which requires the above mentioned “block and unravel” technique. The consistency of the whole ontology then crucially depends on the relations enforced between instances of F and N , and whether the unravelling of the N -part violates atmost number restrictions that instances of F must satisfy. Summing up, a tableaux algorithm for $SHOIQ$ needs to be able to handle both arbitrarily complex relational structures and finite tree structures representing infinite trees, and to make sure that all constraints are satisfied (especially number restrictions on relations between these two parts), while still guaranteeing termination.

Two key intuitions have allowed us to devise a tableaux algorithm that meets all of these requirements. The first intuition is that, when extending a $SHOIQ$ completion graph, we can distinguish those nodes that may be arbitrarily interconnected (so-called *nominal nodes*) from those nodes that still form a tree structure (so-called *blockable nodes*). Fixing a (double exponential) upper bound on the number of nominal nodes is crucial to proving termination; it is not, however, enough to guarantee termination, as we may repeatedly create and merge nominal nodes (a so-called “yo-yo”).

The second intuition is that the yo-yo problem can be overcome by “guessing” the *exact* number of new nominal nodes resulting from interactions between existing nominal nodes, inverse roles, and number restrictions. This guessing is implemented by a new expansion rule, $RO?$, which, when applied to a relevant ($\leq nR.C$) concept, generates (non-deterministically) between 1 and n new nominal nodes, all of which are pairwise disjoint. This prevents the repeated yo-yo construction, and termination is now guaranteed by the upper bound on the number of nominal nodes and the use of standard blocking techniques for the blockable nodes. The non-determinism introduced by this rule could clearly be problematical for large values of n , but large values in number restrictions are already known to be problematical for $SHIQ$. Moreover, the rule has excellent “pay as you go” characteristics: in case number restrictions are functional (i.e., where n is 1),² the new rule becomes deterministic; in case there are no interactions between number restrictions, inverse roles and nominals, the rule will never be applied; in case there are no nominals, the new algorithm will behave like the algorithm for $SHIQ$; and in case there are no inverse roles, the new algorithm will behave like the algorithm for $SHOQ$.

For more examples and full proofs, we refer the reader to [Horrocks & Sattler, 2005].

²A feature of many realistic ontologies; see, e.g., the DAML ontology library at <http://www.daml.org/ontologies/>

2 Preliminaries

In this section, we introduce the syntax, semantics, and inference problems of the DL $SHOIQ$.

Definition 1 Let $\mathbf{R}$ be a set of *role names* with both transitive and normal role names $\mathbf{R}_+ \cup \mathbf{R}_P = \mathbf{R}$, where $\mathbf{R}_P \cap \mathbf{R}_+ = \emptyset$. The set of $SHOIQ$ -roles (or *roles* for short) is $\mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$. A *role inclusion axiom* is of the form $R \sqsubseteq S$, for two roles R and S . A *role hierarchy* is a finite set of role inclusion axioms.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$, the *domain* of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$ which maps every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that, for $P \in \mathbf{R}$ and $R \in \mathbf{R}_+$,

$$\langle x, y \rangle \in P^{\mathcal{I}} \text{ iff } \langle y, x \rangle \in P^{-\mathcal{I}},$$

$$\text{and if } \langle x, y \rangle \in R^{\mathcal{I}} \text{ and } \langle y, z \rangle \in R^{\mathcal{I}}, \text{ then } \langle x, z \rangle \in R^{\mathcal{I}}.$$

An interpretation $\mathcal{I}$ *satisfies a role hierarchy* $\mathcal{R}$ if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ for each $R \sqsubseteq S \in \mathcal{R}$; such an interpretation is called a *model* of $\mathcal{R}$.

We introduce some notation to make the following considerations easier.

1. The inverse relation on roles is symmetric, and to avoid considering roles such as R^{--} , we define a function Inv which returns the inverse of a role: for $R \in \mathbf{R}$, $\text{Inv}(R) := R^-$ and $\text{Inv}(R^-) := R$.
2. Since set inclusion is transitive and $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ implies $\text{Inv}(R)^{\mathcal{I}} \subseteq \text{Inv}(S)^{\mathcal{I}}$, for a role hierarchy $\mathcal{R}$, we introduce $\sqsubseteq_{\mathcal{R}}$ as the transitive-reflexive closure of $\sqsubseteq$ on $\mathcal{R} \cup \{\text{Inv}(R) \sqsubseteq \text{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}$. We use $R \equiv_{\mathcal{R}} S$ as an abbreviation for $R \sqsubseteq_{\mathcal{R}} S$ and $S \sqsubseteq_{\mathcal{R}} R$.
3. Obviously, a role R is transitive if and only if its inverse $\text{Inv}(R)$ is transitive. However, in cyclic cases such as $R \equiv_{\mathcal{R}} S$, S is transitive if R or $\text{Inv}(R)$ is a transitive role name. In order to avoid these case distinctions, the function Trans returns true if R is a transitive role—regardless whether it is a role name, the inverse of a role name, or equivalent to a transitive role name (or its inverse): $\text{Trans}(S, \mathcal{R}) := \text{true}$ if, for some P with $P \equiv_{\mathcal{R}} S$, $P \in \mathbf{R}_+$ or $\text{Inv}(P) \in \mathbf{R}_+$; $\text{Trans}(S, \mathcal{R}) := \text{false}$ otherwise.
4. A role R is called *simple* w.r.t. $\mathcal{R}$ if $\text{Trans}(S, \mathcal{R}) = \text{false}$ for all $S \sqsubseteq_{\mathcal{R}} R$.
5. In the following, if $\mathcal{R}$ is clear from the context, we may use $\sqsubseteq$ and $\text{Trans}(S)$ instead of $\sqsubseteq_{\mathcal{R}}$ and $\text{Trans}(S, \mathcal{R})$.

Definition 2 Let N_C be a set of *concept names* with a subset $N_I \subseteq N_C$ of *nominals*. The set of $SHOIQ$ -concepts (or *concepts* for short) is the smallest set such that

1. every concept name $C \in N_C$ is a concept,
2. if C and D are concepts and R is a role, then $(C \sqcap D)$, $(C \sqcup D)$, $(\neg C)$, $(\forall R.C)$, and $(\exists R.C)$ are also concepts (the last two are called universal and existential restrictions, resp.), and
3. if C is a concept, R is a simple role³ and $n \in \mathbb{N}$, then $(\leq nR.C)$ and $(\geq nR.C)$ are also concepts (called atmost and atleast number restrictions).

³Restricting number restrictions to simple roles is required in order to yield a decidable logic [Horrocks *et al.*, 1999].

The interpretation function $\mathcal{I}$ of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ maps, additionally, every concept to a subset of $\Delta^{\mathcal{I}}$ such that

$$\begin{aligned} (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}, & (C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}}, \\ \neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, & \#o^{\mathcal{I}} &= 1 \text{ for all } o \in N_I, \\ (\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, C) \neq \emptyset\}, \\ (\forall R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, \neg C) = \emptyset\}, \\ (\leq nR.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \#R^{\mathcal{I}}(x, C) \leq n\}, \text{ and} \\ (\geq nR.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \#R^{\mathcal{I}}(x, C) \geq n\}, \end{aligned}$$

where $\#M$ is the cardinality of a set M and $R^{\mathcal{I}}(x, C)$ is defined as $\{y \mid \langle x, y \rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$.

For C and D (possibly complex) concepts, $C \sqsubseteq D$ is called a *general concept inclusion* (GCI), and a finite set of GCIs is called a *TBox*.

An interpretation $\mathcal{I}$ satisfies a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and $\mathcal{I}$ satisfies a TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies each GCI in $\mathcal{T}$; such an interpretation is called a *model* of $\mathcal{T}$.

A concept C is *satisfiable* w.r.t. a role hierarchy $\mathcal{R}$ and a TBox $\mathcal{T}$ if there is a model $\mathcal{I}$ of $\mathcal{R}$ and $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$. Such an interpretation is called a *model* of C w.r.t. $\mathcal{R}$ and $\mathcal{T}$. A concept D *subsumes* a concept C w.r.t. $\mathcal{R}$ and $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{R}, \mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in every model $\mathcal{I}$ of $\mathcal{R}$ and $\mathcal{T}$. Two concepts C, D are *equivalent* w.r.t. $\mathcal{R}$ and $\mathcal{T}$ (written $C \equiv_{\mathcal{R}, \mathcal{T}} D$) if they are mutually subsuming w.r.t. $\mathcal{R}$ and $\mathcal{T}$.

As usual, subsumption and satisfiability can be reduced to each other. Like *SHIQ*, in *SHOIQ*, we can reduce reasoning w.r.t. TBoxes and role hierarchies to reasoning w.r.t. role hierarchies only: we can use an “approximation” of a universal role to *internalise* a TBox [Horrocks *et al.*, 1999; Horrocks & Sattler, 2005]. Hence, in the remainder of this paper, we restrict our attention without loss of generality to the satisfiability of *SHOIQ* concepts w.r.t. a role hierarchy.

Finally, we did not choose to make a *unique name assumption*, i.e., two nominals might refer to the same individual. However, the inference algorithm presented below can easily be adapted to the unique name case by a suitable initialisation of the inequality relation $\neq$ used in our algorithm.

3 A Tableau for *SHOIQ*

For ease of presentation, as usual, we assume all concepts to be in *negation normal form* (NNF). Each concept can be transformed into an equivalent one in NNF by pushing negation inwards, making use of de Morgan’s laws and the duality between existential and universal restrictions, and between at-most and at-least number restrictions, [Horrocks *et al.*, 2000]. For a concept C , we use $\neg C$ to denote the NNF of $\neg C$, and we use $\text{sub}(C)$ to denote the set of all subconcepts of C (including C). As usual, for a concept D and a role hierarchy $\mathcal{R}$, we define the set of “relevant sub-concepts” $\text{cl}(D, \mathcal{R})$ as follows (we will often use $\text{cl}(D)$ instead of $\text{cl}(D, \mathcal{R})$):

$$\begin{aligned} \text{cl}(D, \mathcal{R}) := & \text{sub}(D) \cup \{\neg C \mid C \in \text{sub}(D)\} \cup \\ & \{\forall S.E \mid \{\forall R.E, \neg \forall R.E\} \cap \text{sub}(D) \neq \emptyset \\ & \text{and } S \text{ occurs in } \mathcal{R} \text{ or } D\}. \end{aligned}$$

Definition 3 If D is a *SHOIQ*-concept in NNF, $\mathcal{R}$ a role hierarchy, and $\mathbf{R}_D$ is the set of roles occurring in D or $\mathcal{R}$,

together with their inverses, a tableau T for D w.r.t. $\mathcal{R}$ is defined to be a triple $(\mathbf{S}, \mathcal{L}, \mathcal{E})$ such that: $\mathbf{S}$ is a set of individuals, $\mathcal{L} : \mathbf{S} \rightarrow 2^{\text{cl}(D)}$ maps each individual to a set of concepts which is a subset of $\text{cl}(D)$, $\mathcal{E} : \mathbf{R}_D \rightarrow 2^{\mathbf{S} \times \mathbf{S}}$ maps each role in $\mathbf{R}_D$ to a set of pairs of individuals, and there is some individual $s \in \mathbf{S}$ such that $D \in \mathcal{L}(s)$. For all $s, t \in \mathbf{S}$, $C, C_1, C_2 \in \text{cl}(D)$, $R, S \in \mathbf{R}_D$, and

$$S^T(s, C) := \{t \in \mathbf{S} \mid \langle s, t \rangle \in \mathcal{E}(S) \text{ and } C \in \mathcal{L}(t)\},$$

it holds that:

- (P1) if $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$,
- (P2) if $C_1 \sqcap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$,
- (P3) if $C_1 \sqcup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$,
- (P4) if $\forall R.C \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}(R)$, then $C \in \mathcal{L}(t)$,
- (P5) if $\exists R.C \in \mathcal{L}(s)$, then there is some $t \in \mathbf{S}$ such that $\langle s, t \rangle \in \mathcal{E}(R)$ and $C \in \mathcal{L}(t)$,
- (P6) if $\forall S.C \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}(R)$ for some $R \sqsubseteq S$ with $\text{Trans}(R)$, then $\forall R.C \in \mathcal{L}(t)$,
- (P7) if $(\geq nS.C) \in \mathcal{L}(s)$, then $\#S^T(s, C) \geq n$,
- (P8) if $(\leq nS.C) \in \mathcal{L}(s)$, then $\#S^T(s, C) \leq n$,
- (P9) if $(\leq nS.C) \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}(S)$, then $\{C, \neg C\} \cap \mathcal{L}(t) \neq \emptyset$,
- (P10) if $\langle s, t \rangle \in \mathcal{E}(R)$ and $R \sqsubseteq S$, then $\langle s, t \rangle \in \mathcal{E}(S)$,
- (P11) $\langle s, t \rangle \in \mathcal{E}(R)$ iff $\langle t, s \rangle \in \mathcal{E}(\text{Inv}(R))$, and
- (P12) if $o \in \mathcal{L}(s) \cap \mathcal{L}(t)$ for some $o \in N_I$, then $s = t$.

Lemma 4 A *SHOIQ*-concept D in NNF is satisfiable w.r.t. a role hierarchy $\mathcal{R}$ iff D has a tableau w.r.t. $\mathcal{R}$.

The proof is similar to the one found in [Horrocks *et al.*, 1999]. Roughly speaking, we construct a model $\mathcal{I}$ from a tableau by taking $\mathbf{S}$ as its interpretation domain and adding the missing role-successorships for transitive roles. Then, by induction on the structure of formulae, we prove that, if $C \in \mathcal{L}(s)$, then $s \in C^{\mathcal{I}}$. Most cases are straightforward. As for *SHIQ*, (P7) and (P8) together with the fact that only simple roles occur in number restrictions ensure that this implication holds for number restrictions C , and (P12) ensures that nominals are indeed interpreted as singletons. For the converse, we can easily transform any model into a tableau.

4 A tableau algorithm for *SHOIQ*

From Lemma 4, an algorithm which constructs a tableau for a *SHOIQ*-concept D can be used as a decision procedure for the satisfiability of D with respect to a role hierarchy $\mathcal{R}$. Such an algorithm will now be described in detail.

Definition 5 Let $\mathcal{R}$ be a role hierarchy and D a *SHOIQ*-concept in NNF. A *completion graph* for D with respect to $\mathcal{R}$ is a directed graph $\mathbf{G} = (V, E, \mathcal{L}, \neq)$ where each node $x \in V$ is labelled with a set $\mathcal{L}(x) \subseteq \text{cl}(D) \cup N_I \cup \{(\leq mR.C) \mid (\leq nR.C) \in \text{cl}(D) \text{ and } m \leq n\}$ and each edge $\langle x, y \rangle \in E$ is labelled with a set of role names $\mathcal{L}(\langle x, y \rangle)$ containing (possibly inverse) roles occurring in D or $\mathcal{R}$. Additionally, we keep track of inequalities between nodes of the graph with a symmetric binary relation $\neq$ between the nodes of $\mathbf{G}$.

If $\langle x, y \rangle \in E$, then y is called a *successor* of x and x is called a *predecessor* of y . *Ancestor* is the transitive closure of predecessor, and *descendant* is the transitive closure of successor. A node y is called an *R-successor* of a node x if, for some R' with $R' \sqsubseteq R$, $R' \in \mathcal{L}(\langle x, y \rangle)$; x is called an *R-predecessor* of y if y is an *R-successor* of x . A node y is called a *neighbour* (*R-neighbour*) of a node x if y is a successor (*R-successor*) of x or if x is a successor (*Inv(R)-successor*) of y .

For a role S and a node x in $\mathbf{G}$, we define the set of x 's *S-neighbours* with C in their label, $S^{\mathbf{G}}(x, C)$, as follows:

$$S^{\mathbf{G}}(x, C) := \{y \mid y \text{ is an } S\text{-neighbour of } x \text{ and } C \in \mathcal{L}(y)\}.$$

$\mathbf{G}$ is said to contain a *clash* if

1. for some $A \in N_C$ and node x of $\mathbf{G}$, $\{A, \neg A\} \subseteq \mathcal{L}(x)$,
2. for some node x of $\mathbf{G}$, $(\leq nS.C) \in \mathcal{L}(x)$ and there are $n+1$ *S-neighbours* $y_0, \dots, y_n$ of x with $C \in \mathcal{L}(y_i)$ for each $0 \leq i \leq n$ and $y_i \neq y_j$ for each $0 \leq i < j \leq n$, or
3. for some $o \in N_I$, there are $x \neq y$ with $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$.

If $o_1, \dots, o_\ell$ are all the nominals occurring in D , then the tableau algorithm starts with the completion graph $\mathbf{G} = (\{r_0, r_1, \dots, r_\ell\}, \emptyset, \mathcal{L}, \emptyset)$ with $\mathcal{L}(r_0) = \{D\}$ and $\mathcal{L}(r_i) = \{o_i\}$ for $1 \leq i \leq \ell$. $\mathbf{G}$ is then expanded by repeatedly applying the expansion rules given in Figures 1 and 2, stopping if a clash occurs.

Next, we define and explain some terms and operations used in the expansion rules:

Nominal Nodes and Blockable Nodes We distinguish two types of nodes in $\mathbf{G}$, *nominal* nodes and *blockable* nodes. A node x is a nominal node if $\mathcal{L}(x)$ contains a nominal. A node that is not a nominal node is a blockable node. A nominal $o \in N_I$ is said to be *new* in $\mathbf{G}$ if no node in $\mathbf{G}$ has o in its label.

Comment: like ABox individuals [Horrocks *et al.*, 2000], nominal nodes can be arbitrarily interconnected. In contrast, blockable nodes are only found in tree-like structures rooted in nominal nodes (or in r_0); a branch of such a tree may end with an edge leading to a nominal node.

In RO? , we use *new* nominals to create new nominal nodes—intuitively, to fix the identity of certain, constrained neighbours of nominal nodes. An upper bound on the number of nominal nodes that can be generated in a given completion graph will be crucial for termination of the construction, given that blocking cannot be applied to nominal nodes.

Blocking A node x is *label blocked* if it has ancestors x', y and y' such that

1. x is a successor of x' and y is a successor of y' ,
2. y, x and all nodes on the path from y to x are blockable,
3. $\mathcal{L}(x) = \mathcal{L}(y)$ and $\mathcal{L}(x') = \mathcal{L}(y')$, and
4. $\mathcal{L}(\langle x', x \rangle) = \mathcal{L}(\langle y', y \rangle)$.

In this case, we say that y *blocks* x . A node is *blocked* if either it is label blocked or it is blockable and its predecessor is blocked; if the predecessor of a blockable node x is blocked, then we say that x is *indirectly blocked*.

Comment: blocking is defined exactly as for *SHIQ*, with the only difference that, in the presence of nominals, we must

$\text{R}\sqcap$:	if 1. $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$ then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$
$\text{R}\sqcup$:	if 1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$ then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$\text{R}\exists$:	if 1. $\exists S.C \in \mathcal{L}(x)$, x is not blocked, and 2. x has no safe <i>S-neighbour</i> y with $C \in \mathcal{L}(y)$, then create a new node y with $\mathcal{L}(\langle x, y \rangle) = \{S\}$ and $\mathcal{L}(y) = \{C\}$
$\text{R}\forall$:	if 1. $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. there is an <i>S-neighbour</i> y of x with $C \notin \mathcal{L}(y)$ then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$
$\text{R}\forall+$:	if 1. $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and 2. there is some R with $\text{Trans}(R)$ and $R \sqsubseteq S$, 3. there is an <i>R-neighbour</i> y of x with $\forall R.C \notin \mathcal{L}(y)$ then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{\forall R.C\}$
R? :	if 1. $(\leq nS.C) \in \mathcal{L}(x)$, x is not indirectly blocked, and there 2. is an <i>S-neighbour</i> y of x with $\{C, \neg C\} \cap \mathcal{L}(y) = \emptyset$ then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, \neg C\}$
$\text{R}\geq$:	if 1. $(\geq nS.C) \in \mathcal{L}(x)$, x is not blocked, and 2. there are not n safe <i>S-neighbours</i> $y_1, \dots, y_n$ of x with $C \in \mathcal{L}(y_i)$ and $y_i \neq y_j$ for $1 \leq i < j \leq n$ then create n new nodes $y_1, \dots, y_n$ with $\mathcal{L}(\langle x, y_i \rangle) = \{S\}$, $\mathcal{L}(y_i) = \{C\}$, and $y_i \neq y_j$ for $1 \leq i < j \leq n$.
$\text{R}\leq$:	if 1. $(\leq nS.C) \in \mathcal{L}(z)$, z is not indirectly blocked, and 2. $\#S^{\mathbf{G}}(z, C) > n$ and there are two <i>S-neighbours</i> x, y of z with $C \in \mathcal{L}(x) \cap \mathcal{L}(y)$, and not $x \neq y$ then 1. if x is a nominal node, then $\text{Merge}(y, x)$ 2. else if y is a nominal node or an ancestor of x , then $\text{Merge}(x, y)$ 3. else $\text{Merge}(y, x)$

Figure 1: The tableaux expansion rules for *SHIQ*

take care that none of the nodes between a blocking and a blocked one is a nominal node.

Generating and Shrinking Rules and Safe Neighbours

The rules $\text{R}\geq$, $\text{R}\exists$, and R? are called *generating rules*, and the rules $\text{R}\leq$ and $\text{R}\forall$ are called *shrinking rules*. An *R-neighbour* y of a node x is *safe* if (i) x is blockable or if (ii) x is a nominal node and y is not blocked.

Pruning When a node y is *merged* into a node x , we “prune” the completion graph by removing y and, recursively, all blockable successors of y . More precisely, pruning a node y (written $\text{Prune}(y)$) in $\mathbf{G} = (V, E, \mathcal{L}, \neq)$ yields a graph that is obtained from $\mathbf{G}$ as follows:

1. for all successors z of y , remove $\langle y, z \rangle$ from E and, if z is blockable, $\text{Prune}(z)$;
2. remove y from V .

Merging merging a node y into a node x (written $\text{Merge}(y, x)$) in $\mathbf{G} = (V, E, \mathcal{L}, \neq)$ yields a graph that is obtained from $\mathbf{G}$ as follows:

1. for all nodes z such that $\langle z, y \rangle \in E$
 - (a) if $\{\langle x, z \rangle, \langle z, x \rangle\} \cap E = \emptyset$, then add $\langle z, x \rangle$ to E and set $\mathcal{L}(\langle z, x \rangle) = \mathcal{L}(\langle z, y \rangle)$,
 - (b) if $\langle z, x \rangle \in E$, then set $\mathcal{L}(\langle z, x \rangle) = \mathcal{L}(\langle z, y \rangle) \cup \mathcal{L}(\langle z, x \rangle)$,

- (c) if $\langle x, z \rangle \in E$, then set $\mathcal{L}(\langle x, z \rangle) = \mathcal{L}(\langle x, z \rangle) \cup \{\text{Inv}(S) \mid S \in \mathcal{L}(\langle z, y \rangle)\}$, and
- (d) remove $\langle z, y \rangle$ from E ;
- 2. for all nominal nodes z such that $\langle y, z \rangle \in E$
 - (a) if $\{\langle x, z \rangle, \langle z, x \rangle\} \cap E = \emptyset$, then add $\langle x, z \rangle$ to E and set $\mathcal{L}(\langle x, z \rangle) = \mathcal{L}(\langle y, z \rangle)$,
 - (b) if $\langle x, z \rangle \in E$, then set $\mathcal{L}(\langle x, z \rangle) = \mathcal{L}(\langle x, z \rangle) \cup \mathcal{L}(\langle y, z \rangle)$,
 - (c) if $\langle z, x \rangle \in E$, then set $\mathcal{L}(\langle z, x \rangle) = \mathcal{L}(\langle z, x \rangle) \cup \{\text{Inv}(S) \mid S \in \mathcal{L}(\langle y, z \rangle)\}$, and
 - (d) remove $\langle y, z \rangle$ from E ;
- 3. set $\mathcal{L}(x) = \mathcal{L}(x) \cup \mathcal{L}(y)$;
- 4. add $x \neq z$ for all z such that $y \neq z$; and
- 5. Prune(y).

If y was merged into x , we call x a *direct heir* of y , and we use being an *heir* of another node for the transitive closure of being a “direct heir”.

Comment: merging is a generalisation of what is often done to satisfy an atmost number restriction for a node x in case that x has too many neighbours. In *SHOIQ*, we might need to merge nominal nodes that are related in some arbitrary, non-tree-like way, so we must take care of all incoming and outgoing edges. The usage of “heir” is quite intuitive since, after y has been merged into x , x has “inherited” all of y ’s properties, i.e., its label, its inequalities, and its incoming and outgoing edges (except for any outgoing edges to blockable nodes, which are removed by Prune).

Level (of Nominal Nodes) Let $o_1, \dots, o_\ell$ be all the nominals occurring in the input concept D . We define the *level* of a nominal node y inductively as follows: (i) each (nominal) node x with an $o_i \in \mathcal{L}(x)$, $1 \leq i \leq \ell$, is of level 0, and (ii) a nominal node x is of level i if x is not of some level $j < i$ and x has a neighbour that is of level $i - 1$.

Comment: if a node with a lower level is merged into another node, the level of the latter node may be reduced, but it can never be increased because Merge preserves all edges connecting nominal nodes. The completion graph initially contains only level 0 nodes.

Strategy (of Rule Application) the expansion rules are applied according to the following strategy:

1. RO is applied with highest priority.
2. Next, $R \leq$ and RO? are applied, and they are applied first to nominal nodes with lower levels. In case they are both applicable to the same node, RO? is applied first.
3. All other rules are applied with a lower priority.

Comment: this strategy is necessary for termination, and in particular to fix an upper bound on the number of applications of RO?.

We can now finish the description of the tableau algorithm: A completion graph is *complete* if it contains a clash, or when none of the rules is applicable. If the expansion rules can be applied to D and $\mathcal{R}$ in such a way that they yield a complete and clash-free completion graph, then the algorithm returns “ D is satisfiable w.r.t. $\mathcal{R}$ ”, and “ D is unsatisfiable w.r.t. $\mathcal{R}$ ” otherwise.

RO:	if for some $o \in N_I$ there are 2 nodes x, y with $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$ and not $x \neq y$ then Merge(x, y)
RO?:	if 1. $(\leq n S.C) \in \mathcal{L}(x)$, x is a nominal node, and there is a blockable S -neighbour y of x such that $C \in \mathcal{L}(y)$ and x is a successor of y 2. there is no m with $1 \leq m \leq n$, $(\leq m S.C) \in \mathcal{L}(x)$, and there are m nominal S -neighbours $z_1, \dots, z_m$ of x with $C \in \mathcal{L}(z_i)$ and $y_i \neq y_j$ for all $1 \leq i < j \leq m$. then 1. guess $m \leq n$ and set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{(\leq m S.C)\}$ 2. create m new nodes $y_1, \dots, y_m$ with $\mathcal{L}(\langle x, y_i \rangle) = \{S\}$, $\mathcal{L}(y_i) = \{C, o_i\}$ for $o_i \in N_I$ new in $\mathbf{G}$, and $y_i \neq y_j$ for $1 \leq i < j \leq m$,

Figure 2: The new expansion rules for *SHOIQ*

Lemma 6 Let D be a *SHOIQ* concept in NNF and $\mathcal{R}$ a role hierarchy.

1. When started with D and $\mathcal{R}$, the completion algorithm terminates.
2. D has a tableau w.r.t. $\mathcal{R}$ if and only if the expansion rules can be applied to D and $\mathcal{R}$ such that they yield a complete and clash-free completion graph.

Proof (sketch): Let $m = |\text{cl}(D)|$, k the number of roles and their inverses in D and $\mathcal{R}$, n the maximal number in atmost number restrictions, and $o_1, \dots, o_\ell$ be all nominals occurring in D , and let $\lambda := 2^{2m+k}$.

Termination is a consequence of the usual *SHIQ* conditions (1 – 3 below) with respect to the blockable tree parts of the graph [Horrocks *et al.*, 1999], plus the fact that there is a bound on the number of new nominal nodes that can be added to $\mathbf{G}$ by RO? (4 below): (1) all but the shrinking rules strictly extend the completion graph by adding new nodes and edges or extending node labels, while neither removing nodes, edges, or elements from node labels; (2) new nodes are only added by the generating rules, and each of these rules is applied at most once for a given concept in the label of a given node x or its heirs; (3) the length of a path consisting entirely of blockable nodes is bounded by λ : this is due to the blocking condition and the fact that, if x is blockable, then $\mathcal{L}(x) \subseteq \text{cl}(D, \mathcal{R})$ and thus does not contain nominals; and (4) at most $\mathcal{O}(\ell(mn)^\lambda)$ nominal nodes are generated.

To see (4), observe that RO? can only be applied after a nominal o_i , for $i \leq \ell$, has been added to the label of a blockable node x in a branch of one of the blockable “trees”; otherwise, a blockable node cannot have a nominal node as a successor. In this case, RO will immediately merge x with an existing level 0 node, say r_i . As a consequence of this merging, it is possible that the predecessor of x is merged into a nominal node n_1 by $R \leq$ (due to the pruning part of merging, this cannot happen to a successor of x). By definition, n_1 is of level 0 or 1. Repeating this argument, it is possible that all ancestors of x are merged into nominal nodes. However, as the maximum length of a sequence of blockable nodes is λ , blockable ancestors of x can only be merged into nominal nodes of level below λ . Given the preconditions of RO?, this implies that we can only apply RO? to nominal nodes of level below λ . This, together with (2) above and some counting, yields the bound of $\mathcal{O}(\ell(mn)^\lambda)$.

The remainder of the proof is very similar to the *SHIQ* case. For the second claim in Lemma 6, for the “if” direction, we can obtain a tableau $T = (S, \mathcal{L}', \mathcal{E})$ from a complete and clash-free completion graph G by *unravelling* blockable “tree” parts of the graph as usual. For the “only if” direction, using a tableau $T = (S, \mathcal{L}', \mathcal{E})$ for D w.r.t. $\mathcal{R}$ to steer the non-deterministic rules RL , $R?$, $R\leq$, and $Ro?$, we can indeed construct a complete and clash-free graph. $\square$

Since subsumption can be reduced to (un)satisfiability and *SHOIQ* can internalise general TBoxes, we have the following result.

Theorem 7 The *SHOIQ* tableau algorithm is a decision procedure for satisfiability and subsumption of *SHOIQ* concepts w.r.t. TBoxes and role hierarchies.

5 Discussion

We have presented what is, to the best of our knowledge, the first goal-directed decision procedure for *SHOIQ* (and so *SHOIN*). Given that *SHOIQ* is NExpTime-complete [Tobies, 2000], it is clear that, in the worst case, any decision procedure will behave very badly, i.e., not terminate in practice. However, the algorithm given here is designed to behave well in many typically encountered cases, and to exhibit a “pay as you go” behaviour: if an input TBox, role hierarchy, and concept do not involve any one of inverse roles, number restrictions, or nominals, then $Ro?$ will not be applied, and the corresponding non-deterministic guessing is avoided. This is even true for inputs that do involve all of these three constructors, but only in a “harmless” way. Hence, our *SHOIQ* algorithm can be implemented to perform just as well on *SHIQ* knowledge bases as state-of-the-art DL reasoners for *SHIQ* [Horrocks & Patel-Schneider, 1998; Haarslev & Möller, 2001; Pellet, 2003]. To find out whether our algorithm can handle some non-trivial, “true” *SHOIQ* inputs, we are currently extending a highly optimised DL reasoner, FaCT++ [Tsarkov & Horrocks, 2004], to implement the algorithm described here.

References

- [Baader & Hollunder, 1991] F. Baader and B. Hollunder. Qualifying number restrictions in concept languages. In *Proc. of KR'91*. Morgan Kaufmann, 1991.
- [Bechhofer et al., 2004] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language reference. W3C Recommendation, 10 February 2004. Available at <http://www.w3.org/TR/owl-ref/>.
- [Blackburn & Seligman, 1995] P. Blackburn and J. Seligman. Hybrid languages. *J. of Logic, Language and Information*, 4, 1995.
- [Calvanese et al., 1998] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of KR'98*. Morgan Kaufmann, 1998.
- [Haarslev & Möller, 2001] V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR 2001*, vol. 2083 of *LNAI*. Springer, 2001.
- [Hladik & Model, 2004] J. Hladik and J. Model. Tableau systems for SHIO and SHIQ. In *Proc. of DL 2004*. CEUR, 2004. Available from ceur-ws.org.
- [Horrocks et al., 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, vol. 1705 of *LNAI*. Springer, 1999.
- [Horrocks et al., 2000] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic *SHIQ*. In *Proc. CADE 2000*, vol. 1831 of *LNCS*. Springer, 2000.
- [Horrocks et al., 2003] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From *SHIQ* and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.
- [Horrocks & Patel-Schneider, 1998] I. Horrocks and P. F. Patel-Schneider. FaCT and DLP: Automated reasoning with analytic tableaux and related methods. In *Proc. of TABLEAUX'98*, vol. 1397 of *LNAI*. Springer, 1998.
- [Horrocks & Sattler, 2001] I. Horrocks and U. Sattler. Ontology reasoning in the *SHOQ(D)* description logic. In *Proc. of IJCAI 2001*. Morgan Kaufmann, 2001.
- [Horrocks & Sattler, 2005] I. Horrocks and U. Sattler. A tableaux decision procedure for SHOIQ. available at www.cs.man.ac.uk/~sattler/ulis-ps.html, 2005.
- [McGuinness & Wright, 1998] D. L. McGuinness and J. R. Wright. An industrial strength description logic-based configuration platform. *IEEE Intelligent Systems*, 13(4), 1998.
- [Pacholski et al., 1997] L. Pacholski, W. Szawast, and L. Tendera. Complexity of two-variable logic with counting. In *Proc. of LICS'97*. IEEE Computer Society Press, 1997.
- [Pellet, 2003] Pellet OWL reasoner. Maryland Information and Network Dynamics Lab, 2003. <http://www.mindswap.org/2003/pellet/index.shtml>.
- [Schaerf, 1994] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2), 1994.
- [Tobies, 2000] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research*, 12, 2000.
- [Tobies, 2001] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.
- [Tsarkov & Horrocks, 2004] D. Tsarkov and I. Horrocks. Efficient reasoning with range and domain constraints. In *Proc. of DL 2004*. CEUR, 2004. Available from ceur-ws.org.
- [Vardi, 1997] M. Y. Vardi. Why is modal logic so robustly decidable. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 31. American Mathematical Society, 1997.

Reasoning with Inconsistent Ontologies *

Zhisheng Huang, Frank van Harmelen, and Annette ten Teije

Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands

{huang, Frank.van.Harmelen, annette}@cs.vu.nl

Abstract

In this paper we present a framework of reasoning with inconsistent ontologies, in which pre-defined selection functions are used to deal with concept relevance. We examine how the notion of "concept relevance" can be used for reasoning with inconsistent ontologies. We have implemented a prototype called PION (Processing Inconsistent ONtologies), which is based on a syntactic relevance-based selection function. In this paper, we also report the experiments with PION.

1 Introduction

There are two main ways to deal with inconsistency in ontologies. One is to diagnose and repair it when we encounter inconsistency [Schlobach and Cornet, 2003]. Another approach is to simply avoid the inconsistency and to apply a non-standard reasoning method to obtain meaningful answers. In this paper, we will focus on the latter, which is more suitable for the setting in the web area. For example, in a typical Semantic Web setting, one would be importing ontologies from other sources, making it impossible to repair them, and the scale of the combined ontologies may be too large to make repair effective.

The classical entailment in logics is *explosive*: any formula is a logical consequence of a contradiction. Therefore, conclusions drawn from an inconsistent knowledge base by classical inference may be completely meaningless. The general task of an inconsistency reasoner is: given an inconsistent ontology, return *meaningful* answers to queries.

Reasoning with inconsistency is a well-known topic in logics and AI. Many approaches have been proposed to deal with inconsistency [Benferhat and Garcia, 2002; Beziau, 2000; Lang and Marquis, 2001; Marquis and Porquet, 2003]. The development of paraconsistent logics was initiated to challenge the 'explosive' problem of the standard logics. Paraconsistent logics [Beziau, 2000] allow theories that are inconsistent but non-trivial. There are many different paraconsistent logics. Most of them are defined on a semantics which allows both a letter and its negation to hold for an interpretation. Levesque's limited inference [Levesque, 1989]

allows the interpretation of a language in which a truth assignment may map both a letter and its negation to true. Extending the idea of Levesque's limited inference, [Schaerf and Cadoli, 1995] proposes *S*-3-entailment and *S*-1-entailment for approximate reasoning with tractable results. Based on Schaerf and Cadoli's *S*-3-entailment, [Marquis and Porquet, 2003] presents a framework for reasoning with inconsistency by introducing a family of resource-bounded paraconsistent inference relations. Several policies, e.g., the linear order policy, are proposed. [Chopra *et al.*, 2000] incorporates the local change of belief revision and relevance sensitivity by means of Schaerf and Cadoli's approximate reasoning method, and shows how relevance can be introduced for approximate reasoning in belief revision. Both approaches, Marquis', and Chopra's, depend on syntactic selection procedures for extending the approximation set.

Our approach borrows some ideas from Schaerf and Cadoli's approximation approach, Marquis and Porquet's paraconsistent reasoning approach, and Chopra, Parikh, and Wassermann's relevance approach. However, our main idea is relatively *simple*: given a selection function, which can be defined on the syntactic or semantic relevance, we select some consistent sub-theory from an inconsistent ontology. Then we apply standard reasoning on the selected sub-theory to find meaningful answers. If a satisfying answer cannot be found, the relevance degree of the selection function is made less restrictive thereby extending the consistent subtheory for further reasoning.

The main contributions of this paper are: (1) a set of *formal definitions* to capture reasoning with inconsistent ontologies; (2) a *general framework* for reasoning with inconsistent ontologies based on selection functions, and (3) some *preliminary experiments* with an implementation of this framework using a rather simple selection function.

This paper is organised as follows: Section 2 overviews inconsistency in the Semantic Web by examining several typical examples and scenarios. Section 3 presents the framework of reasoning with inconsistent ontologies. Section 4 discusses the selection functions. Section 5 presents the algorithms of PION. Section 6 examines how the syntactic-relevance-based selection function can be developed. Section 7 describes a prototype of PION and its evaluation. Section 8 discusses further work and concludes the paper.

*The work reported in this paper was partially supported by the EU-funded SEKT project (IST-506826).

2 Inconsistency in the Semantic Web

Here are several typical scenarios which may cause inconsistencies in the Semantic Web.

- **Inconsistency Caused by mis-presentation of defaults.** A notorious example is the bird ontology in which penguins are specified as the birds which cannot fly, however, it contradicts the default statement 'birds are flying animals'. Another typical example is the Mad-Cows ontology¹ in which MadCow is specified as a Cow which eats brains of sheep, whereas a Cow is considered as a vegetarian by default.
- **Inconsistency Caused by Polysemy** Polysemy refers to the concept of words with multiple meanings. An example of an inconsistent ontology which is caused by polysemy is the MarriedWoman Example [Huang et al., 2004], in which the concept 'married woman' is used to refer both to a woman who has a husband and to a woman who *had* a husband but may no longer have one.
- **Inconsistency through Migration from Another Formalism** When an ontology specification is migrated from other data sources, inconsistencies may occur. As has been found in [Schlobach and Cornet, 2003], the DICE terminology (a DL terminology in the Intensive Care domain) suffered from a high number of unsatisfiable concepts due to its migration from a frame-based system. In order to make the semantics as explicit as possible, a very restrictive translation has been chosen to highlight as many ambiguities as possible. [Schlobach and Cornet, 2003] shows the inconsistent ontology specification in the Brain Example, in which a brain is considered to be both a body part and a central nervous system, whereas body parts and nervous systems are considered to be disjoint.
- **Inconsistency Caused by Multiple Sources.** When a large ontology specification is generated from multiple sources, in particular when these sources are created by several authors, inconsistencies easily occur. There are three possibilities for ontology reconciliation: merging, aligning, or integrating. No matter whether a new ontology is generated by merging or integrating multiple sources, in both cases general consistency objectives are rather difficult to achieve.

3 Formal Definitions

We do not restrict ontology specifications to a particular language (although OWL and its underlying description logic are the languages we have in mind). In general, an ontology language can be considered to be a set that is generated by a set of syntactic rules. Thus, we can consider an ontology specification as a formula set. We use a non-classical entailment for inconsistency reasoning. In the following, we use $\models$ to denote the classical entailment, and use $\approx$ to denote some non-standard inference relation, which may be parameterized to remove ambiguities.

With classical reasoning, a query ϕ given an ontology Σ can be expressed as an evaluation of the consequence relation

¹<http://www.daml.org/ontologies/399>

$\Sigma \models \phi$. There are only two answers to that query: either 'yes' ($\Sigma \models \phi$), or 'no' ($\Sigma \not\models \phi$). A 'yes' answer means that ϕ is a logical consequence of Σ . A 'no' answer, however, means that ϕ cannot be deduced from Σ , because we usually don't adopt the closed world assumption when using an ontology. Hence, a 'no' answer does not imply that the negation of ϕ holds given an ontology Σ . For reasoning with inconsistent ontologies, it is more suitable to use Belnap's four valued logic [Belnap, 1977] to distinguish the following four epistemic states of the answers:

Definition 1

- (a) *Over-determined*: $\Sigma \approx \phi$ and $\Sigma \approx \neg\phi$.
- (b) *Accepted*: $\Sigma \approx \phi$ and $\Sigma \not\approx \neg\phi$.
- (c) *Rejected*: $\Sigma \not\approx \phi$ and $\Sigma \approx \neg\phi$.
- (d) *Undetermined*: $\Sigma \not\approx \phi$ and $\Sigma \not\approx \neg\phi$.

For an inconsistency reasoner it is expected that is able to return meaningful answers to queries, given an inconsistent ontology. In the case of a consistent ontology Σ , classical reasoning is sound, i.e., a formula ϕ deduced from Σ holds in every model of Σ . This definition is not preferable for an inconsistent ontology Σ as every formula follows from it using classical entailment. However, often only a small part of Σ has been incorrectly constructed or modelled, while the remainder of Σ is correct. Therefore, we propose the following definition of soundness.

Definition 2 (Soundness) An inconsistency reasoner $\approx$ is sound if the formulas that follow from an inconsistent theory Σ follow from a consistent subtheory of Σ using classical reasoning, namely, the following condition holds:

$$\Sigma \approx \phi \Rightarrow (\exists \Sigma' \subseteq \Sigma)(\Sigma' \not\models \perp \text{ and } \Sigma' \models \phi).$$

In other words, the $\approx$ -consequences must be justifiable on the basis of a consistent subset of the theory. Note however, that in the previous definition the implication should *not hold* in the opposite direction. If the implication would also hold in the opposite direction it would lead to an inconsistency reasoner, which returns inconsistent answers. For example if $\{a, \neg a\} \subseteq \Sigma$, then the inconsistency reasoner would return that both a and $\neg a$ hold given Σ , which is something we would like to prevent. Hence, the inconsistency reasoner should not return answers that follow from *any* consistent subset of Σ , but from *specifically chosen* subsets of Σ . In other words, the selection of specific subsets on which $\approx$ will be based is an integral part of the definition of an inconsistency reasoner, and will be discussed in more detail in the next section.

Definition 3 (Meaningfulness) An answer given by an inconsistency reasoner is meaningful iff it is consistent and sound. Namely, it requires not only the soundness condition, but also the following condition:

$$\Sigma \approx \phi \Rightarrow \Sigma \not\approx \neg\phi.$$

An inconsistency reasoner is said to be meaningful iff all of the answers are meaningful.

Because of inconsistencies, classical completeness is impossible. We suggest the notion of local completeness:

Definition 4 (Local Completeness) An inconsistency reasoner is locally complete with respect to a consistent subtheory Σ' iff for any formula ϕ , the following condition holds:

$$\Sigma' \models \phi \Rightarrow \Sigma \approx \phi.$$

Since the condition can be represented as:

$$\Sigma \not\approx \phi \Rightarrow \Sigma' \not\models \phi,$$

local completeness can be considered as a complement to the soundness property. An answer to a query ϕ_i on Σ is said to be locally complete with respect to a consistent set Σ' iff the following condition holds:

$$\Sigma' \models \phi_i \Rightarrow \Sigma \approx \phi_i.$$

Definition 5 (Maximality) An inconsistency reasoner is maximal iff there is a maximal consistent subtheory such that its consequence set is the same as the consequence set of the inconsistency reasoner:

$$\exists(\Sigma' \subseteq \Sigma)((\Sigma' \not\models \perp) \wedge (\forall \Sigma'' \supset \Sigma' \wedge \Sigma'' \subseteq \Sigma)(\Sigma'' \models \perp) \wedge \forall \phi(\Sigma' \models \phi \Leftrightarrow \Sigma \approx \phi)).$$

We use the same condition to define the maximality for an answer, like we do for local completeness.

Definition 6 (Local Soundness) An answer to a query $\Sigma \approx \phi$ is said to be locally sound with respect to a consistent set $\Sigma' \subseteq \Sigma$, iff the following condition holds:

$$\Sigma \approx \phi \Rightarrow \Sigma' \models \phi.$$

Namely, for any positive answer, it should be implied by the given consistent subtheory Σ' under the standard entailment.

Proposition 3.1

- (a) Local Soundness implies Soundness and Meaningfulness.
- (b) Maximal completeness implies Local Completeness.

Given a query, there might exist more than one maximal consistent subset and more than one locally-complete consistent subset. Such different maximally consistent subsets may give different $\approx$ -consequences for a given query ϕ . Therefore, arbitrary (maximal) consistent subsets may not be very useful for the evaluation of a query by some inconsistency reasoner. The consistent subsets should be chosen on structural or semantic grounds indicating the relevance of the chosen subset with respect to some query.

4 Selection Functions

An inconsistency reasoner uses a selection function to determine which consistent subsets of an inconsistent ontology should be considered in its reasoning process. The general framework is independent of the particular choice of selection function. The selection function can either be based on a syntactic approach, like Chopra, Parikh, and Wassermann's syntactic relevance [Chopra *et al.*, 2000], or based on semantic relevance like for example in computational linguistics as in Wordnet [Budanitsky and Hirst, 2001].

Given an ontology (i.e., a formula set) Σ and a query ϕ , a selection function s is one which returns a subset of Σ at the step $k > 0$. Let $\mathbf{L}$ be the ontology language, which is denoted as a formula set. We have the general definition about selection functions as follows:

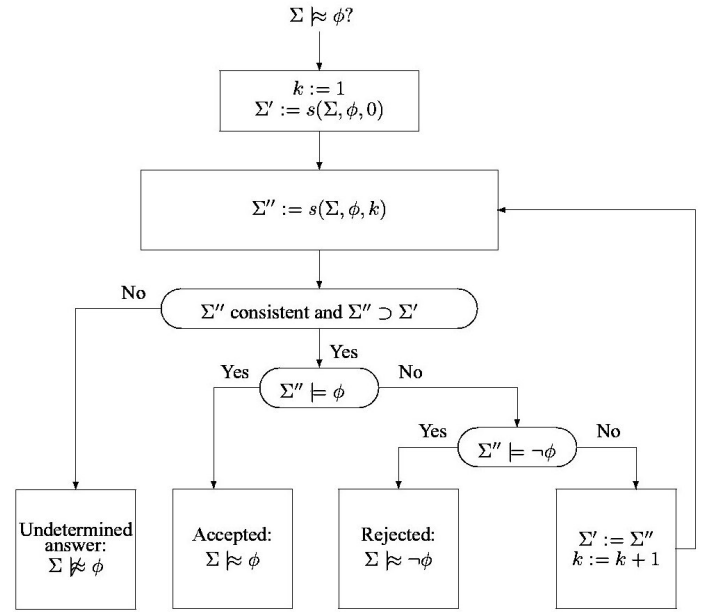


Figure 1: Linear Extension Strategy.

Definition 7 (Selection Functions) A selection function s is a mapping $s : \mathcal{P}(\mathbf{L}) \times \mathbf{L} \times \mathbf{N} \rightarrow \mathcal{P}(\mathbf{L})$ such that $s(\Sigma, \phi, k) \subseteq \Sigma$.

Definition 8 A selection function s is called monotonic if the subsets it selects monotonically increase or decrease, i.e., $s(\Sigma, \phi, k) \subseteq s(\Sigma, \phi, k + 1)$, or vice versa.

For monotonically increasing selection functions, the initial set is either an emptyset, i.e., $s(\Sigma, \phi, 0) = \emptyset$, or a fixed set Σ_0 when the locality is required. For monotonically decreasing selection functions, usually the initial set $s(\Sigma, \phi, 0) = \Sigma$. The decreasing selection functions will reduce some formulas from the inconsistent set step by step until they find a maximally consistent set.

Monotonically increasing selection functions have the advantage that they do not have to return *all* subsets for consideration at the same time. If a query $\Sigma \approx \phi$ can be answered after considering some consistent subset of the ontology Σ for some value of k , then other subsets (for higher values of k) don't have to be considered any more, because they will not change the answer of the inconsistency reasoner.

5 Strategies

An inconsistency reasoner that uses a monotonically increasing/decreasing selection function will be called an inconsistency reasoner that uses a *linear extension strategy* and a *linear reduction strategy* respectively.

A linear extension strategy is carried out as shown in Figure 1. Given a query $\Sigma \approx \phi$, the initial consistent subset Σ' is set. Then the selection function is called to return a consistent subset Σ'' , which extends Σ' , i.e., $\Sigma' \subset \Sigma'' \subseteq \Sigma$ for the linear extension strategy. If the selection function cannot find a consistent superset of Σ' , the inconsistency reasoner returns the answer 'undetermined' (i.e., unknown) to the query. If the

set Σ'' exists, a classical reasoner is used to check if $\Sigma'' \models \phi$ holds. If the answer is ‘yes’, the inconsistency reasoner returns the ‘accepted’ answer $\Sigma \approx \phi$. If the answer is ‘no’, the inconsistency reasoner further checks the negation of the query $\Sigma'' \models \neg\phi$. If the answer is ‘yes’, the inconsistency reasoner returns the ‘rejected’ answer $\Sigma \approx \neg\phi$, otherwise the current result is undetermined (def.1), and the whole process is repeated by calling the selection function for the next consistent subset of Σ which extends Σ'' .

It is clear that the linear extension strategy may result in too many ‘undetermined’ answers to queries when the selection function picks the wrong sequence of monotonically increasing subsets. It would therefore be useful to measure the successfulness of (linear) extension strategies. Notice, that this depends on the choice of the monotonic selection function.

In general, one should use an extension strategy that is not over-determined and not undetermined. For the linear extension strategy, we can prove that the following properties hold:

Proposition 5.1 (Linear Extension) *An inconsistency reasoner using a linear extension strategy satisfies the following properties:*

- (a) never over-determined,
- (b) may be undetermined,
- (c) always sound,
- (d) always meaningful,
- (e) always locally complete,
- (f) may not be maximal,
- (g) always locally sound.

Therefore, an inconsistency reasoner using a linear extension strategy is useful to create meaningful and sound answers to queries. It is always locally sound and locally complete with respect to a consistent set Σ_0 if the selection function always starts with the consistent set Σ_0 (i.e., $s(\Sigma, \phi, 0) = \Sigma_0$). Unfortunately it may not be maximal.

We call this strategy a *linear* one, because the selection function only follows one possible ‘extension chain’ for creating consistent subsets. The advantages of the linear strategy is that the reasoner can always focus on the current working set Σ' . The reasoner doesn’t need to keep track of the extension chain. The disadvantage of the linear strategy is that it may lead to an inconsistency reasoner that is undetermined. There exists other strategies which can improve the linear extension approach, for example, by backtracking and heuristics evaluation. We are going to discuss a backtracking strategy in Section 6. The second reason why we call the strategy linear is that the computational complexity of the strategy is linear with respect to the complexity of the ontology reasoning. Let n be the cardinality $|\Sigma|$ of an ontology Σ and let the complexity of $\models$ be E .

Proposition 5.2 (Complexity of Linear Extension) *The complexity of $\approx$ in the linear extension strategy is $n \cdot E$.*

In other words, the linear extension strategy does not significantly increase the complexity of the ontology reasoning, because typically E is already *PSPACE-complete* for standard concept languages [Donini, 2003].

6 Syntactic Relevance-based Selection Functions

[Chopra *et al.*, 2000] proposes syntactic relevance to measure the relationship between two formulas in belief sets, so that the relevance can be used to guide the belief revision based on Schaerf and Cadoli’s method of approximate reasoning. Given a formula set Σ , two atoms p, q are directly relevant, denoted by $R(p, q, \Sigma)$ iff there is a formula $\alpha \in \Sigma$ such that p, q appear in α . A pair of atoms p and q are k -relevant with respect to Σ iff there exist $p_1, p_2, \dots, p_k \in \mathcal{L}$ such that: (a) p, p_1 are directly relevant; (b) p_i, p_{i+1} are directly relevant, $i = 1, \dots, k-1$; and (c) p_k, q are directly relevant (i.e., directly relevant is k -relevant for $k = 0$).

The notions of relevance above are based on propositional logics. However, ontology languages are usually written in some fragment of first order logic. We extend the ideas of relevance to those first-order logic-based languages by restricting relevance to the co-occurrence of only the predicate letters or constant symbols. The following definition specialises the general definition of relevance for the case where ϕ is a formula in an ontology.

Given a formula ϕ , we use $I(\phi), C(\phi), R(\phi)$ to denote the sets of individual names, concept names, and relation names that appear in the formula ϕ respectively.

Definition 9 (Direct relevance) *Two formula ϕ, ψ are directly relevant iff there is a common name which appears both in formula ϕ and formula ψ , i.e., $I(\phi) \cap I(\psi) \neq \emptyset \vee C(\phi) \cap C(\psi) \neq \emptyset \vee R(\phi) \cap R(\psi) \neq \emptyset$.*

Definition 10 (Direct relevance to a set) *A formula ϕ is relevant to a formula set Σ iff there exists a formula $\psi \in \Sigma$ such that ϕ and ψ are directly relevant.*

We can similarly specialise the notion of k -relevance.

Definition 11 (k-relevance) *Two formulas ϕ, ϕ' are k -relevant with respect to a formula set Σ iff there exist formulas $\psi_0, \dots, \psi_k \in \Sigma$ such that ϕ and ψ_0, ψ_0 and $\psi_1, \dots$, and ψ_k and ϕ' are directly relevant.*

Definition 12 (k-relevance to a set) *A formula ϕ is k -relevant to a formula set Σ iff there exists a formula $\psi \in \Sigma$ such that ϕ and ψ are k -relevant with respect to Σ .*

In inconsistency reasoning we can use syntactic relevance to define a selection function s to extend the query ‘ $\Sigma \approx \phi?$ ’ as follows: We start with the query formula ϕ as a starting point for the selection based on syntactic relevance. Namely, we define:

$$s(\Sigma, \phi, 0) = \emptyset.$$

Then the selection function selects the formulas $\psi \in \Sigma$ which are directly relevant to ϕ as a working set (i.e. $k = 1$) to see whether or not they are sufficient to give an answer to the query. Namely, we define:

$$s(\Sigma, \phi, 1) = \{\psi \in \Sigma \mid \phi \text{ and } \psi \text{ are directly relevant}\}.$$

If the reasoning process can obtain an answer to the query, it stops. Otherwise the selection function increases the relevance degree by 1, thereby adding more formulas that are relevant to the current working set. Namely, we have:

$$s(\Sigma, \phi, k) = \{\psi \in \Sigma \mid \psi \text{ is directly relevant to } s(\Sigma, \phi, k-1)\},$$

for $k > 1$. This leads to a "fan out" behavior of the selection function: the first selection is the set of all formulae that are directly relevant to the query; then all formulae are selected that are directly relevant to that set, etc. This intuition is formalized in the following:

Proposition 6.1 *The syntactic relevance-based selection function s is monotonically increasing.*

Proposition 6.2 *If $k \geq 1$, then*

$$s(\Sigma, \phi, k) = \{\phi \mid \phi \text{ is } (k-1)\text{-relevant to } \Sigma\}$$

The syntactic relevance-based selection functions defined above usually grows up to an inconsistent set rapidly. That may lead to too many undetermined answers. In order to improve it, we can require that the selection function returns a consistent subset Σ'' at the step k when $s(\Sigma, \phi, k)$ is inconsistent such that $s(\Sigma, \phi, k-1) \subset \Sigma'' \subset s(\Sigma, \phi, k)$. It is actually a kind of backtracking strategies which are used to reduce the number of undetermined answers to improve the linear extension strategy. We call the procedure an *over-determined processing* (ODP) of the selection function. Note that the over-determined processing need no to exhaust the powerset of the set $s(\Sigma, \phi, k) - s(\Sigma, \phi, k-1)$, because of the fact that if a consistent set S cannot prove or disprove a query, then nor can any subset of S . Therefore, one approach of ODP is to return just a maximally consistent subset. Let n be $|\Sigma|$ and k be $n - |S|$, i.e., the cardinality difference between the ontology Σ and its maximal consistent subset S (note that k is usually very small), and let C be the complexity of the consistency checking. The complexity of the over-determined processing is polynomial to the complexity of the consistency checking:

Proposition 6.3 (Complexity of ODP) *The complexity of the over-determined processing is $n^k \cdot C$.*

Note that ODP introduces a degree of non-determinism: selecting different maximal consistent subsets of $s(\Sigma, \phi, k)$ may yield different answers to the query $\Sigma \models \phi$. The simplest example of this is $\Sigma = \{\phi, \neg\phi\}$.

7 Prototype of PION

7.1 Implementation

We are implementing the prototype of PION by using SWI-Prolog.² PION implements an inconsistency reasoner based on an linear extension strategy and the syntactic relevance-based selection function as discussed in Sections 5 and 6. The selection function returns the first maximal consistent subset for its over-determined processing. PION is powered by XDIG, an extended DIG Description Logic interface for Prolog [Huang and Visser, 2004]. PION supports the TELL requests in DIG data format and in OWL, and the ASK requests in DIG data format. A prototype of PION is available for download at the website: <http://wasp.cs.vu.nl/sekt/pion>.

7.2 Experiments and Evaluation

We have tested the prototype of PION by applying it on several example ontologies. These example ontologies are the bird example, the brain example, the MarriedWoman example, and the MadCow Ontolog, which are discussed in Section

²<http://www.swi-prolog.org>

2. We compare PION's answers with their intuitive answers which is supposed by a human to see to what extend PION can provide intended answers.

For a query, there might exist the following difference between an answer by PION and its intuitive answer.

- **Intended Answer:** PION's answer is the same as the intuitive answer;
- **Counter-intuitive Answer:** PION's answer is opposite to the intuitive answer. Namely, the intuitive answer is 'accepted' whereas PION's answer is 'rejected', or vice versa.
- **Cautious Answer:** The intuitive answer is 'accepted' or 'rejected', but PION's answer is 'undetermined'.
- **Reckless Answer:** PION's answer is 'accepted' or 'rejected' whereas the intuitive answer is 'undetermined'. We call it a reckless answer, because under this situation PION returns just one of the possible answers without seeking other possibly opposite answers, which may lead to 'undetermined'.

For each concept C in those ontologies, we create an instance 'the_ C ' on them. We make both a positive instance query and a negative instance query of the instance 'the_ C ' for some concepts C' in the ontologies, like a query 'is the_ C a C' ?'. PION test results are shown in Figure 2. Of the four test examples, PION can return at least 85.7% intended answers. Of the 396 queries, PION returns 24 cautious answers or reckless answers, and 2 counter-intuitive answers. However, we would like to point out that the high rate of the intended answers includes many 'undetermined' answers. One interesting (and we believe realistic) property of the Mad Cows Ontology is that many concepts which are intuitively disjoint (such as cows and sheep) are not actually declared as being disjoint (keep in mind that OWL has an open world semantics, and does not make the unique name assumption). As a result, many queries such as "is the_ cow a sheep" are indeed undetermined on the basis of the ontology, and PION correctly reports them as undetermined. The average time cost of the tested queries is about 5 seconds even on a low-end PC (with 550 mhz CPU, 256 MB memory under Windows 2000).

The counter-intuitive results occurs in the MadCows Example. PION returns the 'accepted' answer to the query 'is the_ mad_cow a vegetarian?'. This counter-intuitive answer results from the weakness of the syntactic relevance-based selection function, because it always prefers a shorter relevance path when a conflict occurs. In the mad cow example, the path 'mad cow - cow - vegetarian' is shorter than the path 'mad cow - eat brain - eat bodypart - sheep are animals - eat animal - NOT vegetarian'. Therefore, the syntactic relevance-based selection function finds a consistent sub-theory by simply ignoring the fact 'sheep are animals'. The problem results from the unbalanced specification between Cow and MadCow, in which Cow is directly specified as a vegetarian whereas there is no direct statement 'a MadCow is not a vegetarian'.

There are several alternative approaches to solve this kind of problems. One is to introduce the locality requirement. Namely, the selection function starts with a certain sub-theory which must always be selected. For example, the statement 'sheep are animals' can be considered to be a knowledge

Example	Queries	IA	CA	RA	CIA	IA Rate(%)	ICR Rate(%)
Bird	50	50	0	0	0	100	100
Brain	42	36	4	2	0	85.7	100
MarriedWoman	50	48	0	2	0	96	100
MadCow	254	236	16	0	2	92.9	99

IA = Intended Answers, CA = Cautious Answers, RA = Reckless Answers, CIA = Counter-Intuitive Answers, IA Rate = Intended Answers(%), ICR Rate = IA+CA+RA(%).

Figure 2: PION Test Results

statement which cannot be ignored. Another approach is to add a shortcut path, like the path 'mad cow - eat animal - NOT vegetarian' to achieve the relevance balance between the concepts 'vegetarian' and 'NOT vegetarian', as shown in the second mad cow example of PION testbed. The latter approach can be achieved automatically by accommodation of the semantic relevance from the user queries. The hypothesis is that both concepts appear in a query more frequently, when they are semantically more relevant. Therefore, from a semantic point of view, we can add a relevance shortcut path between strongly relevant concepts.

8 Discussion and Conclusions

In this paper, we have presented a framework for reasoning with inconsistent ontologies. We have introduced the formal definitions of the selection functions, and investigated the strategies of inconsistency reasoning processing based on a linear extension strategy.

One of the novelties of our approach is that the selection functions depend on individual queries. Our approach differs from the traditional one in paraconsistent reasoning, non-monotonic reasoning, and belief revision, in which a predefined preference ordering for all of the queries is required. This makes our approach more flexible, and less inefficient to obtain intended results. The selection functions can be viewed as ones creating query-specific preference orderings.

We have implemented and presented a prototype of PION. In this paper we have provided the evaluation report of the prototype by applying it to the several inconsistent ontology examples. The tests show that our approach can obtain intuitive results in most cases for reasoning with inconsistent ontologies. Considering the fact that standard reasoners always results in either meaningless answers or incoherence errors for queries on inconsistent ontologies, we can claim that PION can do much better, because it can provide a lot of intuitive, thus meaningful answers. This is a surprising result given the simplicity of our selection function.

In future work, we are going to test PION with more large-scale ontology examples. We are also going to investigate different approaches for selection functions (e.g., semantic-relevance based) and different extension strategies as alternatives to the linear extension strategy in combination with different selection functions, and test their performance.

References

[Belnap, 1977] N. Belnap. A useful four-valued logic. In *Modern Uses of Multiple-Valued Logic*, pages 8–37, Dordrecht, 1977. Reidel.

- [Benferhat and Garcia, 2002] S. Benferhat and L. Garcia. Handling locally stratified inconsistent knowledge bases. *Studia Logica*, pages 77–104, 2002.
- [Beziau, 2000] J.-Y. Beziau. What is paraconsistent logic. In *Frontiers of paraconsistent logic*, pages 95–111, Baldock, 2000. Research Studies Press.
- [Budanitsky and Hirst, 2001] A. Budanitsky and G. Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources*, Pittsburgh, PA., 2001.
- [Chopra et al., 2000] S. Chopra, R. Parikh, and R. Wassermann. Approximate belief revision- preliminary report. *Journal of IGPL*, 2000.
- [Donini, 2003] F. Donini. Complexity of reasoning. In *Description Logic Handbook*, pages 96–136, 2003.
- [Huang and Visser, 2004] Z. Huang and C. Visser. *Extended DIG Description Logic Interface Support for Prolog*. SEKT Deliverable D3.4.1.2, 2004.
- [Huang et al., 2004] Z. Huang, F. van Harmelen, A. ten Teije, P. Groot, and C. Visser. *Reasoning with Inconsistent Ontologies: framework and prototype*. SEKT Deliverable D3.4.1, 2004.
- [Lang and Marquis, 2001] J. Lang and P. Marquis. Removing inconsistencies in assumption-based theories through knowledge-gathering actions. *Studia Logica*, pages 179–214, 2001.
- [Levesque, 1989] H. Levesque. A knowledge-level account of abduction. In *Proceedings of IJCAI'89*, pages 1061–1067, 1989.
- [Marquis and Porquet, 2003] P. Marquis and N. Porquet. Resource-bounded paraconsistent inference. *Annals of Mathematics and Artificial Intelligence*, pages 349–384, 2003.
- [Schaerf and Cadoli, 1995] M. Schaerf and M. Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, pages 249–310, 1995.
- [Schlobach and Cornet, 2003] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of IJCAI 2003*, 2003.

Iterated Belief Change: A Transition System Approach

Aaron Hunter and James P. Delgrande

School of Computing Science

Faculty of Applied Sciences

Simon Fraser University

Burnaby, BC, Canada

{amhunter, jim}@cs.sfu.ca

Abstract

We use a transition system approach to reason about the evolution of an agent's beliefs as actions are executed. Some actions cause an agent to perform belief revision and some actions cause an agent to perform belief update, but the interaction between revision and update can be non-elementary. We present a set of basic postulates describing the interaction of revision and update, and we introduce a new belief evolution operator that gives a plausible interpretation to alternating sequences of revisions and updates.

1 Introduction

Formalisms for reasoning about action effects typically focus on the representation of actions that change the state of the world. However, several formalisms have been introduced for reasoning about actions that change the beliefs of an agent without altering the state of the world [Shapiro *et al.*, 2000; van Ditmarsch, 2002; Herzig *et al.*, 2004]. In order to reason about multiple actions in this context, it is necessary to consider sequences of alternating belief revisions and belief updates. However, to date there has been little explicit discussion about the formal properties of such sequences.

This paper makes two contributions to the existing work on epistemic action effects. The first contribution is the development of a transition system framework suitable for reasoning about belief change. The new framework provides a simple tool for reasoning about revision and update in a single formalism, and it facilitates the treatment of conditional updates. The second contribution is the presentation of a principled approach to the interaction between revision and update. There are plausible examples in which agents appear to revise a prior belief state in response to a new observation; such examples are difficult to represent in existing formalisms.

2 Background and Motivation

2.1 Belief Change

We distinguish two kinds of belief change. Belief revision occurs when an agent receives new information about a static world. The original approach to belief revision is the AGM approach [Alchourron *et al.*, 1985]. Due to limitations on

space, we do not review the approach here; instead, we simply assume that the reader is familiar with the AGM postulates. Belief update, on the other hand, is the process in which the beliefs of an agent are modified in response to a world that has changed. One standard approach to belief update is the Katsuno and Mendelzon approach, which follows the AGM tradition by introducing a set of rationality postulates for belief update [Katsuno and Mendelzon, 1992].

2.2 Reasoning about Action

We introduce some standard terminology for describing transition systems [Gelfond and Lifschitz, 1998]. An action signature is a pair $\langle \mathbf{F}, \mathbf{A} \rangle$ of non-empty sets, respectively called the set of *fluent symbols* and the set of *action symbols*. Informally, fluent symbols are propositional variables representing properties of the world that may change over time and action symbols are atomic symbols representing deterministic actions that may be performed. A *state* is an interpretation over $\mathbf{F}$, and a *transition system* is a directed graph where each node is labeled by a state and each edge is labeled by a set of action symbols. The edges in a transition system indicate how the fluents change values in response to the execution of individual actions. We use the capital letter A , possibly with subscripts, to range over actions. The notation A will be used to denote a finite sequence of actions of indeterminate length.

A *belief state* is a set of states. We can think of a belief state as expressing a proposition. Informally, a belief state is the set of states that an agent considers possible. In this paper, we use lower case Greek letters to denote belief states. We are interested in belief change resulting from two distinct kinds of actions: *ontic actions* and *epistemic actions*. Ontic actions are actions that change the state of the world, whereas epistemic actions change the beliefs of an agent without altering the state of the world. Informally, after executing an ontic action, an agent should update the current belief state on a point-wise basis. On the other hand, if an agent executes an epistemic action, then the current belief state should be revised.

2.3 The Basic Problem

As noted above, belief update occurs when an ontic action is performed. Hence, we define belief update operators that take two arguments: a belief state and an ontic action. Epistemic actions are identified with *observations*, which are simply sets of interpretations. As a result, revision operators also

take two arguments, each of which is a set of interpretations. Informally, the observation α provides evidence that the actual world is in α . Let $\diamond$ be an update operator and let $*$ be a revision operator. We are interested in giving a reasonable interpretation to sequences of the form

$$\kappa \diamond A_1 * \alpha_1 \diamond \dots \diamond A_n * \alpha_n.$$

There are intuitively plausible examples in which applying the operators iteratively results in an unsatisfactory result. In the next section, we introduce one such example.

2.4 Illustrative Example

We extend the litmus paper problem originally presented in [Moore, 1985]. In the original problem, there is a beaker containing either an acid or a base, and there is an agent holding a piece of litmus paper that can be dipped into the beaker to determine the contents. The litmus paper turns red if it is placed in an acid and it turns blue if it is placed in a base. We extend the problem by admitting the possibility that the paper is not litmus paper, instead it is just plain white paper.

Boutilier points out that the standard approach to belief change provides an unintuitive representation of this problem [Boutilier, 1995]. One issue is that the standard approach does not allow conditional action effects; an agent simply updates the initial belief state by the new color of the litmus paper. Intuitively, this seems incorrect because there are actually two independent belief changes that occur. First, the agent dips the paper in the beaker and projects each possible world to the outcome of the dipping action. Second, the agent looks at the paper and observes the new color. Hence, the problem involves an update followed by a revision.

Even if the belief change is broken into two steps, the standard approach is limited in that an agent can only revise the current belief state. Suppose that an agent initially believes that the paper is litmus paper, but then it remains white after dipping it in the beaker. In this case, the agent should conclude that the paper was never litmus paper to begin with. This indicates that it is sometimes necessary for agents to revise prior belief states in the face of new knowledge.

We will return to this example periodically as we introduce formal machinery that provides a more natural representation.

3 A Transition System Approach

3.1 Metric Transition Systems

Standard transition systems do not provide a useful basis for performing belief revision, because belief revision generally requires some notion of plausibility among states or formulas. In order to define a revision operator, we introduce a distance function between states. A *metric* over $2^{\mathbf{F}}$ is a function d that maps each pair of states to a non-negative real number, and satisfies the following conditions:

1. $d(w_1, w_2) = 0$ iff $w_1 = w_2$
2. $d(w_1, w_2) = d(w_2, w_1)$
3. $d(w_1, w_2) + d(w_2, w_3) \geq d(w_1, w_3)$.

We will generally be concerned with metrics that return only integral distances, so from here on we will use the term *metric* to refer to integer-valued metrics.

Definition 1 A *metric transition system* is a triple $\langle S, R, d \rangle$ where

1. $S \subseteq 2^{\mathbf{F}}$
2. $R \subseteq S \times \mathbf{A} \times S$
3. d is a metric on S

Informally, if w_1 is close to w_2 , then an agent considers w_1 to be a plausible alternative to w_2 .

3.2 Belief Update

In this section, we define belief update with respect to a transition system. Recall that we update a belief state by an action with effects given by a transition system. This contrasts with the standard approach, in which a belief state is updated by a formula. The advantage of our approach is that it facilitates the representation of actions with conditional effects.

Intuitively, after executing an action A , an agent updates the belief state by projecting every state s to the state s' that would result if they executed A in s .

Definition 2 Let $T = \langle S, R, d \rangle$ be a metric transition system. The *update function* $\diamond : 2^S \times \mathbf{A} \rightarrow 2^S$ is defined as follows

$$\alpha \diamond A = \{f \mid (e, A, f) \in R \text{ for some } e \in \alpha\}.$$

Note that the distance function d does not play any role in belief update.

We briefly illustrate that transition systems can be used to define a standard update operator in which a belief state is updated by a formula rather than an action. Given $\mathbf{F}$, let $\mathbf{A}$ be the set of action symbols of the form A_ϕ , where ϕ is a conjunction of literals over $\mathbf{F}$. Define $T_{\mathbf{F}}$ to be the transition system with $S = 2^{\mathbf{F}}$ and $RsA_\phi s'$ just in case $s' \models \phi$ and s' agrees with s on every atom that does not occur in ϕ . Informally, the action A_ϕ corresponds to an update by ϕ .

Proposition 1 The update operator obtained from $T_{\mathbf{F}}$ satisfies the Katsuno and Mendelzon postulates.

3.3 Belief Revision

With each metric transition system T , we associate a revision function. The revision function associated with T is the distance-based revision function defined in [Delgrande, 2004]. We choose this approach because it requires the introduction of relatively little formal machinery and, provided that d satisfies some natural conditions, this revision operator satisfies the AGM postulates.

Definition 3 Let $T = \langle S, R, d \rangle$ be a metric transition system. The *revision function* $*$: $2^S \times 2^S \rightarrow 2^S$ is defined as follows

$$\kappa * \alpha = \{w \in \alpha \mid \exists v_1 \in \kappa, \forall v_2 \in \alpha, \forall v_3 \in \kappa, d(w, v_1) \leq d(v_2, v_3)\}.$$

Hence, if an agent is in belief state κ , then $\kappa * \alpha$ is the set of all worlds that are minimally distant from some world in κ .

3.4 Litmus Paper Revisited

We revisit the litmus paper problem in the context of metric transition systems. The problem can be represented with the following action signature:

$$\langle \{Red, Blue, Acid, Litmus\}, \{dip\} \rangle.$$

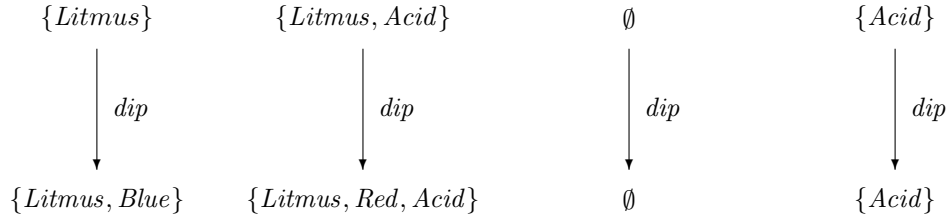


Figure 1: Extended Litmus Paper Problem

Intuitively, *Red* is true if the paper is red, *Blue* is true if the paper is blue, *Acid* is true if the beaker contains an acid, and *Litmus* is true if the paper is litmus paper. The effects of dipping are given by the transition system in Figure 1. We can extend this to a metric transition system by defining d to be the Hamming distance between states. Note also that we assume all actions are executable in all states, but self-loops are omitted from Figure 1 for ease of readability.

Recall that we are interested in an agent that initially believes they are holding a piece of white litmus paper. Hence, the initial belief state E is

$$E = \{\{Litmus\}, \{Litmus, Acid\}\}.$$

After dipping the paper, we update the belief state:

$$E \diamond dip = \{\{Litmus, Blue\}, \{Litmus, Red, Acid\}\}.$$

At this point, the agent observes that the paper is neither blue nor red. This observation is represented by the following set:

$$O = \{\emptyset, \{Litmus\}, \{Acid\}, \{Litmus, Acid\}\}.$$

The naive suggestion is to simply revise $E \diamond dip$ by O , giving

$$E' = \{\{Litmus\}, \{Litmus, Acid\}\}.$$

However, this is not a plausible final belief state. Given the transition system in Figure 1, it is clear that white litmus paper can not remain white after a dipping action. Hence, successively applying the update and revision operators obtained from the transition system does not yield a plausible result.

4 Rationality Postulates

In this section, we give a set of rationality postulates specifying some natural properties that we expect to hold when an update is followed by a revision. The postulates are not overly restrictive and they do not provide a basis for a categorical semantics; they simply provide a point for discussion and comparison. Our underlying intuition is that the most recent observation should always be incorporated, provided that it is consistent with the history of actions that have been executed. Hence, the postulates are only appropriate for action domains in which there are no failed actions.

Assume a fixed propositional signature F . Let κ and α be sets of worlds, and let $\bar{A}$ be a sequence of actions. We adopt the shorthand notation $\kappa \diamond \bar{A}$ as an abbreviation for the sequential update of κ by each element of $\bar{A}$. The following postulates describe some basic properties of the interaction between an update operator $\diamond$ and a revision operator $*$.

1. If $(2^F \diamond \bar{A}) \cap \alpha \neq \emptyset$, then $(\kappa \diamond \bar{A}) * \alpha \subseteq \alpha$
2. If $(2^F \diamond \bar{A}) \cap \alpha = \emptyset$, then $(\kappa \diamond \bar{A}) * \alpha = \kappa \diamond \bar{A}$
3. $(\kappa \diamond \bar{A}) \cap \alpha \subseteq (\kappa \diamond \bar{A}) * \alpha$
4. If $(\kappa \diamond \bar{A}) \cap \alpha \neq \emptyset$, then $(\kappa \diamond \bar{A}) * \alpha \subseteq (\kappa \diamond \bar{A}) \cap \alpha$
5. $(\kappa \diamond \bar{A}) * \alpha \subseteq 2^F \diamond \bar{A}$

We give some intuitive motivation for each postulate.

Postulate 1 is a straightforward AGM-type assertion that α must hold after revising by α , provided α is possible after executing $\bar{A}$. Postulate 2 handles the situation where it is impossible to be in an α -world after executing $\bar{A}$. In this case, we simply discard the observation α . These postulates together formalize the underlying assumption that there are no failed actions.

Taken together, Postulates 3 and 4 assert that revising by α is equivalent to taking the intersection with α , provided the intersection is non-empty. These postulates are similar to the AGM postulates asserting that revisions correspond to expansions, provided the observation is consistent with the knowledge base.

Postulate 5 provides the justification for revising prior belief states in the face of new knowledge. The postulate asserts that, after revising by α , we must still have a belief state that is a possible consequence of executing $\bar{A}$. In some cases, the only way to assure that α holds after executing $\bar{A}$ is to modify the initial belief state. We remark that the postulates do not indicate how the initial belief state should be modified.

5 Belief Evolution

5.1 Representing Histories

Transition systems are only suitable for representing Markovian action effects; that is, effects that are determined entirely by the current state and the action executed. However, examples like the litmus paper problem indicate that sometimes agents need to look at prior belief states as well. Hence, even if ontic action effects are Markovian, it does not follow that changes in belief are Markovian. As such, we need to introduce some formal machinery for representing histories.

Definition 4 A belief trajectory of length n is an n -tuple $\langle \kappa_0, \dots, \kappa_{n-1} \rangle$ of belief states.

Intuitively, a belief trajectory is an agent's subjective view of how the world has changed. We remark that a belief trajectory represents the agent's current beliefs about the world

history, not a historical account of what an agent believed at each point in time.

We will also be interested in observation trajectories and action trajectories, each of which is simply another n -tuple.

Definition 5 An observation trajectory of length n is an n -tuple $OBS = \langle OBS_1, \dots, OBS_n \rangle$ where each $OBS_i \in 2^S$.

Definition 6 An action trajectory of length n is an n -tuple $ACT = \langle ACT_1, \dots, ACT_n \rangle$ where each $ACT_i \in \mathbf{A}$.

Each set OBS_i is interpreted to be evidence that the actual world is in OBS_i at time i . An action trajectory is a history of the actions an agent has executed. Note that, as a matter of convention, we start the indices at 0 for belief trajectories and we start the indices at 1 for observation and action trajectories. The rationale for this convention will be clear later. We also adopt the convention hinted at in the definitions, whereby the n^{th} component of an action trajectory OBS will be denoted by OBS_n , and the n^{th} component of an action trajectory ACT will be denoted by ACT_n .

We define a notion of consistency between action trajectories and observation trajectories. The intuition is that an action trajectory ACT is consistent with an observation trajectory OBS if and only if each observation OBS_i is possible, given that the actions $(ACT_j)_{j \leq i}$ have been executed.

Definition 7 Let $ACT = \langle ACT_1, \dots, ACT_n \rangle$ be an action trajectory and let $OBS = \langle OBS_1, \dots, OBS_n \rangle$ be an observation trajectory. We say that ACT is consistent with OBS if and only if there is a belief trajectory $\langle \kappa_0, \dots, \kappa_n \rangle$ such that, for all $i \geq 1$,

1. $\kappa_i \subseteq OBS_i$
2. $\kappa_i = \kappa_{i-1} \diamond ACT_i$.

If ACT is consistent with OBS , we write $ACT \parallel OBS$.

A pair consisting of an action trajectory and an observation trajectory gives a complete history of all actions that have occurred. As such, it is useful to introduce some terminology.

Definition 8 A world view of length n is a pair $W = \langle ACT, OBS \rangle$, where OBS is an observation trajectory and ACT is an action trajectory, each of length n .

If $ACT \parallel OBS$, we say that $\langle ACT, OBS \rangle$ is consistent.

5.2 A New Belief Change Operator

We introduce a new operator $\circ$ that takes two arguments: a belief state and a world view. Roughly speaking, we would like $\kappa \circ \langle ACT, OBS \rangle$ to be the belief trajectory that results from the initial belief state κ and the alternating action-observation sequence $ACT_1, OBS_1, \dots, ACT_n, OBS_n$. We call $\circ$ a belief evolution operator because it takes a sequence of actions, and returns the most plausible evolution of the world.

The formal definition of $\circ$ is presented in the following sections. The definition relies on a fixed revision operator $*$ and a fixed update operator $\diamond$. As such, it might be more accurate to adopt notation of the form $\circ_{*,\diamond}$, but we opt for the less cumbersome $\circ$ and assume that the underlying operators are clear from the context. We remark, in particular, that every finite metric transition system generates a unique belief evolution operator. However, it is worth noting that the definition of $\circ$ does not rely on any specific approach to revision.

The action domains of interest for belief evolution will be those in which it is reasonable to assume that action trajectories are correct and actions are successful. This is intuitively plausible in a single agent environment, because it simply amounts to assuming that an agent has complete knowledge about the actions that they have executed. Hence, in the definition of $\circ$, the belief trajectory returned will always be consistent with the actions that have been executed.

5.3 Infallible Observations

In this section, we assume that observations are always correct. Formally, this amounts to a restriction on the class of admissible world views. In particular, we need not consider inconsistent world views.

We need to introduce a bit of notation.

Definition 9 Let T be a transition system; let $\bar{A} = A_0, \dots, A_n$; and let α be a set of states. Then $\alpha^{-1}(\bar{A})$ denotes the set of all w such that there is a path from w to an element of α following the edges $A_0, \dots, A_n$.

Hence, $\alpha^{-1}(\bar{A})$ is the set of states that can precede a world in α , given that the sequence $\bar{A}$ has been executed.

For illustrative purposes, it is useful to consider world views of length 1. Suppose we have an initial belief state κ , an ontic action A and an epistemic action α . Without formally defining the belief evolution operator $\circ$, we can give an intuitive interpretation of an expression of the form

$$\kappa \circ \langle \langle A \rangle, \langle \alpha \rangle \rangle = \langle \kappa_0, \kappa_1 \rangle.$$

The agent knows that the actual world is in α at the final point in time, so we must have $\kappa_1 \subseteq \alpha$. Moreover, the agent should believe that κ_1 is a possible result of executing A from κ_0 . In other words, we must have $\kappa_0 \subseteq \alpha^{-1}(A)$. All other things being equal, the agent would like to keep as much of κ as possible; therefore, the natural solution is the following:

1. $\kappa_0 = \kappa * \alpha^{-1}(A)$,
2. $\kappa_1 = \kappa_0 \diamond A$.

This procedure can be applied to world views of length greater than 1. The idea is to trace every observation back to a precondition on the initial belief state. After revising the initial belief state by all preconditions, each subsequent belief state can be determined by a standard update operation.

We have the following formal definition for $\circ$. In the definition, let $\overline{ACT}_i$ denote the subsequence of actions $ACT_1, \dots, ACT_i$.

Definition 10 Let κ be a belief state, let ACT be an action trajectory of length n and let OBS be an observation trajectory of length n such that $ACT \parallel OBS$. Define

$$\kappa \circ \langle ACT, OBS \rangle = \langle \kappa_0, \dots, \kappa_n \rangle$$

where

1. $\kappa_0 = \kappa * \bigcap_i OBS_i^{-1}(\overline{ACT}_i)$
2. for $i \geq 1$, $\kappa_i = \kappa_{i-1} \diamond ACT_1 \diamond \dots \diamond ACT_i$.

We remark that the intersection of observation preconditions in the definition of κ_0 is non-empty, because $ACT \parallel OBS$.

The following propositions are immediate, and they demonstrate that $\circ$ subsumes both revision and update. In each proposition, we assume that $ACT \parallel OBS$.

Proposition 2 Let κ be a belief state, let $ACT = \langle A \rangle$ and let $OBS = \langle 2^F \rangle$. Then

$$\kappa \circ \langle ACT, OBS \rangle = \langle \kappa, \kappa \diamond A \rangle.$$

In the following, we assume that λ is a null action that never changes the state of the world.

Proposition 3 Let κ be a belief state, let $ACT = \langle \lambda \rangle$ and let $OBS = \langle \alpha \rangle$. Then

$$\kappa \circ \langle ACT, OBS \rangle = \langle \kappa * \alpha, \kappa * \alpha \rangle.$$

Hence, both revision and update can be represented through the $\circ$ operator. As such, it is reasonable to adopt the following notation for trajectories of length 1:

$$\kappa \diamond ACT * OBS = \kappa \circ \langle \langle ACT \rangle, \langle OBS \rangle \rangle.$$

Proposition 4 If $ACT || OBS$, the operators $*$ and $\diamond$ (defined above) satisfy the interaction postulates (1)-(5).

The three preceding propositions demonstrate the suitability of $\circ$ as a natural operator for reasoning about the interaction between revision and update.

5.4 Fallible Observations

We address fallible observations by allowing inconsistent world views. If a world view is inconsistent, then there is no initial belief state that supports all of the observations, so some observations need to be ignored. To deal with inconsistency, we adopt the convention previously explored in [Nayak, 1994] and [Papini, 2001], in which more credence is given to the most recent observations. We demonstrate that $\circ$ can be extended naturally to represent belief change under this convention. In order to state the extended definition, we need to be able to extract a maximally consistent sub-view from an inconsistent world view.

Definition 11 Let $W = \langle ACT, OBS \rangle$ be a world view of length n . Define $\tau(W) = \langle ACT, OBS' \rangle$, where $OBS' = \langle OBS'_1, \dots, OBS'_n \rangle$ is defined by the following recursion.

- If $OBS_n^{-1}(ACT) \neq \emptyset$ then $OBS'_n = OBS_n$, otherwise $OBS'_n = 2^F$.
- For $i < n$, if

$$OBS_i^{-1}(\overline{ACT_i}) \cap \bigcap_{j>i} (OBS'_j)^{-1}(\overline{ACT_j}) \neq \emptyset$$

$$\text{then } OBS'_i = OBS_i, \\ \text{otherwise } OBS'_i = 2^F.$$

The observations in $\tau(W)$ are determined by starting with the most recent observation, then working backwards through the observations from most recent until the initial observation. At each point, we keep an observation if it is consistent with the observations that followed it; otherwise, we discard the observation as incorrect.

The following properties are immediate.

- If W is a world view, then $\tau(W)$ is consistent.
- If W is a consistent world view, then $\tau(W) = W$.

Recall that the original definition of $\circ$ applied only to consistent world views. By passing through τ , we can extend the definition to apply to arbitrary world views.

Definition 12 Let κ be a belief state, and let W be a world view of length n . If W is inconsistent, then $\kappa \circ W = \kappa \circ \tau(W)$.

We could equivalently have stated a single definition for $\circ$ by passing all world views through τ . We have presented the definition in two cases in order to highlight the distinct treatment of fallible observations. The extended definition is still satisfactory from the perspective of the rationality postulates.

Proposition 5 The operators $*$ and $\diamond$ (obtained from $\circ$) satisfy the interaction postulates (1)-(5).

Thus far, applying the $\circ$ operator requires tracing action preconditions back to the initial state for revision, then applying action effects to get a complete history. If we are only concerned with the final belief state, then there are many cases in which we do not need to go to so much effort.

Proposition 6 Let κ be a belief state, let ACT be an action trajectory of length n and let α be a belief state such that $\alpha \subseteq \kappa \diamond ACT$. If OBS is the observation trajectory with $n - 1$ null observations followed by α , then the final belief state in $\kappa \circ \langle ACT, OBS \rangle$ is $(\kappa \diamond ACT) * \alpha$.

The proposition indicates that, given a single observation that is consistent with the actions that have been executed, we can simply revise the outcome of the actions and we get the correct final belief state.

5.5 Litmus Paper Concluded

We conclude the litmus paper example by giving a plausible treatment based on a belief evolution operator. The world view $WV = \langle \langle dip \rangle, \langle O \rangle \rangle$ represents a dipping action followed by the observation that the paper is still white. If $\circ$ is defined from the metric transition system in Figure 1, the final belief state in $E \circ WV$ is given by

$$\begin{aligned} E * O^{-1}(dip) \diamond dip &= E * \{\emptyset, \{Acid\}\} \diamond dip \\ &= \{\emptyset, \{Acid\}\}. \end{aligned}$$

This calculation is consistent with our original intuitions, in that the agent revises the initial belief state before updating by the *dip* action. This ensures that we will have a final belief state that is a possible outcome of dipping. Moreover, the initial belief state is revised by the pre-image of the final observation, which means it is modified as little as possible while still guaranteeing that the final observation will be feasible. Note also that the final belief state given by this calculation is intuitively plausible. It simply indicates that the contents of the beaker are still unknown, but the agent now believes the paper is not litmus paper. Hence, a belief evolution operator employs a plausible procedure and returns a desirable result.

6 Relationship with Iterated Revision

If the null action is the only action permitted, then belief evolution is closely related to iterated revision. We briefly consider the suitability of belief evolution operators for reasoning about iterated revision. In the following proposition, let λ denote a sequence of null actions of indeterminate length.

Proposition 7 For any κ and OBS , there is a unique belief state κ' such that $\kappa \circ \langle \lambda, OBS \rangle = \langle \kappa', \dots, \kappa' \rangle$.

This result is consistent with the view that belief evolution operators return a trajectory representing an agent's current beliefs about the evolution of the world. We remark that, in general, κ' is not obtained by successively revising by the elements of OBS . Moreover, we claim that this is appropriate for action domains in which recency determines the plausibility of an observation.

Darwiche and Pearl present a set of four postulates for iterated revision [Darwiche and Pearl, 1997]. It is easy to verify that many belief revision operators do not satisfy the Darwiche-Pearl postulates if applied in succession. However, if we define the iterated revision $\kappa * OBS_1 * \dots * OBS_n$ to be the unique belief state in $\kappa \circ \langle \bar{\lambda}, OBS \rangle$, then we get the following result.

Proposition 8 *Iterated revision, as defined by any belief evolution operator $\circ$, satisfies all four Darwiche-Pearl postulates.*

Hence, given any belief revision operator, we can formulate an adequate approach to iterated revision by passing to the corresponding evolution operator.

7 Discussion

We make some brief remarks about related formalisms for reasoning about epistemic action effects. Our formalism is similar to the multi-agent belief structures of [Herzig *et al.*, 2004] in that both approaches combine revision with update based on actions with conditional effects. However, multi-agent belief structures do not consider any non-elementary interaction between revision and update. On the other hand, the Situation Calculus approach of [Shapiro *et al.*, 2000] implicitly handles the interaction of revision and update correctly. The “revision actions” employed in that formalism are provably equivalent to belief evolution operators. Correctly identifying these actions as evolution operators improves the intuitive plausibility of iterated belief change in the Situation Calculus approach.

The framework presented in this paper requires an agent to believe the most recent observation. A more general approach would attach a plausibility value to every observation, and then an agent would keep the most plausible observations. We are currently working on such a generalization. Similarly, we are looking at attaching plausibilities to action histories in order to represent domains in which an agent is uncertain about the actions that have been executed.

8 Conclusion

We have presented a transition system framework for reasoning about the epistemic effects of actions. We identified ontic action effects with belief update and epistemic action effects with belief revision, and we focused on the interaction between iterated update and revision operators. We illustrated by example that the interaction between update and revision can be non-elementary, so we proposed a set of rationality postulates restricting the interaction and demonstrated that the postulates all hold in our transition system framework. By contrast, existing formalisms for reasoning about epistemic action effects either ignore the interaction between

revision and update or they deal with it implicitly. Hence, our formalism contributes to the existing work on epistemic action effects in two ways. First, it is able to provide simple object level representations of action domains in which prior belief states need to be revised. Second, it provides an explicit treatment of the interaction between revision and update, which has not always been salient in related formalisms.

References

- [Alchourron *et al.*, 1985] C.E. Alchourron, P. Gardenfors, and D. Makinson. On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic*, 50(2):510–530, 1985.
- [Boutilier, 1995] C. Boutilier. Generalized update: Belief change in dynamic settings. In *Proceedings of IJCAI 1995*, pages 1550–1556, 1995.
- [Darwiche and Pearl, 1997] A. Darwiche and J. Pearl. On the logic of iterated belief revision. *Artificial Intelligence*, 89(1-2):1–29, 1997.
- [Delgrande, 2004] J.P. Delgrande. Preliminary considerations on the modelling of belief change operators by metric spaces. In *Proceedings of NMR 2004*, pages 118–125, June 2004.
- [Gelfond and Lifschitz, 1998] M. Gelfond and V. Lifschitz. Action languages. *Linköping Electronic Articles in Computer and Information Science*, 3(16):1–16, 1998.
- [Herzig *et al.*, 2004] A. Herzig, J. Lang, and P. Marquis. Revision and update in multi-agent belief structures. In *Proceedings of LOFT 6*, 2004.
- [Katsuno and Mendelzon, 1992] H. Katsuno and A.O. Mendelzon. On the difference between updating a knowledge base and revising it. In Peter Gardenfors, editor, *Belief Revision*, pages 183–203. Cambridge University Press, 1992.
- [Moore, 1985] R.C. Moore. A formal theory of knowledge and action. In J.R. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex Publishing, 1985.
- [Nayak, 1994] A.C. Nayak. Iterated belief change based on epistemic entrenchment. *Erkenntnis*, 41:353–390, 1994.
- [Papini, 2001] O. Papini. Iterated revision operations stemming from the history of an agent's observations. In H. Rott and M. Williams, editors, *Frontiers in Belief Revision*, pages 279–301. Kluwer Academic Publishers, 2001.
- [Shapiro *et al.*, 2000] S. Shapiro, M. Pagnucco, Y. Lesperance, and H.J. Levesque. Iterated belief change in the situation calculus. In *Proceedings of KR 2000*, pages 527–538. Morgan Kaufmann Publishers, 2000.
- [van Ditmarsch, 2002] H.P. van Ditmarsch. Descriptions of game actions. *Journal of Logic, Language and Information (JoLLI)*, 11:349–365, 2002.

Data Complexity of Reasoning in Very Expressive Description Logics

Ullrich Hustadt

Dept. of Computer Science
Univ. of Liverpool, UK
U.Hustadt@csc.liv.ac.uk

Boris Motik

Forschungszentrum Informatik
Univ. of Karlsruhe, Germany
motik@fzi.de

Ulrike Sattler

Dept. of Computer Science
Univ. of Manchester, UK
sattler@cs.man.ac.uk

Abstract

Data complexity of reasoning in description logics (DLs) estimates the performance of reasoning algorithms measured in the size of the ABox only. We show that, even for the very expressive DL *SHIQ*, satisfiability checking is data complete for NP. For applications with large ABoxes, this can be a more accurate estimate than the usually considered *combined complexity*, which is EXPTIME-complete. Furthermore, we identify an expressive fragment, Horn-*SHIQ*, which is data complete for P, thus being very appealing for practical usage.

1 Introduction

Description logics (DLs) [Baader *et al.*, 2003] are state-of-the-art knowledge representation formalisms with applications in many areas of computer science. Very expressive DLs such as *SHIQ* are interesting mainly due to their high expressivity combined with the clearly defined model-theoretic semantics and known formal properties, such as the computational complexity of reasoning. In particular, the *combined complexity* of checking satisfiability of a *SHIQ* knowledge base *KB* is EXPTIME-complete in $|KB|$ [Schild, 1991; Tobies, 2001].

EXPTIME-completeness is a rather discouraging result since $|KB|$ can be large in practice. However, similar to a database, *KB* consists of a *schema* part *T*, called the *TBox*, and a data or fact part *A*, called the *ABox*. For applications with a fixed schema but varying data part, *data complexity*, measured in the size of *A* only, provides a more precise performance estimate.

Here, we assume that the ABox of *KB* is *extensionally reduced*, i.e. it involves only roles and (possibly negated) atomic concepts. Thus, the terminological knowledge is strictly separated from assertional knowledge, so $|A|$ is the measure of “raw” data. For such a *KB*, we show that checking *KB* satisfiability is NP-complete in $|A|$, and that instance checking is co-NP-complete in $|A|$. Since this might still lead to intractability, we identify Horn-*SHIQ*, a fragment of *SHIQ* analogous to the Horn fragment of first-order logic. Namely, Horn-*SHIQ* provides existential and universal quantifiers, but does not provide means to express disjunctive information. We show that, for Horn-*SHIQ*, the basic

reasoning problems are P-complete in $|A|$. To develop an intuition and to provide a more detailed account of these results, we compare them with similar results for (variants of) datalog [Dantsin *et al.*, 2001].

Our results are important since they provide a formal justification for hoping to provide tractable algorithms for very expressive description logics. Furthermore, Horn-*SHIQ* subsumes DL-lite [Calvanese *et al.*, 2004], a logic aiming to capture most constructs of ER and UML formalisms, while providing polynomial algorithms for satisfiability checking and conjunctive query answering (assuming an unbounded knowledge base, but a bound on the query size). Horn-*SHIQ* additionally allows qualified existential quantification, conditional functionality and role inclusions, while allowing for a reasoning algorithm that runs in time polynomial in the size of data.

In [Schaerf, 1994], the complexity of concept subsumption was contrasted with combined and data complexity of instance checking for various fragments of *ALC*. This work provides a lower bound for the data complexity of reasoning in *SHIQ*.

2 Preliminaries

Description Logics. Given a set of role names N_R , a *SHIQ* role is either some $R \in N_R$ or an *inverse role* R^- for $R \in N_R$. A *SHIQ* RBox $\mathcal{R}$ is a finite set of role inclusion axioms $R \sqsubseteq S$ and transitivity axioms $\text{Trans}(R)$, for R and S *SHIQ* roles. For $R \in N_R$, we set $\text{Inv}(R) = R^-$ and $\text{Inv}(R^-) = R$, and assume that $R \sqsubseteq S \in \mathcal{R} \ (\text{Trans}(R) \in \mathcal{R})$ implies $\text{Inv}(R) \sqsubseteq \text{Inv}(S) \in \mathcal{R} \ (\text{Trans}(\text{Inv}(R)) \in \mathcal{R})$. A role R is said to be *simple* if $\text{Trans}(S) \notin \mathcal{R}$, for each $S \sqsubseteq^* R$, where $\sqsubseteq^*$ is the reflexive-transitive closure of $\sqsubseteq$.

Given a set of *concept names* N_C , the set of *SHIQ* concepts is the minimal set such that each $A \in N_C$ is a *SHIQ* concept and, for C and D *SHIQ* concepts, R a role, S a simple role and n a positive integer, then $\neg C$, $C \sqcap D$, $\forall R.C$ and $\geq n S.C$ are also *SHIQ* concepts. We use $\top$, $\perp$, $C_1 \sqcup C_2$, $\exists R.C$, and $\leq n S.C$ as abbreviations for $A \sqcup \neg A$, $A \sqcap \neg A$, $\neg(\neg C_1 \sqcap \neg C_2)$, $\neg \forall R. \neg C$, and $\neg(\geq (n+1) S.C)$, respectively. Concepts that are not concept names are called *complex*. A *literal* concept is a possibly negated concept name.

A TBox $\mathcal{T}$ is a finite set of concept inclusion axioms of the form $C \sqsubseteq D$. An ABox $\mathcal{A}$ is a finite set of axioms $C(a)$, $R(a, b)$, and (in)equalities $a \approx b$ and $a \not\approx b$. A knowledge

Table 1: Semantics of $\mathcal{SHIQ}$ by Mapping to FOL

Translating Concepts to FOL	
$\pi_y(A, X) = A(X)$	
$\pi_y(C \sqcap D, X) = \pi_y(C, X) \wedge \pi_y(D, X)$	
$\pi_y(\neg C, X) = \neg \pi_y(C, X)$	
$\pi_y(\forall R.C, X) = \forall y : R(X, y) \rightarrow \pi_x(C, y)$	
$\pi_y(\geq n S.C, X) =$ $\exists y_1, \dots, y_n : \bigwedge S(X, y_i) \wedge \bigwedge \pi_x(C, y_i) \wedge \bigwedge y_i \not\approx y_j$	
Translating Axioms to FOL	
$\pi(C \sqsubseteq D) = \forall x : \pi_y(C, x) \rightarrow \pi_y(D, x)$	
$\pi(R \sqsubseteq S) = \forall x, y : R(x, y) \rightarrow S(x, y)$	
$\pi(\text{Trans}(R)) = \forall x, y, z : R(x, y) \wedge R(y, z) \rightarrow R(x, z)$	
$\pi(C(a)) = \pi_y(C, a)$	
$\pi(R(a, b)) = R(a, b)$	
$\pi(a \circ b) = a \circ b \text{ for } \circ \in \{\approx, \not\approx\}$	
Translating KB to FOL	
$\pi(R) = \forall x, y : R(x, y) \leftrightarrow R^-(y, x)$	
$\pi(KB) = \bigwedge_{R \in N_R} \pi(R) \wedge \bigwedge_{\alpha \in T \cup R \cup A} \pi(\alpha)$	
X is a meta variable and is substituted by the actual variable. π_x is obtained from π_y by simultaneously substituting in the definition $x_{(i)}$ for all $y_{(i)}$, respectively, and π_y for π_x .	

base KB is a triple $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. KB is *extensionally reduced* if all ABox axioms of KB contain only literal concepts.

The semantics of KB is given by translating it into first-order logic by the operator π from Table 1. The main inference problem is checking KB satisfiability, i.e. determining if a first-order model of $\pi(KB)$ exists. An individual a is an *instance* of a concept C w.r.t. KB if $\pi(KB) \models \pi_y(C, a)$, which is the case iff $KB \cup \{\neg C(a)\}$ is unsatisfiable.

The logic $\mathcal{ALCHIQ}$ is obtained by disallowing transitivity axioms in $\mathcal{SHIQ}$ RBoxes, and $\mathcal{ALC}$ is obtained by disallowing RBoxes, inverse roles and number restrictions. A logic $\mathcal{L}$ is *between* logics $\mathcal{L}_1$ and $\mathcal{L}_2$ if it contains at least the primitives from $\mathcal{L}_1$ and at most the primitives from $\mathcal{L}_2$.

We measure the *size* of concepts by their length, and assume *unary coding of numbers*, i.e. $|\leq n R.C| = n + 1 + |C|$, and use $|R(a, b)| = |(\neg)A(a)| = 3$.

Disjunctive Datalog. A *datalog term* is a constant or a variable, and a *datalog atom* has the form $A(t_1, \dots, t_n)$ or $t_1 \approx t_2$, where t_i are datalog terms. A *disjunctive datalog program with equality* P is a finite set of rules of the form $A_1 \vee \dots \vee A_n \leftarrow B_1, \dots, B_m$ where A_i and B_j are datalog atoms. Each rule is required to be *safe*, i.e. each variable occurring in the rule must occur in at least one body atom. A *fact* is a rule with $m = 0$. For the semantics, we take a rule to be equivalent to a clause $A_1 \vee \dots \vee A_n \vee \neg B_1 \vee \dots \vee \neg B_m$. We consider only Herbrand models, and say that a model M of P is *minimal* if there is no model M' of P such that $M' \subsetneq M$. A ground literal A is a *cautious answer* of P (written $P \models_c A$) if A is true in all minimal models of P . First-order entailment coincides with cautious entailment for positive ground atoms.

Reducing KB to Disjunctive Datalog. The results in this paper are based on our algorithm from [Hustadt et al., 2004b]. For a $\mathcal{SHIQ}$ knowledge base KB , this algorithm computes a

Table 2: Types of $\mathcal{ALCHIQ}$ -clauses

1	$\neg R(x, y) \vee \text{Inv}(R)(y, x)$
2	$\neg R(x, y) \vee S(x, y)$
3	$\mathbf{P}^f(x) \vee R(x, f(x))$
4	$\mathbf{P}^f(x) \vee R(f(x), x)$
5	$\mathbf{P}_1(x) \vee \mathbf{P}_2(f(x)) \vee \bigvee f_i(x) \approx / \not\approx f_j(x)$
6	$\mathbf{P}_1(x) \vee \mathbf{P}_2(g(x)) \vee \mathbf{P}_3(f(g(x))) \vee \bigvee t_i \approx / \not\approx t_j$ where t_i and t_j are of the form $f(g(x))$ or x
7	$\mathbf{P}_1(x) \vee \bigvee \neg R(x, y_i) \vee \mathbf{P}_2(\mathbf{y}) \vee \bigvee y_i \approx y_j$
8	$\mathbf{R}(\mathbf{a}, \mathbf{b}) \vee \mathbf{P}(\mathbf{t}) \vee \bigvee t_i \approx / \not\approx t_j$ where $t_{(i)}$ are a constant b or a functional term $f_i(a)$

positive disjunctive datalog program with equality $\text{DD}(KB)$ which is equisatisfiable with KB .

A minor obstacle in computing $\text{DD}(KB)$ are the transitivity axioms which, in their clausal form, do not contain a literal in which all variables of a clause occur. Such clauses are known to be difficult to handle, so KB is preprocessed into an equisatisfiable $\mathcal{ALCHIQ}$ knowledge base $\Omega(KB)$. Roughly speaking, a transitivity axiom $\text{Trans}(S)$ is replaced with axioms of the form $\forall R.C \sqsubseteq \forall S.(\forall S.C)$, for each R with $S \sqsubseteq^* R$ and C a concept occurring in KB . This transformation is polynomial, so in the rest of this paper w.l.o.g. we assume KB to be an $\mathcal{ALCHIQ}$ knowledge base.

The next step is to translate $\Omega(KB)$ into clausal first-order logic. Assuming π is defined as in Table 1, $\pi(\Omega(KB))$ is transformed into a set of clauses $\Xi(KB)$ using *structural transformation* to avoid an exponential blowup [Nonnengart and Weidenbach, 2001]. Roughly speaking, the structural transformation introduces a new name for each complex subformula of $\pi(\Omega(KB))$. A specialized version of the structural transformation is presented in detail in Section 4.

A core property of $\Xi(KB)$ is that it only contains clauses of one of the forms given in Table 2; such clauses are called $\mathcal{ALCHIQ}$ -clauses. For a term t , $\mathbf{P}(t)$ denotes a disjunction of the form $(\neg)P_1(t) \vee \dots \vee (\neg)P_n(t)$, and $\mathbf{P}(f(x))$ denotes a disjunction of the form $\mathbf{P}_1(f_1(x)) \vee \dots \vee \mathbf{P}_n(f_n(x))$ (notice that this allows each $\mathbf{P}_i(f_i(x))$ to contain positive and negative literals).

Next, the RBox and TBox clauses of $\Xi(KB)$ are saturated by basic superposition [Bachmair et al., 1995; Nieuwenhuis and Rubio, 1995]—a clausal equational theorem proving calculus. Due to space limitations, we are unable to present the rules of basic superposition; it can be considered to be an optimized version of the well-known paramodulation calculus. Let Γ be the saturated set of clauses. In this key step, all non-ground consequences of KB are computed. In [Hustadt et al., 2004b], we have shown the following key property: (♠) an application of a basic superposition inference rule to $\mathcal{ALCHIQ}$ -clauses produces an $\mathcal{ALCHIQ}$ -clause. The proof examines all inference rules and clause types.

Furthermore, by examining the types of clauses from Table 2, one can show the following property: (♣) for a finite KB , the number of $\mathcal{ALCHIQ}$ -clauses unique up to variable renaming is exponential in $|KB|$. The proof is a straightforward counting exercise since the number of variables and the depth of functional terms in $\mathcal{ALCHIQ}$ -clauses are bounded.

Each inference step can be carried out in polynomial time,

so by ($\spadesuit$) and ($\clubsuit$), after at most exponentially many steps, all $\mathcal{ALCHIQ}$ -clauses are derived, and saturation terminates.

Satisfiability of $\Xi(KB)$ can be decided by further saturating $\Gamma \cup \Xi(\mathcal{A})$ by basic superposition. Since Γ contains all non-ground consequences of $\Xi(KB)$, all remaining inferences will produce only ground clauses, and will not involve clauses of type 4 and 6. These inferences can be simulated in a disjunctive datalog program by transforming Γ into a function-free clause set, and by introducing new constants playing the role of ground functional terms, as described next.

We define an operator λ transforming Γ as follows: (i) each functional term $f(a)$ is replaced with a new, globally unique constant a_f ; (ii) each term $f(x)$ is replaced with a new, globally unique variable x_f ; (iii) for each variable in a clause introduced in step (ii), λ appends a literal $\neg S_f(x, x_f)$, where S_f is a new predicate unique for the function symbol f ; (iv) if some variable x occurs in a positive, but not in a negative literal in a clause, then the literal $\neg HU(x)$ is appended to the clause; (v) for each function symbol f and each constant a from $\Xi(KB)$, the facts $S_f(a, a_f)$, $HU(a)$ and $HU(a_f)$ are added. The set of (function-free) clauses obtained by applying λ to $\Gamma \cup \Xi(\mathcal{A})$ is denoted with $FF(KB)$. An example of applying λ to a clause of type 5 is shown below.

$$\neg C(x) \vee D(f(x)) \Rightarrow^\lambda \neg S_f(x, x_f) \vee \neg C(x) \vee D(x_f)$$

Now each remaining ground inference by basic superposition in $\Gamma \cup \Xi(\mathcal{A})$ can be simulated by a sound inference step in $FF(KB)$, and vice versa [Hustadt et al., 2004b], so KB and $FF(KB)$ are equisatisfiable. Since $FF(KB)$ does not contain functional terms and all its clauses are safe, each clause can be rewritten into a positive disjunctive rule; let $DD(KB)$ be the resulting set of rules. The following theorem summarizes the properties of $DD(KB)$:

Theorem 1 ([Hustadt et al., 2004b]). *For KB an $\mathcal{ALCHIQ}$ knowledge base, the following claims hold: (i) KB is unsatisfiable iff $DD(KB)$ is unsatisfiable; (ii) $KB \models \alpha$ iff $DD(KB) \models_c \alpha$, for α of the form $A(a)$ or $S(a, b)$, A a concept name, and S a simple role; (iii) $KB \models C(a)$ iff $DD(KB \cup \{C \sqsubseteq Q\}) \models_c Q(a)$, for C a complex concept, and Q a new concept name; (iv) the number of rules in $DD(KB)$ is at most exponential in $|KB|$, the number of literals in each rule is at most polynomial in $|KB|$, and $DD(KB)$ can be computed in time exponential in $|KB|$.*

3 Data Complexity of Reasoning in $\mathcal{SHIQ}$

For an extensionally reduced $\mathcal{SHIQ}$ knowledge base KB , an upper bound for the data complexity follows from the reduction of KB to $DD(KB)$. Before presenting the details, we first discuss the intuition behind this result.

By Theorem 1, $|DD(KB)|$ is exponential in $|KB|$. However, a closer inspection of the reduction algorithm reveals that the exponential blowup is caused by the rules obtained by saturating $\mathcal{ALCHIQ}$ -clauses of types 1 – 7, which correspond to TBox and RBox, but not to ABox clauses. Hence, the size of the rules of $DD(KB)$ is exponential in the size of TBox and RBox; however, the size of the facts in $DD(KB)$ is linear in the size of the ABox. Therefore, data complexity of checking satisfiability of $DD(KB)$ corresponds to data

complexity of checking satisfiability of a positive disjunctive datalog program, and is thus in NP. Intuitively, this is due to nice property of $\mathcal{SHIQ}$ that TBox and RBox reasoning does not “interfere” with ABox reasoning, i.e. all non-ground consequences of KB can be computed without taking the ABox into account. Notice that this result holds even for binary number coding.

Lemma 1 (Membership). *For KB an extensionally reduced $\mathcal{SHIQ}$ knowledge base, satisfiability of KB can be decided in non-deterministic polynomial time in $|A|$.*

Proof. Let c be the number of constants, f the number of function symbols in the signature of $\Xi(KB)$, and s the number of facts in $\Xi(KB)$ (which is equal to the number of facts in $\mathcal{A}$). By definition of λ , the number of constants in $DD(KB)$ is bounded by $\ell_1 = c + cf$ (cf accounts for constants of the form a_f), and the number of facts in $DD(KB)$ is bounded by $\ell_2 = s + c + 2cf$ (c accounts for facts of the form $HU(a)$, one cf accounts for facts of the form $S_f(a, a_f)$, and the other cf accounts for facts of the form $HU(a_f)$). All function symbols are introduced by skolemizing TBox concepts $\exists R.C$ and $\geq n R.C$. Since $|T|$ and $|R|$ are constant, f is also a constant, so both ℓ_1 and ℓ_2 are linear in $|A|$.

Hence, $|DD(KB)|$ can be exponential in $|KB|$ only because the non-ground rules in $DD(KB)$ are obtained from exponentially many clauses of types 1 – 7. Since these clause types do not contain ABox clauses, the number of clauses obtained after saturation is obviously exponential in $|T| + |R|$ only. Since we assume that the latter is constant, both the number of rules in $DD(KB)$ and their length are bounded by constants, so $|DD(KB)|$ is polynomial in $|A|$, and can be computed from KB in time polynomial in $|A|$.

As KB and $DD(KB)$ are equisatisfiable, the data complexity of checking satisfiability of KB follows from the data complexity of checking satisfiability of $DD(KB)$, which is NP-complete [Dantsin et al., 2001]: assuming $DD(KB)$ contains r rules and at most v variables in a rule, the number of literals in a ground instantiation $\text{ground}(DD(KB))$ is bounded by $r \cdot \ell_1^v + \ell_2$ (in each rule, each variable can be replaced in ℓ_1 possible ways). Assuming r and v are constants, $\wp = |\text{ground}(DD(KB))|$ is polynomial in $|A|$. Satisfiability of $\text{ground}(DD(KB))$ can be checked by non-deterministically generating an interpretation of size at most $\wp$, and then checking whether it is a model. Both tasks can be performed in polynomial time, so the overall algorithm is obviously non-deterministically polynomial in $|A|$. $\square$

The hardness of the satisfiability checking problem follows from [Schaerf, 1994, Lemma 4.2.7]. Actually, the lemma shows co-NP-hardness of instance checking, by a reduction of satisfiability of 2-2-CNF propositional formulae. The reduction produces an extensionally reduced ABox and a single TBox axiom, so it applies in our case as well. Hence, we immediately obtain the following result:

Theorem 2. *Let KB be an extensionally reduced knowledge base in any logic between $\mathcal{ALC}$ and $\mathcal{SHIQ}$. Then (i) deciding KB satisfiability is data complete for NP and (ii) deciding whether $KB \models (\neg)C(a)$ with $|C|$ bounded is data complete for co-NP.*

4 A Horn Fragment of $\mathcal{SHIQ}$

Horn logic is a well-known fragment of first-order logic where formulae are restricted to clauses containing at most one positive literal. The main limitation of Horn logic is its inability to represent disjunctive information; however, its main benefit is the existence of practical refutation procedures. Furthermore, data complexity of query answering in Horn logic without function symbols is P-complete [Dantsin *et al.*, 2001], which makes it appealing for practical usage.

Following this idea, in this section we identify a Horn fragment of $\mathcal{SHIQ}$, where disjunction is traded for P-complete data complexity. Roughly speaking, in Horn- $\mathcal{SHIQ}$, only axioms of the form $\bigcap C_i \sqsubseteq D$ are allowed, where each C_i has the form A or $\exists R.A$, and D has the form A , $\perp$, $\exists R.A$, $\forall R.A$, $\geq n R.A$ or $\leq 1 R$. Whereas such a definition succinctly demonstrates the expressivity of the fragment, in general it is too restricting: e.g., the axiom $A_1 \sqcup A_2 \sqsubseteq \neg B$ is not Horn, but it is equivalent to Horn axioms $A_1 \sqcap B \sqsubseteq \perp$ and $A_2 \sqcap B \sqsubseteq \perp$. Similarly, a non-Horn axiom $A \sqsubseteq \exists R.(\exists R.B)$ can be transformed into Horn axioms $A \sqsubseteq \exists R.Q$ and $Q \sqsubseteq \exists R.B$ by introducing a new name Q for the subconcept $\exists R.B$. To avoid dependency on such obvious syntactic transformations, we give a rather technical definition of Horn- $\mathcal{SHIQ}$.

We first adapt the notions of positions and polarity in first-order formulae to DL. A *position* p is a finite sequence of integers; the empty position is denoted with ϵ . If a position p_1 is a proper prefix of a position p_2 , then p_1 is *above* p_2 , and p_2 is *below* p_1 . For a concept α , the subterm at a position p , written $\alpha|_p$, is defined as follows: $\alpha|_\epsilon = \alpha$; $(\neg D)|_{1p} = D|_p$; $(D_1 \circ D_2)|_{ip} = D_i|_p$ for $\circ \in \{\sqcap, \sqcup\}$ and $i \in \{1, 2\}$; $\alpha|_1 = R$ and $\alpha|_{2p} = D|_p$ for $\alpha = \diamond R.D$ and $\diamond \in \{\exists, \forall\}$; and $\alpha|_1 = n$, $\alpha|_2 = R$ and $\alpha|_{3p} = D|_p$ for $\alpha = \bowtie n R.D$ and $\bowtie \in \{\leq, \geq\}$. A *replacement* of a subterm of α at position p with a term β is defined in the standard way and is denoted as $\alpha[\beta]_p$. For a concept α and a position p such that $\alpha|_p$ is a concept, the *polarity* of $\alpha|_p$ in α , denoted as $\text{pol}(\alpha, p)$, is defined as follows:

$$\begin{aligned} \text{pol}(C, \epsilon) &= 1; \\ \text{pol}(C_1 \circ C_2, ip) &= \text{pol}(C_i, p) \text{ for } \circ \in \{\sqcap, \sqcup\}, i \in \{1, 2\}; \\ \text{pol}(\diamond R.C, 2p) &= \text{pol}(C, p) \text{ for } \diamond \in \{\exists, \forall\}; \\ \text{pol}(\geq n R.C, 3p) &= \text{pol}(C, p); \\ \text{pol}(\neg C, 1p) &= -\text{pol}(C, p); \\ \text{pol}(\leq n R.C, 3p) &= -\text{pol}(C, p). \end{aligned}$$

Intuitively, $\text{pol}(\alpha, p)$ equals 1 if $\alpha|_p$ occurs in α under an even number of explicit and implicit negations, and -1 otherwise.

Definition 1. In Table 3, we define two mutually recursive functions pl^+ and pl^- , where $\text{sgn}(0) = 0$ and $\text{sgn}(n) = 1$ for $n > 0$. For a concept C and a position p of a subconcept in C , let $\text{pl}(C, p) = \text{pl}^+(C|_p)$ if $\text{pol}(C, p) = 1$, and let $\text{pl}(C, p) = \text{pl}^-(C|_p)$ if $\text{pol}(C, p) = -1$.

A concept C is a *Horn concept* if $\text{pl}(C, p) \leq 1$ for each position p of a subconcept in C (including the empty position ϵ). An extensionally reduced $\mathcal{ALCHIQ}$ knowledge base KB is Horn if, for each axiom $C \sqsubseteq D \in KB$, the concept $\neg C \sqcup D$ is Horn. An extensionally reduced $\mathcal{SHIQ}$ knowledge base KB is Horn if $\Omega(KB)$ is Horn.

Table 3: Definitions of pl^+ and pl^-

D	$\text{pl}^+(D)$	$\text{pl}^-(D)$
$\top$	0	0
$\perp$	0	0
A	1	0
$\neg C$	$\text{pl}^-(C)$	$\text{pl}^+(C)$
$\bigcap C_i$	$\max_i \text{sgn}(\text{pl}^+(C_i))$	$\sum_i \text{sgn}(\text{pl}^-(C_i))$
$\bigcup C_i$	$\sum_i \text{sgn}(\text{pl}^+(C_i))$	$\max_i \text{sgn}(\text{pl}^-(C_i))$
$\exists R.C$	1	$\text{sgn}(\text{pl}^-(C))$
$\forall R.C$	$\text{sgn}(\text{pl}^+(C))$	1
$\geq n R.C$	1	$\frac{(n-1)n}{2} + n \text{sgn}(\text{pl}^+(C))$
$\leq n R.C$	$\frac{n(n+1)}{2} + (n+1)\text{sgn}(\text{pl}^-(C))$	1

It is easy to see that, for a concept C without complex subconcepts, $\text{pl}^+(C)$ yields the maximal number of positive literals in clauses obtained by clausifying $\forall x : \pi_y(C, x)$. To clausify a concept C containing a complex subconcept at a position p , one should consider if $C|_p$ occurs in C under positive or negative polarity. E.g., in $\neg(\neg A \sqcap \neg B)$ the concepts A and B occur effectively positive, and $\sqcap$ is effectively $\sqcup$. Hence, $\text{pl}^+(C|_p)$ ($\text{pl}^-(C|_p)$) counts the number of positive literals used to clausify $C|_p$, provided that $C|_p$ occurs in C under positive (negative) polarity. The function $\text{sgn}(\cdot)$ takes into account that $C|_p$ will be replaced in C by structural transformation with only one concept name, even if clausification of $C|_p$ produces more than one positive literal. E.g., to clausify $C = \forall R.(D_1 \sqcup D_2)$, the structural transformation replaces $D_1 \sqcup D_2$ with an new concept name Q , yielding $C' = \forall R.Q$; then clausifying C' produces a clause with only one positive literal. Now a concept C is Horn if the maximal number of positive literals obtained by clausifying subconcepts of C is at most one.

If a concept C has a complex subconcept at position p , special care has to be taken in introducing a new name α for $C|_p$. Consider the Horn concept $C = \forall R.D_1 \sqcup \forall R.\neg D_2$. To apply structural transformation to C , one might replace $\forall R.D_1$ and $\forall R.\neg D_2$ with new concept names Q_1 and Q_2 , yielding concepts $\neg Q_1 \sqcup \forall R.D_1$, $\neg Q_2 \sqcup \forall R.\neg D_2$ and $Q_1 \sqcup Q_2$. The problem with such a straight-forward application of structural transformation is that a Horn concept C was reduced to a non-Horn concept $Q_1 \sqcup Q_2$, so the structural transformation destroyed Horn-ness. To remedy this, we modify the structural transformation to replace each $C|_p$ with a literal concept α such that clausifying α and $C|_p$ requires the same number of positive literals. In the above example, this would mean that $\forall R.D_1$ should be replaced with Q_1 , but $\forall R.\neg D_2$ should be replaced with $\neg Q_2$, yielding concepts $\neg Q_1 \sqcup \forall R.D_1$, $Q_2 \sqcup \forall R.\neg D_2$ and $Q_1 \sqcup \neg Q_2$, which are all Horn.

Although transitivity axioms are translated by π into Horn clauses, recall that the algorithm from Section 2 replaces them with axioms of the form $\forall R.C \sqsubseteq \forall S.(\forall S.C)$. Now $\text{pl}^+(\exists R.\neg C \sqcup \forall S.(\forall S.C)) = 1 + \text{pl}^+(C)$, so if $\text{pl}^+(C) > 0$, $\Omega(KB)$ is not a Horn knowledge base. Hence, the presence of transitivity axioms can make a knowledge base non-Horn.

Definition 2. Let C be a concept and Λ a function mapping C to the set of positions $p \neq \epsilon$ of subconcepts of C such that $C|_p$ is not a literal concept and, for all positions q below p ,

$C|_q$ is a literal concept. Then $\text{Def}(C)$ is defined recursively as follows, where $\alpha = Q$ if $\text{pl}(C, p) > 0$, and $\alpha = \neg Q$ if $\text{pl}(C, p) = 0$, with Q a new atomic concept, and $\neg(\neg Q) = Q$:

- $\text{Def}(C) = \{C\}$ if $\Lambda(C) = \emptyset$, or
- if $\Lambda(C) \neq \emptyset$, then choose some $p \in \Lambda(C)$ and let

$$\text{Def}(C) = \begin{cases} \{\neg\alpha \sqcup C|_p\} \cup \text{Def}(C|_p) & \text{if } \text{pol}(C, p) = 1 \\ \{\neg\alpha \sqcup \neg C|_p\} \cup \text{Def}(C|_p) & \text{if } \text{pol}(C, p) = -1 \end{cases}$$

Let $\text{Cls}(\varphi)$ denote the set of clauses obtained by clausifying a formula φ in the standard way and let

$$\text{Cls}(C) = \bigcup_{D \in \text{Def}(C)} \text{Cls}(\forall x : \pi_y(D, x)).$$

For an $\mathcal{ALCHIQ}$ knowledge base KB , $\Xi(KB)$ is the small set of clauses such that: (i) for each role name $R \in N_{R_a}$, $\text{Cls}(\pi(R)) \subseteq \Xi(KB)$; (ii) for each RBox or ABox axiom α in KB , $\text{Cls}(\pi(\alpha)) \subseteq \Xi(KB)$; (iii) for each TBox axiom $C \sqsubseteq D$ in KB , $\text{Cls}(\neg C \sqcup D) \subseteq \Xi(KB)$.

By [Nonnengart and Weidenbach, 2001], $\forall x : \pi_y(C, x)$ and $\bigwedge_{D \in \text{Def}(C)} \forall x : \pi_y(D, x)$ are equisatisfiable, so $\Xi(KB)$ and $\pi(KB)$ are equisatisfiable as well.

Lemma 2. For a Horn- $\mathcal{SHIQ}$ knowledge base KB , each clause from $\Xi(KB)$ contains at most one positive literal.

Proof. We first show the following property (*): for a Horn concept C , all concepts in $\text{Def}(C)$ are Horn concepts. The proof is by induction on the recursion depth in Def . The induction base for $\Lambda(C) = \emptyset$ is obvious. Consider an application of $\text{Def}(C)$, where C is a Horn concept and p a position of a subconcept of C , such that $C|_p$ is not a literal concept and, for each position q below p , $C|_q$ is a literal concept. Observe that in all cases, we have $\text{pl}^+(\alpha) = \text{pl}(C, p)$ and $\text{pl}^+(\neg\alpha) = 1 - \text{pl}(C, p)$. We now consider two cases, depending on $\text{pol}(C, p)$:

- $\text{pol}(C, p) = 1$. Now we have $\text{pl}^+(\neg\alpha \sqcup C|_p) = \text{pl}^+(\neg\alpha) + \text{pl}^+(C|_p) = \text{pl}^+(\neg\alpha) + \text{pl}(C, p) = 1$. Furthermore, $\text{pl}(C, p) = \text{pl}(C|_p)$, so $C|_p$ is Horn.
- $\text{pol}(C, p) = -1$. Now we have $\text{pl}^+(\neg\alpha \sqcup \neg C|_p) = \text{pl}^+(\neg\alpha) + \text{pl}^+(\neg C|_p) = \text{pl}^+(\neg\alpha) + \text{pl}(C, p) = 1$. Furthermore, $\text{pl}(C, p) = \text{pl}(C|_p)$, so $C|_p$ is Horn.

Hence, each application of the operator Def decomposes a Horn concept C into two simpler Horn concepts, so (*) holds. Furthermore, for each $C|_p$ or $\neg C|_p$ in the definition of Def , each immediate subconcept is a literal.

For $D \in \text{Def}(C)$, by definition of π from Table 1, it is easy to see that $\text{pl}^+(D)$ gives the maximal number of positive literals occurring in a clause from $\text{Cls}(\forall x : \pi_y(D, x))$. Thus, if C is a Horn concept, all clauses from $\text{Cls}(C)$ contain at most one positive literal. Finally, clauses obtained by translating RBox and ABox axioms of $\Omega(KB)$ also contain at most one positive literal. $\square$

As stated by the following lemma, a basic superposition inference, when applied to Horn premises, produces a Horn conclusion. The full proof of the lemma is given in [Hustadt et al., 2004a].

Lemma 3. If all premises of an inference by basic superposition contain at most one positive literal, then inference conclusions also contain at most one positive literal.

Proof. (Sketch) Consider a resolution inference with clauses $A \vee C$ and $\neg B \vee D$, where all literals in C are negative and at most one literal in D is positive. Obviously, the number of positive literals in the conclusion $C\sigma \vee D\sigma$ is equal to the number of positive literals in D . Similarly, consider a paramodulation inference from a clause $s \approx t \vee C$ into a clause $A \vee D$, where all literals in C and D are negative. Obviously, the conclusion $A\sigma[t\sigma]_p \vee C\sigma \vee D\sigma$ has only one positive literal. Similar considerations hold for a paramodulation into a negative literal. $\square$

By Lemma 2 and 3, if KB is a Horn- $\mathcal{SHIQ}$ knowledge base, then $\text{DD}(KB)$ is a non-disjunctive program. This is enough for the following result:

Theorem 3. For KB an extensionally reduced Horn knowledge base in any logic between $\mathcal{ALC}$ and $\mathcal{SHIQ}$, deciding KB (un)satisfiability, and deciding whether $KB \models (\neg)C(a)$ with $|C|$ bounded, is P-complete in $|A|$.

Proof. Membership in P is a consequence of the fact that $\text{DD}(KB)$ is a non-disjunctive program, whose satisfiability can be checked in polynomial time [Dantsin et al., 2001].

For hardness, consider the well-known P-complete problem of deciding whether a path from a node a_1 to a node a_n in a graph G exists [Papadimitriou, 1993]. For a graph G , let KB_G be a knowledge base containing the assertions $\text{edge}(a, b)$ and $\text{edge}(b, a)$ for each edge $\langle a, b \rangle$ in G , the axioms $C(a_1)$ and $\neg C(a_n)$, and the TBox axiom $C \sqsubseteq \forall \text{edge}.C$. Obviously, a_1 is reachable from a_n if and only if KB_G is unsatisfiable, thus implying P-completeness of unsatisfiability checking. The other inference problems can be reduced to unsatisfiability as usual. $\square$

5 Discussion

To better understand the results from the previous two sections, we contrast them with well-known results for (disjunctive) datalog [Dantsin et al., 2001]. Since datalog has been successfully applied in practice, this analysis gives interesting insights into the practical applicability of DLs.

Interestingly, the data complexity of datalog variants and of corresponding $\mathcal{SHIQ}$ fragments coincide. Namely, without disjunctions, a $\mathcal{SHIQ}$ knowledge base and a datalog program always have at most one model, which can be computed in polynomial time. With disjunctions, several models are possible, and this must be dealt with using reasoning-by-cases. Intuitively, one needs to “guess” a model, which increases data complexity to NP.

The key difference between datalog and DLs is revealed by considering the effects that various parameters have on the complexity. For a datalog program P and a ground atom α , checking whether $P \models \alpha$ can be performed in time $\mathcal{O}(|P|^v)$, where v is the maximal number of distinct variables in a rule of P [Vardi, 1995]. Namely, the problem can be solved by grounding P , i.e. by replacing, in each rule of P , all variables with individuals from P in all possible ways. The size of the

grounding is bounded by $|P|^v$, and propositional Horn logic is P-complete, giving the above estimate. Now in general, v is linear in $|P|$, so the size of the grounding is exponential; thus, the combined complexity of datalog coincides with the combined complexity of *SHIQ*. However, in practical applications v is usually small, so it makes sense to assume it is bounded. Under this assumption, datalog actually exhibits polynomial behavior.

By an analogy, one might try to limit the length of concepts in axioms or the number of variables. For the former, structural transformation can be used to polynomially reduce “long” axioms with complex concepts to “short” axioms with just elementary concepts. For the latter, we note that DLs are closely related to the two-variable fragment of first-order logic: e.g., *ALC* concepts correspond to first-order formulae with only two variables regardless of nesting (see, e.g., [Baader et al., 2003, ch. 4]). Therefore, the number of variables in DL axioms is “intrinsically” bounded (assuming a bound on the numbers occurring in number restrictions). Hence, neither restriction actually reduces complexity.

We summarize our discussion as follows: assuming a bound on the axiom length, but not on the number of axioms, satisfiability checking in datalog is (non-deterministically) polynomial, but in DLs it is exponential. The reason for this is that DLs such as *ALC* provide existential and universal quantification and general inclusion axioms, which can be used to succinctly encode models with paths of exponential length. The saturation step eliminates function symbols introduced by existential quantification, but it also incurs an exponential blowup in the program size to account for such paths. Hence, although combined complexity of both datalog and DLs is exponential, the reasons for this are different.

In [Baader et al., 2003, ch. 5] two sources of complexity in DLs have been identified: OR-branching caused by the existence of several possible models, and AND-branching caused by the existence of paths within a model. Our results show that OR-branching is not so “bad” as AND-branching: the former incurs “only” an increase to NP, whereas the latter incurs an increase in complexity to EXPTIME.

6 Conclusion

In many application of DLs, the TBox can be assumed to be rather stable — like a database schema — whereas the ABox is varying and possibly very large — like a database extension. Hence, we study the complexity of reasoning in expressive DLs measured in the size of the ABox. In particular, we show that checking satisfiability of a *SHIQ* knowledge base is NP-complete, and that checking unsatisfiability and instance checking are co-NP-complete in the size of the ABox. Furthermore, we identify Horn-*SHIQ*, a fragment of *SHIQ* which, analogously to Horn logic, does not allow to represent disjunctive knowledge, and for which the basic reasoning problems are P-complete in the size of the ABox.

Our results indicate that reasoning with large ABoxes may be feasible if the TBox is not “too” large and if we stay within Horn-*SHIQ*. To verify these assumptions, we are currently implementing our algorithms and plan to conduct a thorough performance analysis.

Acknowledgments

We thank Enrico Franconi for a stimulating discussion which lead us to the results presented in this paper.

References

- [Baader et al., 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
- [Bachmair et al., 1995] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic Paramodulation. *Information and Computation*, 121(2):172–192, 1995.
- [Calvanese et al., 2004] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, and G. Vetere. DL-Lite: Practical Reasoning for Rich DLs. In *Proc. DL 2004*, volume 104 of *CEUR Workshop Proceedings*, 2004.
- [Dantsin et al., 2001] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Comp. Surveys*, 33(3):374–425, 2001.
- [Hustadt et al., 2004a] U. Hustadt, B. Motik, and U. Sattler. Reasoning in Description Logics with a Concrete Domain in the Framework of Resolution. Technical Report 2-8-1/04, FZI, Karlsruhe, Germany, February 2004. <http://www.fzi.de/wim/publikationen.php?id=1144>.
- [Hustadt et al., 2004b] U. Hustadt, B. Motik, and U. Sattler. Reducing *SHIQ*[−] Description Logic to Disjunctive Datalog Programs. In *Proc. KR2004*, pages 152–162. AAAI Press, 2004.
- [Nieuwenhuis and Rubio, 1995] R. Nieuwenhuis and A. Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *Journal of Logic and Computation*, 19(4):312–351, April 1995.
- [Nonnengart and Weidenbach, 2001] A. Nonnengart and C. Weidenbach. Computing Small Clause Normal Forms. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 6, pages 335–367. Elsevier Science, 2001.
- [Papadimitriou, 1993] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1993.
- [Schaerf, 1994] A. Schaerf. *Query Answering in Concept-Based Knowledge Representation Systems: Algorithms, Complexity, and Semantic Issues*. PhD thesis, Università degli studi di Roma “La Sapienza”, 1994.
- [Schild, 1991] K. Schild. A correspondence theory for terminological logics: preliminary report. In *Proc. IJCAI ‘91*, pages 466–471, Sidney, Australia, 1991.
- [Tobies, 2001] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001.
- [Vardi, 1995] M. Y. Vardi. On the Complexity of Bounded-Variable Queries. In *Proc. PODS 1995*, pages 266–276. ACM Press, 1995.

Equivalence in Abductive Logic

Katsumi Inoue

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430
Japan
ki@nii.ac.jp

Chiaki Sakama

Wakayama University
Sakaedani, Wakayama 640-8510
Japan
sakama@sys.wakayama-u.ac.jp

Abstract

We consider the problem of identifying equivalence of two knowledge bases which are capable of abductive reasoning. Here, a knowledge base is written in either first-order logic or nonmonotonic logic programming. In this work, we will give two definitions of abductive equivalence. The first one, *explainable equivalence*, requires that two abductive programs have the same explainability for any observation. Another one, *explanatory equivalence*, guarantees that any observation has exactly the same explanations in each abductive framework. Explanatory equivalence is a stronger notion than explainable equivalence. In first-order abduction, explainable equivalence can be verified by the notion of extensional equivalence in default theories. In nonmonotonic logic programs, explanatory equivalence can be checked by means of the notion of relative strong equivalence. We also show the complexity results for abductive equivalence.

1 Introduction

Nowadays, abduction is used in many AI applications, including diagnosis, design, updates, and discovery. Abduction is an important paradigm for problem solving, and is incorporated in programming technologies, i.e., *abductive logic programming* (ALP) [12]. Automated abduction is also studied in the literature as an extension of deductive methods or a part of inductive systems, and its computational properties have also been studied [22; 2; 3; 4].

In this work, we are concerned with such computational issues on abductive reasoning. Despite being a problem-solving paradigm, ALP has a lot of issues which have not been fully understood yet. In particular, there are no concrete methods for (a) evaluation of abductive power in ALP, (b) measurement of efficiency in abductive reasoning, (c) semantically correct simplification and optimization, (d) debugging and verification in ALP, and (e) standardization in ALP. Since all these topics are important for any programming paradigm, the lack of them is a serious drawback of ALP. Then, it can be recognized that all the above issues are related to different notions of identification or *equivalence* in ALP. In particular,

the item (c) is related to understanding the semantics of ALP with respect to modularity and *contexts*.

Abduction can be formalized in various logics [13]. Then, we can consider several notions of equivalence in several logics for abduction. In this paper, we will give two definitions of abductive equivalence in two logical frameworks for abduction. Two logics we consider here are *first-order logic* (FOL) and *abductive logic programming* (ALP). The first abductive equivalence, called *explainable equivalence*, requires that two abductive programs have the same explainability for any observation. Another one, *explanatory equivalence*, guarantees that any observation has exactly the same explanations in each abductive framework. Explanatory equivalence is stronger than explainable equivalence.

In this paper, we characterize these two notions of abductive equivalence in terms of other well-known concepts in AI and logic programming. In abduction in first-order logic, we will see that explainable equivalence can be verified by the notion of *equivalence* in *default logic* [18], which is defined for the families of *extensions* of two default theories. On the other hand, abductive equivalence in ALP is more complicated than in the case of FOL due to the nonmonotonicity in logic programs. In fact, equivalence between two abductive logic programs has little been discussed in the literature except that effects of partial deduction in ALP are analyzed in [20]. Here, we will see that explanatory equivalence in ALP can be characterized by the notion of (*relative*) *strong equivalence* [14; 10; 24].

The rest of this paper is organized as follows. Section 2 presents two definitions for abductive equivalence. Section 3 considers first-order logic as the representation language, while Section 4 considers nonmonotonic logic programming for ALP. Section 5 gives concluding remarks.

2 Abductive Equivalence

We start with the question as to when two abductive frameworks are equivalent. As far as the authors know, there is no answer for such a question in the literature of ALP. There are many parameters which should be considered important in defining equivalence notions in abductive frameworks. In the *world*, both background knowledge and observations are surely essential. In an *agent* who performs abduction, on the other hand, her abductive power must depend on her *logic*

(language, syntax, semantics) of background knowledge, observations and hypotheses. Moreover, the quality of abduction is relevant to other parameters such as axioms, inference procedures, logics of explanations, and criteria of best explanations. If we would take all such parameters into account, the task of defining the equivalence notion might become combinatorial and too complex.

In the following, we thus consider a rather simple framework for our problem while we try to hold the essence of equivalence notions as much as possible. First, logic, background knowledge and hypotheses are put as input parameters in each abductive framework. Secondly, a logic of explanations is taken into account in a definition, but its diversity is reflected in different notions of abductive equivalence.

The following definition of abductive frameworks is a standard one [13; 22; 2; 3]. As a notation, $\Sigma \models_L F$ means that a formula F is derived from a set Σ of formulas in a logic L .

Definition 2.1 Let B and H be sets of formulas in some underlying logic L . An *abductive framework* is defined as a triple (L, B, H) , where B is called *background knowledge* and each element of H is called a *candidate hypothesis*.

Definition 2.2 Let (L, B, H) be an abductive framework, and O a formula in L , and E a formula belonging to H . We define that E is an *explanation* of an *observation* O in (L, B, H) if $B \cup E \models_L O$ and $B \cup E$ is consistent in L . We say that O is *explainable* in (L, B, H) if it has an explanation in (L, B, H) .

We now give two definitions for abductive equivalence. We assume that the underlying logic L is common when two abductive theories are compared.

Definition 2.3 Two abductive frameworks (L, B_1, H_1) and (L, B_2, H_2) are *explainably equivalent* if, for any observation O , there is an explanation of O in (L, B_1, H_1) iff there is an explanation of O in (L, B_2, H_2) .

Explainable equivalence requires that two abductive frameworks have the same *explainability* for any observation. Explainable equivalence may reflect a situation that two programs have different knowledge to derive the same goals.

Definition 2.4 Two abductive frameworks (L, B_1, H_1) and (L, B_2, H_2) are *explanatorily equivalent* if, for any observation O , E is an explanation of O in (L, B_1, H_1) iff E is an explanation of O in (L, B_2, H_2) .

Explanatory equivalence assures that two abductive frameworks have the same *explanation power* for any observation. Explanatory equivalence is stronger than explainable equivalence as follows.

Proposition 2.1 If abductive frameworks (L, B_1, H_1) and (L, B_2, H_2) are *explanatorily equivalent*, then they are *explainably equivalent*.

For explanatory equivalence, we can assume that the hypotheses H are common in two abductive frameworks in Definition 2.4, as the following property holds.

Proposition 2.2 If $A_1 = (L, B_1, H_1)$ and $A_2 = (L, B_2, H_2)$ are *explanatorily equivalent*, then $H'_1 = H'_2$, where $H'_i = \{h \in H_i \mid B_i \cup \{h\} \text{ is consistent in } L\}$ for $i = 1, 2$.

Proof. Assume that $H'_1 \setminus H'_2 \neq \emptyset$. Then, for a formula $\varphi \in H'_1 \setminus H'_2$, $\{\varphi\}$ is an explanation of φ in A_1 because $B_1 \cup \{\varphi\}$ is consistent in L . However, $\{\varphi\}$ is not an explanation of φ in A_2 . Hence, A_1 and A_2 are not explanatorily equivalent. $\square$

Note in Proposition 2.2 that any hypothesis h in $H_i \setminus H'_i$ cannot be added without violating the consistency of $B_i \cup \{h\}$ in L . Thus, H'_i is the set of hypotheses that can be actually used in explanations of some formulas.

Example 2.1 Suppose the abductive frameworks $A_1 = (\text{FOL}, \{a \supset p\}, \{a, b\})$ and $A_2 = (\text{FOL}, \{b \supset p\}, \{a, b\})$. Then, A_1 and A_2 are explainably equivalent, but are not explanatorily equivalent. On the other hand, $A_3 = (\text{FOL}, \{a \supset p\}, \{b\})$ and $A_4 = (\text{FOL}, \{b \supset p\}, \{b\})$ are neither explainably equivalent nor explanatorily equivalent.

3 Abduction in First-order Logic

Abduction is used in many AI applications, and classical first-order logic is most often used as the underlying logic for abduction [17; 13; 2; 22]. When the underlying logic L is FOL, the relation $\models_L$ becomes the usual entailment relation $\models$. In first-order abduction, explanations are usually defined as a set of ground instances of elements of H is an *explanation* of O in (FOL, B, H) if $\Sigma \cup E \models O$ and $\Sigma \cup E$ is consistent.

In the following, $Th(\Sigma)$ denotes the set of logical consequences of a set Σ of first-order formulas. That is, $Th(\Sigma) = \{F \mid \Sigma \models F\}$. The next definition is originally given for *default logic* by Reiter [18].

Definition 3.1 [19; 17] Let B and H be sets of first-order formulas. An *extension* of B with respect to H is $Th(B \cup S)$ where S is a maximal subset of ground instances of elements from H such that $B \cup S$ is consistent.

Using the notion of extensions, explainable equivalence can be characterized in first-order abduction.

Theorem 3.1 Two abductive frameworks (FOL, B_1, H_1) and (FOL, B_2, H_2) are *explainably equivalent* iff the extensions of B_1 with respect to H_1 coincide with the extensions of B_2 with respect to H_2 .

Proof. First, we claim that the union of the extensions of B with respect to H are exactly the set of formulas explainable in (FOL, B, H) . To see this, we can use a well-known theorem [17; 22] that a formula O can be explained in (FOL, B, H) iff there is a consistent extension X of B with respect to H such that X contains O . Thus, the set of all explainable formulas are precisely those formulas contained in at least one extension of B with respect to H .

Now, let $A_1 = (\text{FOL}, B_1, H_1)$ and $A_2 = (\text{FOL}, B_2, H_2)$ be two abductive frameworks. Suppose that the extensions of B_1 with respect to H_1 coincide with those of B_2 with respect to H_2 . By the above claim, the set of formulas explainable in A_1 is equal to the set of formulas explainable in A_2 . This means that A_1 and A_2 are explainably equivalent.

Conversely, assume that there is an extension X_2 of B_2 with respect to H_2 which is not an extension of B_1 with respect to H_1 . Let F_{X_2} be a first-order formula which is logically equivalent to X_2 . Such a formula actually exists be-

cause $X_2 = Th(B_2 \cup S)$ holds for some maximally consistent subset S of H_2 , and hence X_2 is logically equivalent to $\bigwedge_{f \in B_2} f \wedge \bigwedge_{g \in S} g$. Since X_2 is consistent, F_{X_2} is consistent too. Then, F_{X_2} is explainable in A_2 because S is an explanation of F_{X_2} .

Now, if F_{X_2} is not explainable in A_1 , then obviously A_1 and A_2 are not explainably equivalent. Hence, there is an explanation of F_{X_2} in A_1 . Then, there is an extension X_1 of B_1 with respect to H_1 which contains F_{X_2} . Since X_2 is not an extension of B_1 with respect to H_1 , $X_1 \neq X_2$ holds. Then, $X_2 \subset X_1$. Let F_{X_1} be a formula which is logically equivalent to X_1 . By the same argument as above, F_{X_1} is explainable in A_1 . However, this F_{X_1} cannot be explained in A_2 . This is because, if F_{X_1} were explained in A_2 , there must be an extension X'_2 of B_2 with respect to H_2 such that $X_2 \subset X'_2$, which is impossible because any extension is *orthogonal* to another extension in a default theory [18]. In any case, A_1 and A_2 are not explainably equivalent. $\square$

In [15], Reiter's default theories $\Delta_1 = (D_1, B_1)$ and $\Delta_2 = (D_2, B_2)$ are said to be *equivalent* if the extensions of Δ_1 are the same as the extensions of Δ_2 . When an abductive framework (FOL, B, H) is given, we can associate a default theory $\Delta = (D, B)$ where D is the set of *prerequisite-free normal defaults* $\{\frac{!d}{d} \mid d \in H\}$ such that there is a one-to-one correspondence between the extensions of Δ and the extensions of B with respect to H [17].

Corollary 3.2 *Two abductive frameworks (FOL, B_1, H_1) and (FOL, B_2, H_2) are explainably equivalent iff the default theories (D_1, B_1) and (D_2, B_2) are equivalent where $D_i = \{\frac{!d}{d} \mid d \in H_i\}$ for $i = 1, 2$.*

Example 3.1 Suppose two abductive frameworks, $A_1 = (\text{FOL}, B_1, H_1)$ and $A_2 = (\text{FOL}, B_2, H_2)$, where

$$\begin{aligned} B_1 &= \{a \supset p, b \supset \neg p\}, \\ H_1 &= \{a, b, a \equiv c, b \equiv d, p \equiv q\}, \\ B_2 &= \{c \supset q, d \supset \neg q\}, \text{ and} \\ H_2 &= \{c, d, a \equiv c, b \equiv d, p \equiv q\}. \end{aligned}$$

Then, A_1 and A_2 are explainably equivalent. In fact, the two extensions of B_1 with respect to H_1 are $Th(B_1 \cup (H_1 \setminus \{b\})) = Th(\{a, \neg b, c, \neg d, p, q\})$ and $Th(B_1 \cup (H_1 \setminus \{a\})) = Th(\{\neg a, b, \neg c, d, \neg p, \neg q\})$, which are respectively equivalent to the two extensions of B_2 with respect to H_2 , $Th(B_2 \cup (H_2 \setminus \{d\}))$ and $Th(B_2 \cup (H_2 \setminus \{c\}))$.

It is interesting to see that we can transform any abductive framework to an explainably equivalent abductive framework whose background theory is empty. The next property is also derived by the representation theory for default logic [15].

Corollary 3.3 *For any abductive framework (FOL, B, H) , there is an abductive framework $(\text{FOL}, \emptyset, H')$ which is explainably equivalent to (FOL, B, H) .*

Proof. Put $H' = \{h \wedge \varphi \mid h \in H\} \cup \{\varphi\}$, where $\varphi = \bigwedge_{f \in B} f$. Then, it holds for any O that, $B \cup E \models O$ iff $E' \models O$ where $E \subseteq H$ and $E' = \{h \wedge \varphi \mid h \in E\} \cup \{\varphi\} \subseteq H'$. $\square$

An abductive framework (L, B, H) is called *compatible* if $B \cup H$ is consistent. Explainable equivalence can be easily verified for compatible frameworks.

Corollary 3.4 *Two compatible abductive frameworks (FOL, B_1, H_1) and (FOL, B_2, H_2) are explainably equivalent iff $B_1 \cup H_1 \equiv B_2 \cup H_2$.*

An abductive framework $(\text{FOL}, B, \mathcal{L})$ is called *assumption-free* where $\mathcal{L}$ is the set of all literals in the underlying language. It is known that the complexity of finding explanations in assumption-free abductive frameworks is not harder than that in assumption-based frameworks [22; 4]. Explainable equivalence in the assumption-free case can also be simply characterized as follows.

Corollary 3.5 *Two abductive frameworks $(\text{FOL}, B_1, \mathcal{L})$ and $(\text{FOL}, B_2, \mathcal{L})$ are explainably equivalent iff $B_1 \equiv B_2$.*

Proof. For an assumption-free abductive framework $(\text{FOL}, B, \mathcal{L})$, each extension of B with respect to $\mathcal{L}$ is logically equivalent to a model of B . Hence, explainable equivalence implies that the models of B_1 coincide with the models of B_2 , and vice versa. $\square$

For explanatory equivalence in first-order abduction, logical equivalence of background theories is necessary and sufficient.

Theorem 3.6 *Two abductive frameworks (FOL, B_1, H) and (FOL, B_2, H) are explanatorily equivalent iff $B_1 \equiv B_2$.*

Proof. If $B_1 \equiv B_2$, then for any E and any O , it holds that, $B_1 \cup E \models O$ iff $B_2 \cup E \models O$, and that, $B_1 \cup E$ is consistent iff $B_2 \cup E$ is consistent. Hence, (FOL, B_1, H) and (FOL, B_2, H) are explanatorily equivalent.

Conversely, suppose that (FOL, B_1, H) and (FOL, B_2, H) are explanatorily equivalent. Then, for any formula O and any E from H , it holds that $B_1 \cup E \models O$ iff $B_2 \cup E \models O$. Then, for any E , we have $Th(B_1 \cup E) = Th(B_2 \cup E)$. That is, $B_1 \cup E \equiv B_2 \cup E$ holds for any E . This implies $B_1 \equiv B_2$ when $E = \emptyset$. $\square$

The complexity of abductive equivalence in the propositional case can be given as follows.

Lemma 3.7 [1] *Let $\Delta = (D, W)$ be a prerequisite-free normal default theory. Then, a formula φ is true in all extensions of Δ iff $\Delta' = (D, W \cup \{\varphi\})$ and Δ are equivalent.*

Theorem 3.8 *Deciding explainable equivalence in propositional abduction is Π_2^P -complete.*

Proof. By Corollary 3.2 and Lemma 3.7, cautious reasoning in default logic can be transformed to explainable equivalence via equivalence of default theories. This transformation is obviously feasible in polynomial time. Because cautious reasoning from prerequisite-free normal default theories is Π_2^P -complete [6], the decision problem of explainable equivalence is Π_2^P -hard.

We now prove membership in Π_2^P . Two abductive frameworks $A_1 = (\text{FOL}, B_1, H_1)$ and $A_2 = (\text{FOL}, B_2, H_2)$ are not explainably equivalent iff there is an extension of B_1 with respect to H_1 which is not an extension of B_2 with respect to H_2 . Given a subset $S \subseteq H_1$ as a guess, deciding if $X = Th(B_1 \cup S)$ is an extension of B_1 with respect to H_1 can be checked by computing the “reduct” S' of H_1 by X and then checking if $B_1 \cup S \equiv B_1 \cup S'$. Here, computing the reduct requires satisfiability tests, and this computation as

well as testing logical equivalence can be done in polynomial time with an NP-oracle. Once we know that X is an extension of B_1 with respect to H_1 , we need to check if X is not an extension of B_2 with respect to H_2 , which can also be done in the same way as the former test. Thus, we can construct a polynomial-time nondeterministic Turing machine with an NP-oracle to decide if A_1 and A_2 are not explainably equivalent. Hence, the original problem is the complement of this, and belongs to $\text{coNP}^{\text{NP}} = \Pi_2^P$. $\square$

Theorem 3.9 *The following decision problems in propositional abduction are coNP-complete.*

- (1) *Explainable equivalence of two compatible abductive frameworks.*
- (2) *Explainable equivalence of two assumption-free abductive frameworks.*
- (3) *Explanatory equivalence of two abductive frameworks.*

Proof. By Corollaries 3.4 and 3.5 and Theorem 3.6, these problems can be solved by checking logical equivalence of two propositional theories, which is coNP-complete. $\square$

4 Abductive Logic Programming

Abductive logic programming (ALP) is another popular formalization of abduction in AI [12; 3]. Background knowledge in ALP is called a *logic program*, and the candidate hypotheses are given as literals called *abducibles*. The most significant difference between abduction in FOL and ALP is that ALP allows the nonmonotonic *negation-as-failure* operator *not* in background knowledge. In abduction, addition of hypotheses may invalidate explanations of some observations if the background theory is nonmonotonic.

Recall that a (*logic*) *program* is a set of rules of the form

$$L_1; \dots; L_k; \text{not } L_{k+1}; \dots; \text{not } L_l \\ \leftarrow L_{l+1}, \dots, L_m, \text{not } L_{m+1}, \dots, \text{not } L_n$$

where each L_i is a literal ($n \geq m \geq l \geq k \geq 0$), and *not* is *negation as failure* (NAF). The symbol $;$ represents a disjunction. The left-hand side of the rule is the *head*, and the right-hand side is the *body*. A rule with variables stands for the set of its ground instances. In this paper, the semantics of a logic program is given by its *answer sets* [5; 9], while another semantics can be considered as well in ALP [12; 3]. Problem solving by representing knowledge as a logic program and then computing its answer sets is called *answer set programming* (ASP).

Definition 4.1 An *abductive (logic) program* is defined as a pair $\langle P, \mathcal{A} \rangle$, where P is a logic program and $\mathcal{A}$ is a set of literals called *abducibles*. Instead of using the notation $(\text{ALP}, P, \mathcal{A})$, we also use $\langle P, \mathcal{A} \rangle$ to represent an abductive framework whose underlying logic is ALP.

Definition 4.2 Let $\langle P, \mathcal{A} \rangle$ be an abductive program, and G a conjunction of ground literals called *observations*. A set $E \subseteq \mathcal{A}$ is a (*credulous*) *explanation* of G in $\langle P, \mathcal{A} \rangle$ ¹ if every ground literal in G is true in a consistent answer set of $P \cup E$.

¹Another, *skeptical* notion for explanations is defined as $E \subseteq \mathcal{A}$ such that G is true in all consistent answer sets of $P \cup E$.

Note that both abducibles and observations are restricted to ground literals in ALP. However, it is known for this framework that rules can be allowed in abducibles and that observations can contain NAF formulas as well as literals [8]. Definition 4.2 can also be represented in a different way as follows [8]. A *belief set (with respect to E)* of an abductive program $\langle P, \mathcal{A} \rangle$ is a consistent answer set of a logic program $P \cup E$ where $E \subseteq \mathcal{A}$. Then, $E \subseteq \mathcal{A}$ is an explanation of G if G is true in a belief set of $\langle P, \mathcal{A} \rangle$ with respect to E .

Definition 4.3 Let $A_1 = \langle P_1, \mathcal{A}_1 \rangle$ and $A_2 = \langle P_2, \mathcal{A}_2 \rangle$ be abductive programs. A_1 and A_2 are *explainably equivalent* if, for any ground literal G , G is explainable in A_1 iff G is explainable in A_2 . A_1 and A_2 are *explanatorily equivalent* if, for any conjunction of ground literals G , E is an explanation of G in A_1 iff E is an explanation of G in A_2 .

Explainable equivalence in ALP guarantees the same explainability for any ground literal as a *single observation*, but it does not matter how each observation is explained. Hence, we do not have to care about whether multiple observations can be *jointly* explained by a common explanation. On the other hand, explanatory equivalence in ALP guarantees that, any *conjunction* (or *set*) of observations² has exactly the same credulous explanations. Hence, explanatory equivalence implies that any set of abducibles $E \subseteq \mathcal{A}$ should explain the same set of observations in each abductive program.

We now show that explainable equivalence in ALP can be checked by comparing the belief sets of two abductive programs. Because there exist several methods to compute belief sets using ASP [21; 8; 9], checking explainable equivalence is also possible using such methods. In the following, we denote the set of all belief sets of $\langle P, \mathcal{A} \rangle$ as $BS(P, \mathcal{A})$.

Theorem 4.1 *Abductive programs $\langle P_1, \mathcal{A}_1 \rangle$ and $\langle P_2, \mathcal{A}_2 \rangle$ are explainably equivalent iff*

$$\bigcup_{S \in BS(P_1, \mathcal{A}_1)} S = \bigcup_{S \in BS(P_2, \mathcal{A}_2)} S.$$

Proof. Recall that $E \subseteq \mathcal{A}$ is an explanation of a ground literal G iff G is true in a belief set of $\langle P, \mathcal{A} \rangle$ with respect to E . Then, the set of all explainable literals are precisely those literals contained in some belief sets of $\langle P, \mathcal{A} \rangle$ with respect to some E . Hence, the union of the belief sets of $\langle P, \mathcal{A} \rangle$ are exactly the set of literals explainable in $\langle P, \mathcal{A} \rangle$. Therefore, two abductive programs are explainably equivalent iff the unions of the belief sets of two abductive programs coincide. $\square$

In some case of compatible problems, explanatory equivalence can be easily verified. Here, a logic program is *definite* if every its rule is NAF-free and has exactly one atom in the head and only atoms in the body. A definite program has a unique answer set that is equivalent to its *least model*. An abductive program $\langle P, \mathcal{A} \rangle$ is called *definite* if P is a definite logic program and $\mathcal{A}$ is a set of atoms.

Corollary 4.2 *Two definite abductive programs $\langle P_1, \mathcal{A}_1 \rangle$ and $\langle P_2, \mathcal{A}_2 \rangle$ are explainably equivalent if the least model of $P_1 \cup \mathcal{A}_1$ coincides with that of $P_2 \cup \mathcal{A}_2$.*

²We assume that the set of observations includes the special atom $\top$, which represents the empty conjunction of observations. Note that $\top$ is always true in any consistent set of ground literals.

Example 4.1 Given the common set of abducibles $\mathcal{A} = \{a, b\}$ and three logic programs:

$$\begin{aligned} P_1 &= \{p \leftarrow a, \quad q \leftarrow b\}, \\ P_2 &= \{p \leftarrow b, \quad q \leftarrow a\}, \\ P_3 &= \{p \leftarrow, \quad q \leftarrow a, \quad \leftarrow a, b\}, \end{aligned}$$

the three abductive programs $\langle P_i, \mathcal{A} \rangle$ (for $i = 1, 2, 3$) are all explainably equivalent, but none of them are explanatorily equivalent. In particular, the least model of $P_1 \cup \mathcal{A}$ is $\{p, q, a, b\}$, which is identical to that of $P_2 \cup \mathcal{A}$. P_3 is not definite because of the third rule, but $\langle P_3, \mathcal{A} \rangle$ has three belief sets: $\{p\}$, $\{p, q, a\}$, $\{p, b\}$, the union of which is equal to that of $\langle P_i, \mathcal{A} \rangle$ for $i = 1, 2$.

Explanatory equivalence in ALP, on the other hand, requires a more semantical notion of logic programming.³ For this purpose, we need to utilize the concept of equivalence in logic programming and ASP.

There are several notions for equivalence in logic programming, and *weak equivalence* and *strong equivalence* are most well known. We say that two programs are *weakly equivalent* if they simply agree with their answer sets. The notion of weak equivalence is similar to that of *logical equivalence* in FOL and other classical logics. Given two abductive programs $\langle P_1, \mathcal{A} \rangle$ and $\langle P_2, \mathcal{A} \rangle$, however, weak equivalence of P_1 and P_2 is not a sufficient condition for explanatory equivalence of them, and is not even a sufficient condition for explainable equivalence. Weak equivalence is meaningful only when the abducibles are empty.

Proposition 4.3 *Abductive programs $\langle P_1, \emptyset \rangle$ and $\langle P_2, \emptyset \rangle$ are explanatorily equivalent iff P_1 and P_2 are weakly equivalent.*

On the other hand, *strong equivalence* [14] is a more context-sensitive notion for equivalence of logic programs. Two logic programs P_1 and P_2 are said to be *strongly equivalent* if for any additional logic program R , $P_1 \cup R$ and $P_2 \cup R$ have the same answer sets. When we allow NAF in logic programs, weak equivalence is too fragile as a criterion. For example, $\{p \leftarrow \text{not } a\}$ and $\{p \leftarrow\}$ are weakly equivalent with the same unique answer set $\{p\}$, but adding a to both results in the withdrawal of p in the former. Often, we can restrict the language for additional programs R to some subset $\mathcal{R}$ of the whole language of programs. Then, two programs P_1 and P_2 are said to be *strongly equivalent with respect to $\mathcal{R}$* if $P_1 \cup R$ and $P_2 \cup R$ have the same answer sets for any program $R \subseteq \mathcal{R}$ [10]. The equivalence notion with such restriction is called *relative strong equivalence* [10; 24]. Using this notion, explanatory equivalence can be characterized as follows.

Theorem 4.4 *Two abductive programs $\langle P_1, \mathcal{A} \rangle$ and $\langle P_2, \mathcal{A} \rangle$ are explanatorily equivalent iff P_1 and P_2 are strongly equivalent with respect to $\mathcal{A}$.*

Proof. Suppose that $\langle P_1, \mathcal{A} \rangle$ and $\langle P_2, \mathcal{A} \rangle$ are explanatorily equivalent. Then, for any conjunction G of literals and any

³Explanatory equivalence of $\langle P_1, \mathcal{A} \rangle$ and $\langle P_2, \mathcal{A} \rangle$ implies $BS(P_1, \mathcal{A}) = BS(P_2, \mathcal{A})$, but the converse does not hold. For example, when $P_1 = \{a \leftarrow, p \leftarrow a\}$, $P_2 = \{a \leftarrow \text{not } a, p \leftarrow a\}$ and $\mathcal{A} = \{a\}$, $BS(P_1, \mathcal{A}) = BS(P_2, \mathcal{A}) = \{\{a, p\}\}$. However, $\emptyset$ is an explanation of p, a and $\top$ in $\langle P_1, \mathcal{A} \rangle$, but is not in $\langle P_2, \mathcal{A} \rangle$.

$E \subseteq \mathcal{A}$, it holds that, E is an explanation of G in $\langle P_1, \mathcal{A} \rangle$ iff E is an explanation of G in $\langle P_2, \mathcal{A} \rangle$. The latter equivalence then implies that, for any G and any E , we have that, G is true in a belief set of $\langle P_1, \mathcal{A} \rangle$ with respect to E iff G is true in a belief set of $\langle P_2, \mathcal{A} \rangle$ with respect to E . Then, for any G and any E , G is true in an answer set of $P_1 \cup E$ iff G is true in an answer set of $P_2 \cup E$. That is, for any E and any set S of literals, S is an answer set of $P_1 \cup E$ iff S is an answer set of $P_2 \cup E$. Hence, P_1 and P_2 are strongly equivalent with respect to $\mathcal{A}$. The converse direction can also be proved by tracing the above proof backward. $\square$

Example 4.2 Given the common set of abducibles $\mathcal{A} = \{a, b\}$, consider three programs

$$\begin{aligned} P_1 &= \{p \leftarrow a, \quad a \leftarrow b\}, \\ P_2 &= \{p \leftarrow a, \quad p \leftarrow b, \quad a \leftarrow b\}, \\ P_3 &= \{p \leftarrow b, \quad a \leftarrow b\}. \end{aligned}$$

Then, the three abductive programs $\langle P_i, \mathcal{A} \rangle$ (for $i = 1, 2, 3$) are explainably equivalent. Although $\langle P_1, \mathcal{A} \rangle$ is explanatorily equivalent to $\langle P_2, \mathcal{A} \rangle$, it is not to $\langle P_3, \mathcal{A} \rangle$ [20]. In fact, P_1 and P_2 are strongly equivalent with respect to $\mathcal{A}$, while P_1 and P_3 are not because the addition of a derives p in P_1 but this is not the case in P_3 . This example shows that unfold/fold transformation [23] does not preserve explanatory equivalence in ALP [20] even when P_1 and P_2 are definite.

The complexity results of abductive equivalence in ALP are given as follows.

Theorem 4.5 *Deciding explainable equivalence in propositional ALP is in Δ_3^P .*

Proof. Explanation-finding, i.e., deciding if each literal has an explanation in an ALP, is a Σ_2^P -complete problem [3]. To decide explainable equivalence, we need to check if explainability agrees in two abductive frameworks for each literal. Thus we can construct a polynomial-time deterministic Turing machine with an oracle for the explanation-finding problem in order to decide explainable equivalence. Hence, the problem is in $P^{\Sigma_2^P} = \Delta_3^P$. $\square$

Theorem 4.6 *Deciding explainable equivalence in propositional ALP is Π_2^P -hard.*

Proof. The problem contains the case that the abducibles are empty. In this case, explainable equivalence and explanatory equivalence coincide. Then, by Proposition 4.3, the problem reduces to deciding weak equivalence of two logic programs, which is known to be Π_2^P -complete [16]. $\square$

Corollary 4.7 *Explainable equivalence of two definite abductive programs can be decided in polynomial time.*

Theorem 4.8 *Deciding explanatory equivalence in propositional ALP is Π_2^P -complete.*

Proof. From a set $\mathcal{A}$ of literals, we construct a logic program

$$\mu(\mathcal{A}) = \{\delta_l; \text{not } \delta_l \leftarrow, \quad l \leftarrow \delta_l \mid l \in \mathcal{A}\},$$

where δ_l is a new atom which is uniquely associated with l . Then, it can be shown that, P_1 and P_2 are strongly equivalent with respect to $\mathcal{A}$ iff $P'_1 = P_1 \cup \mu(\mathcal{A})$ and $P'_2 = P_2 \cup \mu(\mathcal{A})$ are weakly equivalent. By Theorem 4.4, explanatory equivalence of $\langle P_1, \mathcal{A} \rangle$ and $\langle P_2, \mathcal{A} \rangle$ reduces to weak equivalence of P'_1 and P'_2 , which is Π_2^P -complete [16]. $\square$

5 Conclusion

We have introduced the notion of abductive equivalence in this paper. We have considered two definitions of abductive equivalence in two logics. Two important differences between FOL and ALP as the underlying logics are that (1) explainability is considered for all formulas in FOL while only literals are considered as observations in ALP, and that (2) nonmonotonicity by NAF appears in ALP while this is not the case in FOL. Intuitively, the restriction of observations to literals in ALP gives more chances for two abductive programs to be equivalent, but the existence of nonmonotonicity in ALP makes comparison of abductive programs more complicated. In each case, we can observe that explanatory equivalence is not computationally harder than explainable equivalence.

In [11], the notion of abductive equivalence in this paper is further applied to *extended abduction* [7], where hypotheses can not only be added to a program but also be removed from the program to explain an observation. In extended abduction, explanatory equivalence can be characterized by the notion of *update equivalence* [10].

We have observed that logical equivalence of background theories in FOL or weak equivalence of logic programs does not simply imply abductive equivalence except for some very simple cases. That is why we need to characterize abductive equivalence in terms of other known concepts in classical or nonmonotonic logics. Having such characterizations in this paper, the next target is to develop transformation techniques which preserve abductive equivalence. In another future work, we can consider other underlying logics for background theories, hypotheses and observations as well as the criteria of best explanations for abductive equivalence.

Acknowledgment

We thank Kazuhisa Makino for his valuable comments.

References

- [1] Jürgen Dix. Default theories of Poole-type and a method for constructing cumulative version of default logic. In: *Proceedings of ECAI-92*, pages 289–293, 1992.
- [2] Thomas Eiter and George Gottlob. The complexity of logic-based abduction. *J. ACM*, 42(1):3–42, 1995.
- [3] Thomas Eiter, George Gottlob and Nicola Leone. Abduction from logic programs: semantics and complexity. *Theoretical Computer Science*, 189:129–177, 1998.
- [4] Thomas Eiter and Kazuhisa Makino. Abduction and the dualization problem. In: *Proceedings of the 6th International Conference on Discovery Science*, LNAI 2843, pages 1–20, Springer, 2003.
- [5] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
- [6] George Gottlob. Complexity results for nonmonotonic logics. *J. Logic and Computation*, 2:397–425, 1992.
- [7] Katsumi Inoue and Chiaki Sakama. Abductive framework for nonmonotonic theory change. In: *Proceedings of IJCAI-95*, pages 204–210, Morgan Kaufmann, 1995.
- [8] Katsumi Inoue and Chiaki Sakama. A fixpoint characterization of abductive logic programs. *J. Logic Programming*, 27:107–136, 1996.
- [9] Katsumi Inoue and Chiaki Sakama. Negation as failure in the head. *J. Logic Programming*, 35(1):39–78, 1998.
- [10] Katsumi Inoue and Chiaki Sakama. Equivalence of logic programs under updates. In: *Proceedings of the 9th European Conference on Logics in Artificial Intelligence*, LNAI 3229, pages 174–186, Springer, 2004.
- [11] Katsumi Inoue and Chiaki Sakama. On abductive equivalence. In: Lorenzo Magnani, editor, *Model-based Reasoning in Science and Engineering*, submitted, 2005.
- [12] A. C. Kakas, R. A. Kowalski and F. Toni. The role of abduction in logic programming. In: D. M. Gabbay, C. J. Hogger and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, Volume 5, pages 235–324, Oxford University Press, 1998.
- [13] Hector J. Levesque. A knowledge-level account of abduction (preliminary version). In: *Proceedings of IJCAI-89*, pages 1061–1067, Morgan Kaufmann, 1989.
- [14] Vladimir Lifschitz, David Pearce and Agustín Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.
- [15] V. Wiktor Marek, Jan Treur and Mirosław Truszczyński. Representation theory for default logic. *Annals of Mathematics and Artificial Intelligence*, 21:343–358, 1997.
- [16] Emilia Oikarinen and Tomi Janhunen. Verifying the equivalence of logic programs in the disjunctive case. In: *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning*, LNAI 2923, pages 180–193, Springer, 2004.
- [17] David Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.
- [18] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
- [19] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:571–95, 1987.
- [20] Chiaki Sakama and Katsumi Inoue. The effect of partial deduction in abductive reasoning. In: *Proceedings of the 12th International Conference on Logic Programming*, pages 383–397, MIT Press, 1995.
- [21] Ken Satoh and Noboru Iwayama. Computing abduction by using the TMS. In: *Proceedings of the 8th International Conference on Logic Programming*, pages 505–518, MIT Press, 1991.
- [22] Bart Selman and Hector J. Levesque. Support set selection for abductive and default reasoning. *Artificial Intelligence*, 82(1–2):259–272, 1996.
- [23] Hisao Tamaki and Taisuke Sato. Unfold/fold transformation of logic programs. In: *Proceedings of the 2nd International Conference on Logic Programming*, pages 127–138, Uppsala University, Uppsala, Sweden, 1984.
- [24] Stefan Woltran. Characterizations for relativized notions of equivalence in answer set programming. In: *Proceedings of the 9th European Conference on Logics in Artificial Intelligence*, LNAI 3229, pages 161–173, Springer, 2004.

Iterated Belief Revision, Revised

Yi Jin and Michael Thielscher

Department of Computer Science
Dresden University of Technology
{yijin,mit}@inf.tu-dresden.de

Abstract

The AGM postulates for belief revision, augmented by the DP postulates for iterated belief revision, provide generally accepted criteria for the design of operators by which intelligent agents adapt their beliefs incrementally to new information. These postulates alone, however, are too permissive: They support operators by which all newly acquired information is canceled as soon as an agent learns a fact that contradicts some of its current beliefs. In this paper, we present a formal analysis of the deficiency of the DP postulates, and we show how to solve the problem by an additional postulate of *independence*. We give a representation theorem for this postulate and prove that it is compatible with AGM and DP.

1 Introduction

Belief revision is the process of changing the beliefs of an agent to accommodate new, more precise, or more reliable evidence that is possibly inconsistent with the existing beliefs. In situations where the new evidence is consistent with the existing beliefs, the two can just be merged; we call this *mild revision*. More interesting and complicated are situations where the evidence conflicts with the prior beliefs, in which case the agent needs to remove some of its currently held beliefs in order to accommodate the new evidence. This kind of revision is referred to as *severe revision*.

In the literature, the classical approach of belief revision is the AGM framework [Alchourrón *et al.*, 1985; Gärdenfors and Makinson, 1988; Gärdenfors, 1988]. Given an underlying logic language $\mathcal{L}$, the beliefs of an agent are represented by a set of sentences in $\mathcal{L}$ (known as *belief set*) which is closed under logical consequence. New evidence is also a sentence in $\mathcal{L}$, and a belief revision operator maps the current belief set and the new evidence to a revised belief set. To provide general design criteria for belief revision operators, Alchourrón, Gärdenfors, and Makinson (AGM) have developed a set of postulates [Alchourrón *et al.*, 1985]. The guiding principle is that of *economy of information* or *minimal change* of belief sets, which means not to give up currently held beliefs and not to generate new beliefs unless necessary.

For the incremental adaptation of beliefs, these postulates proved to be too weak [Darwiche and Pearl, 1994; 1997]. This has led to the development of additional postulates for iterated belief revision by Darwiche and Pearl (DP), among others.

Still, however, the two sets of postulates are too permissive in that they support belief revision operators which assume arbitrary dependencies among the pieces of information which an agent acquires along its way. A universal dependency among all new evidences has a drastic effect when the agent makes an observation which contradicts its currently held beliefs: The agent has to cancel everything it has learned up to this point [Nayak *et al.*, 1996; 2003]. In this paper, we first give a formal analysis of this problem of implicit dependence, and then we present, as a solution, an *Independence* postulate for iterated belief revision. We give a representation theorem for our postulate and prove its consistency by defining a concrete belief revision operator. We also contrast the postulate of independence to the so-called Recalcitrance postulate of [Nayak *et al.*, 1996; 2003] and argue that the latter is too strict in that it rejects reasonable belief revision operators.

The rest of the paper is organized as follows. In the next section, we recall the classical AGM approach in a propositional setting as formulated by [Katsuno and Mendelzon, 1991], followed by the approach of [Darwiche and Pearl, 1994] for iterated belief revision. In Section 3, we formally analyze the problem of the DP postulates to be overly permissive. In Section 4, we present an additional postulate to overcome this deficiency. We give a representation theorem for the postulate along with a concrete revision operator. We conclude in Section 5 with a detailed comparison to related work.

2 Preliminaries

It has been observed by many researchers that belief sets alone are not sufficient to determine a unique strategy for belief revision [Gärdenfors and Makinson, 1988; Spohn, 1988]. Any concrete belief revision operator requires additional information concerning the firmness of different beliefs to determine the revision strategy. In particular, this (extra-logical) information should uniquely determine a set of *conditional beliefs*: An agent is said to hold a conditional belief $\alpha \gg \beta$ (with α, β sentences in $\mathcal{L}$) precisely when

it will believe β after a revision with α [Gärdenfors, 1988; Boutilier, 1993]. The *Triviality Theorem* of [Gärdenfors and Makinson, 1988] shows that using the AGM postulates it is improper to include conditional beliefs into the belief sets. As a consequence, we need to distinguish a belief set (referred to as *propositional beliefs*) from a *belief state* (also called *epistemic state*). The latter contains, in addition to its belief set, the conditional beliefs which determine the revision strategy.

2.1 (Modified) KM Postulates

Katsuno and Mendelzon (KM) rephrased the AGM postulates for the propositional setting [Katsuno and Mendelzon, 1991]. The beliefs of an agent are represented by a sentence ψ in a finitary propositional language $\mathcal{L}$.¹ Any new evidence is a sentence μ in $\mathcal{L}$, and the result of revising ψ with μ is also a sentence (denoted as $\psi * \mu$) belonging to $\mathcal{L}$.

To avoid an inconsistency with their postulates for iterated revision, [Darwiche and Pearl, 1994; 1997] suggested to weaken the original KM postulates by regarding belief revision operators as functions on belief states (rather than on belief sets). This resulted in the Postulate (R*1)–(R*6) shown below.

For the sake of the simplicity, we will abuse notation by using interchangeably a belief state Ψ and its belief set $Bel(\Psi)$. For example, Ψ and $\Psi * \mu$ in Postulate (R*1) refer, respectively, to the current belief state and to the posterior belief state; and $\Psi * \mu \models \mu$ is just shorthand for $Bel(\Psi * \mu) \models \mu$. The following are the (modified) KM postulates, which in turn are a reformulation of the AGM postulates:

- (R*1) $\Psi * \mu \models \mu$.
- (R*2) If $\Psi \wedge \mu$ is satisfiable, then $\Psi * \mu \equiv \Psi \wedge \mu$.
- (R*3) If μ is satisfiable, then $\Psi * \mu$ is also satisfiable.
- (R*4) If $\Psi_1 = \Psi_2$ and $\mu_1 \equiv \mu_2$, then $\Psi_1 * \mu_1 \equiv \Psi_2 * \mu_2$.
- (R*5) $(\Psi * \mu) \wedge \phi \models \Psi * (\mu \wedge \phi)$.
- (R*6) If $(\Psi * \mu) \wedge \phi$ is satisfiable, then $\Psi * (\mu \wedge \phi) \models (\Psi * \mu) \wedge \phi$.

Darwiche and Pearl have given a representation theorem for Postulates (R*1)–(R*6) wrt. a revision mechanism based on total pre-orders over possible worlds:

Definition 2.1 Let $\mathcal{W}$ be the set of all worlds (interpretations) of a propositional language $\mathcal{L}$. A function that maps each belief state Ψ to a total pre-order $\leq_\Psi$ on $\mathcal{W}$ is called a *faithful assignment* iff

- If $w_1, w_2 \models \Psi$, then $w_1 =_\Psi w_2$.
- If $w_1 \models \Psi$ and $w_2 \not\models \Psi$, then $w_1 <_\Psi w_2$.
- If $\Psi = \Phi$, then $\leq_\Psi = \leq_\Phi$.

where $w_1 =_\Psi w_2$ is defined as $w_1 \leq_\Psi w_2$ and $w_2 \leq_\Psi w_1$; and $w_1 <_\Psi w_2$ means $w_1 \leq_\Psi w_2$ and $w_2 \not\leq_\Psi w_1$.

The intuitive meaning of $w_1 \leq_\Psi w_2$ is that w_1 is at least as plausible as w_2 in Ψ .

¹We assume that the corresponding belief set $Bel(\psi)$ contains all logical consequences of ψ .

Theorem 2.2 [Darwiche and Pearl, 1997] A revision operator $*$ satisfies Postulates (R*1)–(R*6) iff there exists a faithful assignment that maps a belief state Ψ to a total pre-order $\leq_\Psi$ such that

$$Mods(\Psi * \mu) = \min(Mods(\mu), \leq_\Psi)$$

Here, $Mods(\mu)$ denotes the set of all models of μ , and $\min(Mods(\mu), \leq_\Psi)$ is the set of minimal elements of $Mods(\mu)$ wrt. the pre-order $\leq_\Psi$.

2.2 DP Postulates

In order to allow for successive revisions, in each revision step it must be fully specified how the conditional beliefs are to be modified. Following the principle of economy of information, minimal change should also be applied to the conditional beliefs. Unfortunately, although the KM postulates put much emphasis on the preservation of propositional beliefs, they do not constrain the modification of conditional beliefs. Darwiche and Pearl have shown that the KM postulates alone are too weak to adequately characterize iterated belief revision, because they support unreasonable revision behaviors [Darwiche and Pearl, 1994]. To overcome this deficiency, they proposed these four additional postulates [Darwiche and Pearl, 1997]:

- (C1) If $\beta \models \mu$, then $(\Psi * \mu) * \beta \equiv \Psi * \beta$.
- (C2) If $\beta \models \neg\mu$, then $(\Psi * \mu) * \beta \equiv \Psi * \beta$.
- (C3) If $\Psi * \beta \models \mu$, then $(\Psi * \mu) * \beta \models \mu$.
- (C4) If $\Psi * \beta \not\models \neg\mu$, then $(\Psi * \mu) * \beta \not\models \neg\mu$.

Motivation and interpretation of these postulates can be found in [Darwiche and Pearl, 1994; 1997].

To provide formal justifications, Darwiche and Pearl have presented an extension of the above representation theorem for Postulates (C1)–(C4):

Theorem 2.3 [Darwiche and Pearl, 1997] Suppose that a revision operator satisfies Postulates (R*1)–(R*6). The operator satisfies Postulates (C1)–(C4) iff the operator and its corresponding faithful assignment satisfy:

- (CR1) If $w_1, w_2 \models \mu$, then $w_1 \leq_\Psi w_2$ iff $w_1 \leq_{\Psi * \mu} w_2$.
- (CR2) If $w_1, w_2 \not\models \mu$, then $w_1 \leq_\Psi w_2$ iff $w_1 \leq_{\Psi * \mu} w_2$.
- (CR3) If $w_1 \models \mu$ and $w_2 \not\models \mu$, then $w_1 <_\Psi w_2$ implies $w_1 <_{\Psi * \mu} w_2$.
- (CR4) If $w_1 \models \mu$ and $w_2 \not\models \mu$, then $w_1 \leq_\Psi w_2$ implies $w_1 \leq_{\Psi * \mu} w_2$.

Furthermore, [Darwiche and Pearl, 1997] showed that their additional postulates are consistent with the (modified) KM postulates. They did so by defining a concrete revision operator which satisfies both (R*1)–(R*6) and (C1)–(C4).

2.3 Why Not Absolute Minimization?

A different approach to studying iterated belief revision is by defining concrete revision operators. For instance, [Boutilier, 1993] proposed a specific revision operator (known as *natural revision*) which satisfies the modified KM postulates and the following one:

- (CB) If $\Psi * \mu \models \neg\beta$, then $(\Psi * \mu) * \beta \equiv \Psi * \beta$.

As shown by [Darwiche and Pearl, 1997; Boutilier, 1996], postulate (CB) imposes absolute minimization on the change of conditional beliefs:

Theorem 2.4 *Suppose that a revision operator satisfies Postulates (R*1)–(R6). The operator satisfies Postulate (CB) iff the operator and its corresponding faithful assignment satisfy*

$$(CBR) \text{ If } w_1 \models \neg(\Psi * \mu) \text{ and } w_2 \models \neg(\Psi * \mu) \text{ then } w_1 \leq_{\Psi} w_2 \text{ iff } w_1 \leq_{\Psi * \mu} w_2$$

Condition (CBR) says that the agent should keep as much of the ordering information in Ψ as possible while satisfying the KM postulates. At first glance, therefore, it seems that Condition (CBR) complies with the principle of economy of information. However, Postulate (CB) is too radical since any severe revision necessarily cancels all previous evidences [Darwiche and Pearl, 1994; Zhang, 2004]. This suggests that absolute minimization of the change of conditional beliefs is overly strict and not desirable in general. It is easy to see that the DP postulates are a weakening of Postulate (CB), in the sense Postulate (CB) implies all of the DP postulates but not vice versa.

3 The Problem of Implicit Dependence

Although most counter-examples in [Darwiche and Pearl, 1997] against the KM postulates are solved by adding the DP postulates, several open problems remain. For instance, the DP postulates are consistent with (CB), hence they do not block counter-examples against natural revision, like the following [Darwiche and Pearl, 1997]:

Example 1 We encounter a strange new animal and it appears to be a bird, so we believe the animal is a bird. As it comes closer to our hiding place, we see clearly that the animal is red, so we believe that it is a red bird. To remove further doubts about the animal birdhood, we call in a bird expert who takes it for examination and concludes that it is not really a bird but some sort of mammal. The question now is whether we should still believe that the animal is red.

As argued in [Darwiche and Pearl, 1997], we have every reason to keep our belief that the animal is red, since birdhood and color are not correlated. However, natural revision enforces us to give up the belief of the animal's color: According to Postulate (CB), since $bird * red \models \neg(\neg bird)$ it follows that $(bird * red) * \neg bird \equiv bird * \neg bird$.

In being compatible with (CB), the DP postulates are not strong enough to guarantee that the belief of the animal's color is retained. This can be intuitively explained as follows: After observing the animal's color, we are actually acquiring a new conditional belief, namely, that the animal is red even if it were not a bird i.e., $\neg bird \gg red$. However, the DP postulates do not enforce the acquisition of conditional beliefs. In the sequel, we first give a formal analysis of this weakness of the DP postulates, and then we present an additional postulate by which this problem is overcome.

It is well known (see, e.g., [Gärdenfors and Makinson, 1988]) that if a belief state Ψ suffices to uniquely determine a revision strategy that satisfies the AGM (or the KM) postulates, then the belief state determines a unique, total pre-order

$\leq_{\Psi}$ (known as *epistemic entrenchment*) over $\mathcal{L}$ which satisfies certain conditions. Given an epistemic entrenchment, the corresponding belief revision operator is defined by the following condition: For any β ,

$$(C^*) \Psi * \mu \models \beta \text{ if either } \models \neg \mu \text{ or } \neg \mu <_{\Psi} \neg \mu \vee \beta$$

Other forms of total pre-orderings on $\mathcal{L}$ have been proposed, e.g., [Rott, 1991; Williams, 1992]. All of these orderings require extra-logical information, that is, they cannot be determined by pure logical relations among the sentences. Although we focus on epistemic entrenchment, the analysis that follows does not depend on this specific ordering; it can be easily adapted to the approaches just mentioned.

To begin with, we define the notion of dependence between sentences wrt. a belief state as follows [Fariñas del Cerro and Herzig, 1996]:

Definition 3.1 A sentence β *depends* on another sentence μ in belief state Ψ precisely when $\Psi \models \beta$ and $\Psi * \neg \mu \not\models \beta$. Two sentences μ, β are called *dependent* in Ψ if either μ depends on β or β depends on μ in Ψ .

Consider, now, a (non-tautological) new evidence μ . Whenever $\Psi \models \beta$, condition (C*) implies that if $\mu \geq_{\Psi} \mu \vee \beta$, then β is (implicitly) dependent on μ in Ψ . This kind of dependency could be problematic. In particular, it is possible that two initially independent sentences become, undesirably, dependent after a revision step. In Example 1, say, *red* becomes dependent on *bird* after revising by *red* if natural revision is used.

Epistemic entrenchment reveals that the problem of natural revision is that it assigns the lowest degree of belief to a new evidence without asserting conditional beliefs for independence. Thus the new evidence depends on all other beliefs which survive the revision process. This explains why severe revision always cancels all previous evidences.

4 A Postulate of Independence

The analysis in the previous section shows that in order to overcome the problem of implicit dependence, the revision operator must explicitly assert some conditional beliefs. It is easy to see that the DP postulates only require the *preservation* of conditional beliefs when a belief state Ψ is revised with μ : Postulates (C1) and (C2) neither require to add nor to remove certain conditional beliefs (namely, those conditioned on β) in case $\beta \models \mu$ or $\beta \models \neg \mu$; Postulate (C3) requires to retain the conditional belief $\beta \gg \mu$; finally, Postulate (C4) requires not to obtain the new conditional belief $\beta \gg \neg \mu$. Since none of the DP postulates requires to make independence assumptions, a new postulates is necessary to avoid undesired dependencies.

The revision process may introduce undesirable dependencies in both directions. That is, it could be that the new evidence becomes dependent on existing beliefs, or the other way around. Prior to stating the new postulate, we show that the DP postulates impose some constraints on the retention of the independence information in one direction. In the presence of the KM postulates, Postulate (C2) implies the following:

$$(C2') \text{ If } \Psi * \neg \mu \models \beta, \text{ then } (\Psi * \mu) * \neg \mu \models \beta$$

This essentially means that if β is not dependent on the new evidence μ in Ψ , then it also does not depend on μ in $\Psi * \mu$.

In order to ensure the explicit assertion of independence information in the other direction, we propose the following postulate of *Independence* to complement the DP postulates:

(Ind) If $\Psi * \beta \not\models \neg \mu$ then $(\Psi * \mu) * \beta \models \mu$

Basically, Postulate (Ind) says that if the belief in $\beta \supset \neg \mu$ is not strongly held in Ψ , then μ does not depend on $\neg \beta$ in $\Psi * \mu$.

Postulate (Ind) is sufficient to overcome the problem of implicit dependence, as can be shown by reconsidering Example 1. According to (Ind), $(bird * red) * \neg bird \models red$, given that $bird * \neg bird \not\models \neg red$. This shows that the new postulate blocks unreasonable behaviors which are admitted by the DP postulates. In Section 5, we will also argue that Postulate (Ind) is not overly strict.

In order to formally justify our new postulate, we will first provide a representation theorem along the line of Theorem 2.3. Thereafter, we will give a concrete belief revision operator which satisfies (Ind).

4.1 A Representation Theorem

For the proof of the representation theorem, we need the following observation, which is a consequence of Theorem 2.2.

Observation 4.1 Suppose that a revision operator satisfies Postulates (R*1)–(R*6). If $\not\models \neg \beta$, then $\Psi * \beta \models \mu$ precisely when there exists a world w such that $w \models \mu \wedge \beta$ and $w <_{\Psi} w'$ for any $w' \models \neg \mu \wedge \beta$, where $\leq_{\Psi}$ is the corresponding faithful assignment.

Theorem 4.2 Suppose that a revision operator satisfies Postulates (R*1)–(R*6). The operator satisfies Postulate (Ind) iff the operator and its corresponding faithful assignment satisfy:

(IndR) If $w_1 \models \mu$ and $w_2 \models \neg \mu$, then $w_1 \leq_{\Psi} w_2$ implies $w_1 <_{\Psi * \mu} w_2$.

Proof: “ $\Leftarrow$ ”: Assume $\Psi * \beta \not\models \neg \mu$. From Observation 4.1, it follows that for any world $w \models \beta \wedge \neg \mu$, there exists another world $w' \models \beta \wedge \mu$ such that $w' \leq_{\Psi} w$. Hence, since $\leq_{\Psi}$ is total, there must be a world w_1 such that $w_1 \models \mu \wedge \beta$ and $w_1 \leq_{\Psi} w_2$ for any $w_2 \models \neg \mu \wedge \beta$. Condition (IndR) then implies that $w_1 <_{\Psi * \mu} w_2$ for any $w_2 \models \neg \mu \wedge \beta$. Due to Observation 4.1, we have $(\Psi * \mu) * \beta \models \mu$.

“ $\Rightarrow$ ”: Assume $w_1 \models \mu$, $w_2 \models \neg \mu$, and $w_1 \leq_{\Psi} w_2$. Let β be such that $\text{Mods}(\beta) = \{w_1, w_2\}$. From Theorem 2.2, it follows that $w_1 \in \text{Mods}(\Psi * \beta)$. Hence $\Psi * \beta \not\models \neg \mu$. Postulate (Ind) implies $(\Psi * \mu) * \beta \models \mu$. Due to Postulates (R*1) and (R*3), $\text{Mods}((\Psi * \mu) * \beta) = \{w_1\}$. From Theorem 2.2, it follows that $w_1 <_{\Psi * \mu} w_2$. $\square$

An immediate consequence of Theorem 2.3 and 4.2 is that Postulate (Ind) implies both (C3) and (C4). Theorem 4.2 also shows that Postulate (Ind) is not overly constrained: Condition (IndR) only requires to decrease the plausibility of the worlds violating the new evidence μ and not to decrease the plausibility of the worlds confirming μ .

4.2 Properties of the Independence Postulate

We suggest to use the KM postulates along with Postulates (C1), (C2), and (Ind) to govern iterated belief revision. To show that these postulates together are consistent, we present a concrete revision operator which satisfies all of them. The operator is based on Spohn’s proposal of revising ordinal conditional functions [Spohn, 1988], which can be viewed as a qualitative version of Jeffrey’s Rule of probabilistic conditioning [Goldszmidt, 1992].

An ordinal conditional function is a function k from a given set of worlds into the class of ordinals. Intuitively, the ordinals represent degrees of plausibility. The lower its ordinal, the more plausible is a world. An ordinal conditional function encodes both a belief set and the conditional beliefs. The belief set $\text{Bel}(k)$ is the set of sentences which hold in all worlds of rank 0 (the smallest ordinal):

$$\text{Mods}(\text{Bel}(k)) = \{w \mid k(w) = 0\}$$

From now on, we use an ordinal conditional function and its belief set interchangeably; e.g., $\mu \in k$ means $\mu \in \text{Bel}(k)$, and $k \wedge \mu$ denotes $\bigwedge \text{Bel}(k) \wedge \mu$. A ranking of worlds can be extended to a ranking of sentences as follows:

$$k(\mu) = \begin{cases} (\max_w k(w)) + 1 & \text{If } \models \mu \\ \min_{w \models \neg \mu} k(w) & \text{Otherwise} \end{cases} \quad (1)$$

Put in words, the rank of a formula is the lowest rank of a world in which the formula does not hold. Hence, the higher the rank of a sentence, the firmer the belief in it. In fact, it is not hard to see that an ordinal conditional function k determines an epistemic entrenchment as follows:

$$\alpha \leq_k \beta \text{ iff } k(\alpha) \leq k(\beta)$$

Our belief revision operator allows to assign different plausibility degrees to new evidences; standard KM/DP revision is easily obtained as a special case by using a fixed value in all iterations [Darwiche and Pearl, 1997]. An ordinal conditional function k is revised according to new evidence μ with plausibility degree $m > 0$ as follows:

$$(k_{\mu, m}^*)(w) = \begin{cases} k(w) - k(\neg \mu) & \text{If } w \models \mu \\ k(w) + m & \text{If } w \models \neg \mu \end{cases} \quad (2)$$

Assuming the same degree of plausibility for any new evidence, satisfiability of the KM postulates along with Postulates (C1), (C2), and (Ind) by the revision defined in (2) is an almost direct consequence of Theorem 2.2, 2.3, and 4.2. But we can even prove a stronger result with varying plausibility values m .

Theorem 4.3 For any $m > 0$, the belief revision operator defined by (2) satisfies all KM postulates (R*1)–(R*6), where $\text{Bel}(\Psi)$ and $\Psi * \mu$ are, respectively, identified with $\bigwedge \text{Bel}(k)$ and $k_{\mu, m}^*$.

Lemma 4.4 Let k be an arbitrary ordinal conditional function and μ a new evidence with plausibility degree m , then for any non-tautological sentence β ,

$$k_{\mu, m}^*(\beta) = \begin{cases} k(\beta) + m & \text{If } \models \mu \supset \beta \\ t - k(\neg \mu) & \text{Else if } t = k(\beta) \\ \min(t - k(\neg \mu), k(\beta) + m) & \text{Otherwise} \end{cases}$$

where $t = k(\mu \supset \beta)$.

Theorem 4.5 For arbitrary $m_1, m_2 > 0$, the belief revision operator defined by (2) satisfies the following conditions:²

- (EC1) If $\alpha \models \mu$, then $(k_{\mu, m_1}^*)_{\alpha, m_2}^* \equiv k_{\alpha, m_2}^*$.
- (EC2) If $\alpha \models \neg\mu$, then $(k_{\mu, m_1}^*)_{\alpha, m_2}^* \equiv k_{\alpha, m_2}^*$.
- (EInd) If there exists m such that $k_{\neg\beta, m}^* \not\models \neg\mu$, then $(k_{\mu, m_1}^*)_{\neg\beta, m_2}^* \models \mu$.

Proof: If $\models \neg\alpha$, Condition (EC1) holds trivially. Assume that $\alpha \models \mu$ and $\not\models \neg\alpha$. By (2),

$$k_{\alpha, m_2}^*(w) = 0 \text{ iff } w \models \alpha \text{ and } k(w) = k(\neg\alpha) \quad (3)$$

Likewise,

$$(k_{\mu, m_1}^*)_{\alpha, m_2}^*(w) = 0 \text{ iff } w \models \alpha \text{ and } k_{\mu, m_1}^*(w) = k_{\mu, m_1}^*(\neg\alpha) \quad (4)$$

Since $\alpha \models \mu$, for any $w \models \alpha$ we have $k_{\mu, m_1}^*(w) = k(w) - k(\neg\mu)$ by (2). Since $\mu \supset \neg\alpha \equiv \neg\alpha$ and $\not\models \neg\alpha$, it follows from Lemma 4.4 that $k_{\mu, m_1}^*(\neg\alpha) = k(\neg\alpha) - k(\neg\mu)$. Hence, (4) is equivalent to

$$(k_{\mu, m_1}^*)_{\alpha, m_2}^*(w) = 0 \text{ iff } w \models \alpha \text{ and } k(w) = k(\neg\alpha)$$

This and (3) implies $(k_{\mu, m_1}^*)_{\alpha, m_2}^* \equiv k_{\alpha, m_2}^*$. Condition (EC2) can be proved analogously.

We prove Condition (EInd) by contradiction. To begin with, from the assumption that $k_{\neg\beta, m}^* \not\models \neg\mu$ it follows that $\not\models \beta$ and $\not\models \mu \supset \beta$. Furthermore, there exists w such that $k_{\neg\beta, m}^*(w) = 0$, $w \models \neg\beta \wedge \mu$, and $k(w) = k(\beta)$. With the help of (1), this implies $k(\beta) = k(\mu \supset \beta)$.

Now assume that $(k_{\mu, m_1}^*)_{\neg\beta, m_2}^* \not\models \mu$. It follows that there exists w' such that $(k_{\mu, m_1}^*)_{\neg\beta, m_2}^*(w') = 0$, $w' \models \neg\beta \wedge \neg\mu$, and $k_{\mu, m_1}^*(w') = k_{\mu, m_1}^*(\beta)$. Since $k(w) = k(\beta)$ and $w' \models \neg\beta$, we have $k(w') \geq k(w)$. Hence by (2), $k_{\mu, m_1}^*(w') = k(w') + m_1 > k(\beta)$. But from Lemma 4.4 it follows that $k_{\mu, m_1}^*(\beta) \leq k(\beta)$, since $\not\models \beta$ and $\not\models \mu \supset \beta$. This contradicts $k_{\mu, m_1}^*(w') = k_{\mu, m_1}^*(\beta)$. $\square$

Theorem 4.3 and 4.5 show that Postulate (Ind) is consistent with the KM and DP postulates. On the other hand, (Ind) does not follow from these postulates, which can be seen by the fact that (Ind) is incompatible with (CB), the postulate that characterizes natural revision.

It is worth mentioning that revisions based on ordinal conditional functions are particularly suitable for implementations of belief revision. For instance, [Jin and Thielscher, 2004] presented a method (and its implementation) for revision of belief bases which is equivalent to the belief revision defined by (2).

5 Related Work and Conclusions

The first DP approach [Darwiche and Pearl, 1994] seemed excessively strong because one of the postulates, (C2), is inconsistent with the classical AGM framework, as pointed

out by [Freund and Lehmann, 1994]. Apart from the solution by [Darwiche and Pearl, 1997], namely, to use a modified version of KM (which we followed in this paper), several other proposals were made to overcome the inconsistency. For instance, Lehmann proposed a different way of weakening the AGM postulates accompanied by quite different additional postulates for iterated revision, in which a belief state consists of a sequence of revision sentences [Lehmann, 1995]. Nayak *et al.* suggested to retain the original AGM framework and to consider a belief revision operator as a unary function (associated with a belief state). The idea is to view belief revision as dynamic, in the sense that the operator evolves after each step [Nayak *et al.*, 1996; 2003]. On the other hand, one point has been generally accepted, namely, that iterated belief revision should not be considered as a purely set-theoretical change of belief sets but as an evolution of belief states, which encapsulate beliefs with some kind of information of the firmness of beliefs [Zhang, 2004].

The problem of the DP postulates to be overly permissive was already studied by Nayak *et al.*, [1996; 2003]. They suggested to strengthen the DP postulates by the following so-called postulate of Conjunction:

(Conj) If $\mu \not\models \neg\beta$, then $(\Psi * \mu) * \beta \equiv \Psi * (\beta \wedge \mu)$.

It is easy to see that (Conj) implies another postulate, which they called postulate of Recalcitrance:

(Rec) If $\not\models \beta \supset \neg\mu$, then $(\Psi * \mu) * \beta \models \mu$.

In the following, we argue that Postulate (Conj), while strengthening the DP postulates, is overly strict. To this end, we show that even Postulate (Rec) is too strong. The latter says that, as long as $\beta \supset \neg\mu$ is not a tautology, it should be canceled after a successive revision by μ followed by β , no matter how strong the initial belief in $\beta \supset \neg\mu$. A simple example shows that this behavior is not reasonable:

Example 2 All her childhood, Alice was taught by her parents that a person who has told a lie is not a good person. So Alice believed, initially, that if Bob has told a lie then he is not a good person. After her first date with Bob, she began to believe that he is a good guy. Then a reliable friend of Alice warns her that Bob is in fact a liar, and Alice chooses to believe her. Now, should Alice still believe that Bob is a good guy?

According to Postulate (Rec), Alice should not challenge Bob's morality and still believe he is good, and hence to disbelieve what her parents taught her. But it is at least as reasonable to give up the belief that Bob is good. This shows that Postulate (Rec) is too strict a criterion for belief revision operators.

With regard to the postulate we have proposed in this paper, it is easy to see that (Ind) is a weakening of Postulate (Rec). This raises the question whether Postulate (Ind) weakens too much. Let us consider an example, taken from [Nayak *et al.*, 1996], which, at first glance, seems to show that this is indeed the case.

Example 3 Our agent believes that Tweety is a singing bird. However, since there is no strong correlation between singing and birdhood, the agent is prepared to retain the belief that

²Note that, as before, we abuse notation by simply writing $k_{\mu, m}^*$ instead of $\bigwedge Bel(k_{\mu, m}^*)$ etc.

Tweety sings even after accepting the information that Tweety is not a bird, and conversely, if the agent were to be informed that Tweety does not sing, she would still retain the belief that Tweety is a bird. Imagine that the agent first receives the information that Tweety is in fact not a bird, and later learns that Tweety does not sing.

Nayak *et al.* claimed that it is only reasonable to assume that the agent should, in the end, believe that Tweety is a non-singing non-bird. Indeed, with $\Psi \equiv \textit{singing} \wedge \textit{bird}$ it follows from Postulate (Rec) that $(\Psi * \neg \textit{bird}) * \neg \textit{singing} \models \neg \textit{bird}$, since $\not\models \neg \textit{singing} \supset \textit{bird}$. Postulate (Ind), on the other hand, does not apply in this case. But the behavior which is claimed to be the only reasonable one is not generally justified. Suppose, for example, the agent initially believes firmly that $\neg \textit{singing} \supset \textit{bird}$. It is then possible, after revising by $\neg \textit{bird}$, that the belief in $\neg \textit{singing} \supset \textit{bird}$ is stronger than the belief in $\neg \textit{bird}$. In this case, after further revising by $\neg \textit{singing}$, the agent believes that Tweety is a bird after all.

In [Zhang, 2004], it has also been argued that Postulate (Rec) is too radical because only those revision operators are admissible which assign the highest belief degree to the new evidence. This can be considered a further justification of our Postulate (Ind). Moreover, it is easy to verify that Postulate (Ind) is satisfied by all revision operators with memory [Konieczny and Pérez, 2000].

We have formally justified our new postulate by a representation theorem and a concrete revision operator which satisfies both (Ind) and KM/DP. Other concrete revision operators have been proposed, e.g., in [Williams, 1995], which also use ordinal conditional functions. The main difference is that these operators do not satisfy the Independence postulate.

References

- [Alchourrón *et al.*, 1985] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic* 50(2), pages 510–530, 1985.
- [Boutilier, 1993] C. Boutilier. Revision sequences and nested conditionals. In *Proceedings of IJCAI*, pages 519–525, 1993.
- [Boutilier, 1996] C. Boutilier. Iterated revision and minimal change of conditional beliefs. *Journal of Philosophical Logic*, 25(3), 1996.
- [Darwiche and Pearl, 1994] A. Darwiche and J. Pearl. On the logic of iterated belief revision. In R. Fagin, editor, *Proceedings of TARK*, pages 5–23. Kaufmann, San Francisco, CA, 1994.
- [Darwiche and Pearl, 1997] A. Darwiche and J. Pearl. On the logic of iterated belief revision. *Artificial Intelligence* 89, 1–29, 1997.
- [Fariñas del Cerro and Herzig, 1996] L. Fariñas del Cerro and A. Herzig. Belief change and dependence. In Yoav Shoham, editor, *Proceedings of TARK*, pages 147–162. Morgan Kaufmann, 1996.
- [Freund and Lehmann, 1994] M. Freund and D. Lehmann. Belief revision and rational inference. Technical Report TR-94-16, Institute of Computer Science, The Hebrew University of Jerusalem, Jerusalem, 91904, Israel, 1994.
- [Gärdenfors and Makinson, 1988] P. Gärdenfors and D. Makinson. Revisions of knowledge systems using epistemic entrenchment. In *Proceedings of TARK*, pages 83–95, Asilomar, CA, 1988.
- [Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, 1988.
- [Goldschmidt, 1992] M. Goldschmidt. *Qualitative probabilities: a normative framework for commonsense reasoning*. PhD thesis, University of California at Los Angeles, 1992.
- [Jin and Thielscher, 2004] Y. Jin and M. Thielscher. Representing beliefs in the fluent calculus. In R. L. de Mántas and L. Saitta, editors, *Proceedings of ECAI*, pages 823–827, Valencia, Spain, August 2004. IOS Press.
- [Katsuno and Mendelzon, 1991] H. Katsuno and A. O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294, 1991.
- [Konieczny and Pérez, 2000] S. Konieczny and R. Pérez. A framework for iterated revision. *Journal of Applied Non-Classical Logics*, 10(3-4), 2000.
- [Lehmann, 1995] D. J. Lehmann. Belief revision, revised. In C. S. Mellish, editor, *Proceedings of IJCAI*, pages 1534–1540, 1995.
- [Nayak *et al.*, 1996] A. C. Nayak, N. Y. Foo, M. Pagnucco, and A. Sattar. Changing conditional beliefs unconditionally. In Yoav Shoham, editor, *Proceedings of TARK*, pages 119–135. Morgan Kaufmann, San Francisco, 1996.
- [Nayak *et al.*, 2003] A. C. Nayak, M. Pagnucco, and P. Pappas. Dynamic belief revision operators. *Artificial Intelligence*, 146(2):193–228, 2003.
- [Rott, 1991] H. Rott. A nonmonotonic conditional logic for belief revision. In A. Fuhrmann and M. Morreau, editors, *The logic of theory change*, pages 135–183. Springer, Berlin, 1991.
- [Spohn, 1988] W. Spohn. Ordinal conditional functions: A dynamic theory of epistemic state. In William L. Harper and Brian Skyrms, editors, *Causation in Decision: Belief, Change and Statistics: Proceedings of the Irvine Conference on Probability and Causation*, volume II, pages 105–134, Dordrecht, 1988. Kluwer Academic Publisher.
- [Williams, 1992] M.-A. Williams. Two operators for theory bases. In *Proceedings of the Australian Joint Artificial Intelligence Conference*, pages 259–265. World Scientific, 1992.
- [Williams, 1995] M.-A. Williams. Iterated theory base change: A computational model. In C. S. Mellish, editor, *Proceedings of IJCAI*, pages 1541–1547, 1995.
- [Zhang, 2004] D. Zhang. Properties of iterated multiple belief revision. In *7th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 314–325. Springer, 2004.

Reasoning under inconsistency: the forgotten connective

Sébastien Konieczny
CRIL - Université d'Artois
62307 Lens, France
konieczny@cril.univ-artois.fr

Jérôme Lang
IRIT - Université Paul Sabatier
31062 Toulouse, France
lang@irit.fr

Pierre Marquis
CRIL - Université d'Artois
62307 Lens, France
marquis@cril.univ-artois.fr

Abstract

In many frameworks for reasoning under inconsistency, it is implicitly assumed that the formulae from the belief base are connected using a weak form of conjunction. When it is consistent, a belief base $B = \{\varphi_1, \dots, \varphi_n\}$, where the φ_i are propositional formulae, is logically equivalent to the base $\{\varphi_1 \wedge \dots \wedge \varphi_n\}$. However, when it is not consistent, both bases typically lead to different conclusions. This illustrates the fact that the comma used in base B has to be considered as an additional, genuine connective, and not as a simple conjunction. In this work we define and investigate a propositional framework with such a “comma connective”. We give it a semantics and show how it generalizes several approaches for reasoning from inconsistent beliefs.

1 Introduction

There are many different situations that can lead to reason with sets of (mutually) inconsistent formulae (see e.g. [Bloch and Hunter, 2001] for a classification).

When dealing with an inconsistent belief base, a rational agent must refrain from using classical entailment to exploit her belief base, since from an inconsistent belief base, every formula is a logical consequence.

Many approaches have been proposed so far to address the *paraconsistency issue*, i.e., the problem of designing inference relations that do not necessarily trivialize when the belief base is inconsistent, but preserve many expected consequences nevertheless. Among them, many approaches to avoid trivialization in presence of a contradiction consist in *weakening the inconsistent belief base by inhibiting some formulae* in it. To be more precise, the key idea of such approaches is to select some consistent subsets of the belief base B , then to reason deductively from some of them.

Clearly enough, such approaches require a belief base to be a *set* of formulae, not a single formula (inhibition consists in removing formulae from the set). As a consequence, a (finite) inconsistent set of formulae cannot be considered as equivalent to the conjunction of its elements (and this contrasts with what happens when consistent sets are considered). For instance, the set of consequences of $\{\varphi, \neg\varphi, \psi\}$ typically differs from the set of consequences of $\{\varphi \wedge \neg\varphi \wedge \psi\}$.

Handling in a different way conjunctions of formulae and sets of formulae when they are classically inconsistent is an idea which underlies many paraconsistent logics for decades. Such logics are called *non-adjunctive logics*, which just means that the inference rule which consists in considering the conjunctive formula $\phi \wedge \psi$ as derivable from the set $\{\varphi, \psi\}$ fails. For instance, Jaśkowski's discursive logic [Jaśkowski, 1969], one of the first paraconsistent logic pointed out so far, is non-adjunctive: in this logic, a propositional formula φ is considered as a consequence of a set of formulae $\{\varphi_1, \dots, \varphi_n\}$ if and only if $\diamond\varphi$ is a consequence of $\{\diamond\varphi_1, \dots, \diamond\varphi_n\}$ in epistemic logic **S5**; this amounts to say that φ is a consequence of $\{\varphi_1, \dots, \varphi_n\}$ in Jaśkowski's logic if and only if φ is a classical consequence of one of the φ_i 's. This approach has been further refined by Rescher and Manor [Rescher and Manor, 1970], who suggest to conjoin up to maximal consistency, and then by many AI researchers who introduce conjoining policies that can take advantage of additional information about the relative certainty of pieces of belief.

Nevertheless, as far as we know and apart of the recent work [Mengin, 2004] discussed in Section 8, considering comma as a plain *connective* has never been overstepped up to now, in particular the languages of previous paraconsistent logics do not enable sets of formulae to be nested. Defining and investigating a logical framework in which such a “comma connective” is used is the main purpose of this paper, which is organized as follows. After some formal preliminaries, we present both the syntactical and the semantical aspects of the “comma logic” in Section 3. Some properties of the corresponding inference relation are given in Section 4. The generality of the proposed framework is discussed in Section 5. Some translatability results and some complexity results are given in Section 6. An alternative semantics for the comma connective based on cardinality is briefly discussed in Section 7. Due to space limitations, most proofs are only sketched. Section 8 concludes the paper.

2 Preliminaries

Let PS be a *finite* set of propositional variables: a, b, c , etc. $\mathcal{L}_{PS}$ is the propositional language built from PS , the constants $\top$ (verum) and $\perp$ (falsum) and the standard connectives $\{\wedge, \vee, \neg, \rightarrow\}$ in the usual way. Formulae of $\mathcal{L}_{PS}$ are noted with small Greek letters φ, ψ , etc. An interpretation (world) M is a total function from PS to $\{0, 1\}$, typically represented

as the set of all the variables of PS it satisfies. $\mathcal{M}_{PS}$ denotes the set of interpretations built over PS . M is a model of a formula φ , denoted $M \models \varphi$, if and only if it makes it true (in the usual truth-functional way). $Mod(\varphi)$ denotes the set of models of φ . $\models$ denotes classical entailment and $\equiv$ denotes equivalence between formulae from $\mathcal{L}_{PS}$.

Let $\subset$ denotes strict inclusion. Let $B = \{\varphi_1, \dots, \varphi_n\}$ be a finite set of formulae of $\mathcal{L}_{PS}$, and let $B' \subseteq B$, we note $\bigwedge B' = \bigwedge \{\varphi_i \mid \varphi_i \in B'\}$. B' is a *consistent* subset of B if and only if $\bigwedge B'$ is consistent. B' is a *maximal consistent* subset (or *maxcons*) of B if and only if B' is consistent and there is no consistent B'' such that $B' \subset B'' \subseteq B$. $MaxCons(B)$ denotes the set of the maximal consistent subsets of B .

3 The “comma logic”

Let us start with the definition of the language of our comma logic.

Definition 1 The language $\mathcal{L}_{PS}^C$ of the propositional comma logic is defined inductively as follows:

- $PS \cup \{\top, \perp\} \subseteq \mathcal{L}_{PS}^C$;
- If $\varphi \in \mathcal{L}_{PS}^C$ and $\psi \in \mathcal{L}_{PS}^C$ then $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ belong to $\mathcal{L}_{PS}^C$;
- If $n \geq 0$ and $\{\varphi_1, \dots, \varphi_n\} \subseteq \mathcal{L}_{PS}^C$ then $\langle \varphi_1, \dots, \varphi_n \rangle$ belongs to $\mathcal{L}_{PS}^C$.

Parentheses can be omitted when no ambiguity is possible.

Clearly enough, $\mathcal{L}_{PS}^C$ is a superset of $\mathcal{L}_{PS}$: nonclassical formulae obtained by taking advantage of the comma connective can be formed. Here, we abuse words slightly: comma is actually considered as a set constructor, hence formally comma denotes a family of connectives, one per possible arity. Especially, comma is not always binary: one can apply the comma connective to no formula at all (the empty set), one formula (singletons), or more than two formulae. This forces us to introduce additional separators (the chevrons $\langle$ and $\rangle$ – which must not be confused with parentheses) but enables to keep the notations not too heavy.

It proves also interesting to define the *depth* of a formula of $\mathcal{L}_{PS}^C$, and the languages $\{\mathcal{L}_{PS}^{C[n]}, n \in \mathbb{N}\}$ as:

Definition 2 The depth of a formula φ from $\mathcal{L}_{PS}^C$ is defined inductively as follows:

- If $\varphi \in PS \cup \{\top, \perp\}$ then $depth(\varphi) = 0$;
- $depth(\neg\varphi) = depth(\varphi)$;
- $depth(\varphi \wedge \psi) = depth(\varphi \vee \psi) = depth(\varphi \rightarrow \psi) = \max(depth(\varphi), depth(\psi))$;
- $depth(\langle \varphi_1, \dots, \varphi_n \rangle) = \max_{i=1 \dots n} depth(\varphi_i) + 1$.¹

For $n \in \mathbb{N}$, $\mathcal{L}_{PS}^{C[n]}$ is the set of all formulae of $\mathcal{L}_{PS}^C$ whose depth is less or equal to n .

One can easily check that $\mathcal{L}_{PS}^C = \bigcup_{n \in \mathbb{N}} \mathcal{L}_{PS}^{C[n]}$ and $\mathcal{L}_{PS}^{C[0]} = \mathcal{L}_{PS}$. Finally, for all $n \geq 1$, we define the language $\mathcal{L}_{PS}^{C[s,n]}$ (set of simple formulae of depth n) as the

¹By convention $depth(\langle \rangle) = 1$.

subset of $\mathcal{L}_{PS}^{C[n]}$ containing all formulae $\langle \varphi_1, \dots, \varphi_k \rangle$, where $\varphi_1, \dots, \varphi_k$ are formulae of $\mathcal{L}_{PS}^{C[n-1]}$, i.e., the set of formulae of depth n where comma is the main functor. Here are some examples of formulae of $\mathcal{L}_{PS}^C$, where $PS = \{a, b, c\}$:

- $\varphi_1 = a \wedge b \wedge (\neg b \vee c) \wedge \neg c$;
- $\varphi_2 = \langle a \wedge b, \neg b \vee c, \neg c \rangle$;
- $\varphi_3 = \langle a \wedge (b \vee \langle c, a \vee b, \neg c \rangle), \langle \neg b \vee c, \neg c \rangle \wedge \langle a, \top \rangle \rangle$;
- $\varphi_4 = \langle \rangle$.

We have $depth(\varphi_1) = 0$; $depth(\varphi_2) = 1$; $depth(\varphi_3) = 3$.

Note also that $\varphi_2 \in \mathcal{L}_{PS}^{C[s,1]}$ (as φ_4) and that $\varphi_3 \in \mathcal{L}_{PS}^{C[s,3]}$.

Let us now present the semantics of our new language:

Definition 3 The notion of satisfaction of a formula of $\mathcal{L}_{PS}^C$ by an interpretation M is given by the relation $\models_c$ on $\mathcal{M}_{PS} \times \mathcal{L}_{PS}^C$ defined inductively as:

- $M \models_c \top$;
- $M \not\models_c \perp$;
- If $a \in PS$, $M \models_c a$ if and only if $a \in M$;
- $M \models_c \neg\varphi$ if and only if $M \not\models_c \varphi$;
- $M \models_c \varphi \wedge \psi$ if and only if $M \models_c \varphi$ and $M \models_c \psi$;
- $M \models_c \varphi \vee \psi$ if and only if $M \models_c \varphi$ or $M \models_c \psi$;
- $M \models_c \varphi \rightarrow \psi$ if and only if $M \models_c \neg\varphi$ or $M \models_c \psi$;
- $M \models_c \langle \varphi_1, \dots, \varphi_n \rangle$ if and only if there is no interpretation $M' \in \mathcal{M}_{PS}$ such that $\{i \mid M' \models_c \varphi_i\}$ strictly contains $\{i \mid M \models_c \varphi_i\}$.

We now extend several standard notions (and the corresponding notations) from classical logic to comma logic. Let $Mod_C(\varphi) = \{M \in \mathcal{M}_{PS} \mid M \models_c \varphi\}$. A formula of $\mathcal{L}_{PS}^C$ is said to be *consistent* (or *satisfiable*) if and only if $Mod_C(\varphi) \neq \emptyset$, and *valid* if and only if $Mod_C(\varphi) = \mathcal{M}_{PS}$. If φ and ψ are two formulae of $\mathcal{L}_{PS}^C$, ψ is said to be a consequence of φ (denoted $\varphi \models_c \psi$) if and only if $Mod_C(\varphi) \subseteq Mod_C(\psi)$. When $\varphi = \top$, we simply write $\models_c \psi$ instead of $\top \models_c \psi$. Two formulae φ and ψ of $\mathcal{L}_{PS}^C$ are said to be *equivalent*, noted $\varphi \equiv_c \psi$, if and only if $\varphi \models_c \psi$ and $\psi \models_c \varphi$.

4 Some logical properties

It is easy to show that, in contrast to the other connectives in the logic, the comma connective is not truth-functional: in the general case, the truth value of $\langle \varphi_1, \dots, \varphi_n \rangle$ from $\mathcal{L}_{PS}^C$ in interpretation M cannot be determined from the truth values of each φ_i ($i \in 1 \dots n$) in M only. For instance, let $PS = \{a, b\}$, $M = \{a\}$, $\varphi_1 = \langle a, b \rangle$, $\varphi_2 = \langle a \wedge \neg b, b \rangle$: M satisfies the “first” element of φ_1 and φ_2 , but does not satisfy the “second” one; while M is a model of φ_2 , it is not a model of φ_1 . This lack of truth-functionality explains why the comma connective is really genuine in the logic. (It cannot be defined by just combining connectives of classical logic).

Actually, the discrepancy between classical logic and comma logic lies only in the comma connective, since:

Proposition 1 Comma logic is a conservative extension of classical logic: for every pair of formulae φ, ψ in which the comma connective does not occur, then $\varphi \models_c \psi$ if and only if $\varphi \models \psi$.

Obviously, new valid formulae are obtained in comma logic, like $\langle \varphi, \neg \varphi \rangle$, because the language of propositional logic has been enriched, but in any case classical theorems are preserved.

Furthermore, the replacement metatheorem of classical logic can be extended to comma logic:

Proposition 2 *Let φ be a valid formula from $\mathcal{L}_{PS}^C$ and let $x \in PS$. Let ψ be any formula from $\mathcal{L}_{PS}^C$. The formula obtained by replacing in φ every occurrence of x by an occurrence of ψ is valid as well.*

Let us now make precise some properties of the inference relation $\models_c$. First, a direct consequence of its definition is that $\models_c$ is a *Tarskian consequence relation* (since it is defined as model containment); from this observation, a number of immediate properties follows, especially the fact that $\models_c$ is reflexive and transitive. By the way, note that while in classical logic the basic entailment relation $\models$ on $\mathcal{L}_{PS} \times \mathcal{L}_{PS}$ can be easily extended to a relation on $2^{\mathcal{L}_{PS}} \times \mathcal{L}_{PS}$ by defining the models of a set of formulae as the interpretations satisfying every formula from the set, the similar construction would not be meaningful here; indeed, in comma logic, sets of formulae are interpreted in a nonstandard way. This renders the usual characterization of Tarskian consequence relations inadequate in our setting since $\models_c$ does satisfy neither strong reflexivity nor monotonicity:

- $\langle a, \neg a \rangle \not\models_c a$;
- $\langle a \rangle \models_c a$ but $\langle a, \neg a \rangle \not\models_c a$.

Here are some other valuable properties:

Proposition 3

1. If $\varphi_1 \models_c \varphi_2$ and $\langle \varphi_1, \varphi_2 \rangle \models_c \psi$, then $\varphi_1 \models_c \psi$.
2. If $\langle \varphi_1, \varphi_2 \rangle \models_c \psi$, then $\varphi_1 \models_c (\varphi_2 \rightarrow \psi)$.
3. If $\varphi_1 \equiv_c \varphi_2$ and φ_1 is a subformula of ψ , then any formula obtained by replacing in ψ occurrences of φ_1 by φ_2 is equivalent to ψ .
4. $\varphi \models_c \psi_1 \wedge \psi_2$ if and only if $\varphi \models_c \psi_1$ and $\varphi \models_c \psi_2$.
5. If $\langle \varphi, \varphi_1 \rangle \models_c \psi$ and $\langle \varphi, \varphi_2 \rangle \models_c \psi$, then $\langle \varphi, (\varphi_1 \vee \varphi_2) \rangle \models_c \psi$.

Item (1) is a form of the cut rule. Item (2) is a form of the deduction metatheorem. Item (3) is the substitution metatheorem for the “comma logic”. In particular, it shows that $\forall k$ if $\forall i \in 1 \dots k$ $\varphi_i \equiv_c \varphi'_i$ then $\langle \varphi_1, \dots, \varphi_k, \dots, \varphi_n \rangle \equiv_c \langle \varphi'_1, \dots, \varphi'_k, \dots, \varphi_n \rangle$. Item (4) is a form of the “and rule”. Item (5) is a form of proof by cases (or “or rule”).

Other standard metatheorems of classical logic fails in the general case for full comma logic, i.e., when sets of formulae are considered. For instance, modus ponens fails ($\langle a, (a \rightarrow \neg a) \rangle \not\models_c \neg a$), and-elimination and or-introduction as well ($\langle \perp, \varphi \rangle \not\models_c \perp$ and $\langle \perp \rangle \not\models_c \perp \vee \varphi$); reduction ad absurdum fails ($\varphi \models_c \varphi$ while $\langle \varphi, \neg \varphi \rangle \not\models_c \perp$).

Let us now present additional properties which state how the comma connective interacts with the other connectives (and itself).

Proposition 4

6. For any permutation σ of $\{1, \dots, n\}$, $\langle \varphi_1, \dots, \varphi_n \rangle$ is equivalent to $\langle \varphi_{\sigma(1)}, \dots, \varphi_{\sigma(n)} \rangle$.

7. $\langle \varphi_1, \dots, \varphi_1, \varphi_2, \dots, \varphi_n \rangle \equiv_c \langle \varphi_1, \dots, \varphi_n \rangle$.
8. $\forall n \geq 0$ $\langle \varphi_1, \dots, \varphi_n \rangle \not\models_c \perp$.
9. $\langle \varphi_1, \varphi_2, \dots, \varphi_n \rangle \equiv_c \langle \varphi_2, \dots, \varphi_n \rangle$ if φ_1 is inconsistent or $(\exists i \in 2 \dots n, \varphi_i \text{ is consistent and } \forall i \in 2 \dots n, \varphi_i \models \varphi_1)$.
10. $\varphi_1 \wedge \dots \wedge \varphi_n \models_c \langle \varphi_1, \dots, \varphi_n \rangle$.
11. $\langle \varphi_1, \dots, \varphi_n \rangle \wedge \langle \psi_1, \dots, \psi_m \rangle \models_c \langle \varphi_1 \wedge \psi_1, \dots, \varphi_1 \wedge \psi_m, \dots, \varphi_n \wedge \psi_1, \dots, \varphi_n \wedge \psi_m \rangle$.
12. $\langle \varphi_1, \dots, \varphi_n \rangle \wedge \langle \psi_1, \dots, \psi_m \rangle \models_c \langle \varphi_1, \dots, \varphi_n, \psi_1, \dots, \psi_m \rangle$.
13. If φ_i is consistent (for at least one $i \in 1 \dots n$), then $\langle \varphi_1, \dots, \varphi_n \rangle \models_c \varphi_1 \vee \dots \vee \varphi_n$.
14. If φ is consistent, then $\langle \varphi_1 \vee \varphi, \dots, \varphi_n \vee \varphi \rangle \equiv_c (\varphi_1 \wedge \dots \wedge \varphi_n) \vee \varphi$.
15. $\neg(\neg \varphi) \equiv_c \varphi$.

Item (6) shows that the “comma connective” is symmetric in every argument. Item (7) shows that the “comma connective” is idempotent.

Item (8) shows that for every $n > 0$, every formula from $\mathcal{L}_{PS}^{C[s,n]}$ is consistent (obviously, this cannot be extended to the full language $\mathcal{L}_{PS}^C$). This is sufficient to ensure that $\models_c$ achieves a form of paraconsistency when starting from a set of formulae.

Item (9) gives a sufficient condition to simplify a set of formulae (note that in contrast to classical logic, it is not possible in general to remove a formula from a set as soon as it is entailed by another formula; for instance, while $a \wedge b \models_c a \vee b$, we do not have $\langle a \wedge b, \neg a, a \vee b \rangle$ equivalent to $\langle a \wedge b, \neg a \rangle$).

Items (10), (11), (12) make precise the connection between the comma connective and conjunction. None of the converse relation holds in the general case (for (10), $\varphi_1 \wedge \dots \wedge \varphi_n$ should be consistent; the converse of (11) does not hold even in the restricted case $m = 1$ and $\langle \varphi_1, \dots, \varphi_n \rangle \wedge \psi_1$ is consistent: $\langle a \wedge \neg b, \neg a \wedge \neg b, \neg a \wedge \neg b \rangle \not\models_c \langle a, \neg a \vee b, \neg a \rangle \wedge \neg b$; the converse of (12) fails since for instance $\langle a, \neg a \rangle \not\models_c \langle a \rangle \wedge \langle \neg a \rangle$). Note also that there is in general no deductive connections between a formula where nested commas occur and its “unfold” counterpart. Especially, in general, we have:

$$\langle \langle \varphi_1, \dots, \varphi_n \rangle, \langle \psi_1, \dots, \psi_m \rangle \rangle \not\models_c \langle \varphi_1, \dots, \varphi_n, \psi_1, \dots, \psi_m \rangle$$

For example, $\langle \langle a, b \rangle, \langle \neg a, c \rangle \rangle \not\models_c \langle a, \neg a, b, c \rangle$.

In general, we also have:

$$\langle \varphi_1, \dots, \varphi_n, \psi_1, \dots, \psi_m \rangle \not\models_c \langle \langle \varphi_1, \dots, \varphi_n \rangle, \langle \psi_1, \dots, \psi_m \rangle \rangle$$

For example, $\langle a, \neg a, b, c \rangle \not\models_c \langle \langle a, b, \neg a \rangle, \langle \neg a, c \rangle \rangle$.

Item (13) and (14) make precise the connection between the comma connective and disjunction. No other significant properties involving disjunction hold; for instance $\langle a, b, \neg b \rangle \vee \neg a \not\models_c \langle a \vee \neg a, b \vee \neg a, \neg b \vee \neg a \rangle$. Together with the previous properties concerning conjunction, it shows that we can consider the comma connective as a “weak” conjunction connective, since it is equivalent to it when the conjunction is consistent, but gives a consistent result even when it is not. Furthermore, when all the formulae are pairwise contradictory but one of them is consistent, comma is equivalent to the disjunction connective. Thus, if we specialize it to the case two formulae φ and ψ are considered, we get the easy corollary:

Corollary 1 $\langle \varphi, \psi \rangle \equiv_c \begin{cases} \varphi \wedge \psi & \text{if consistent, else} \\ \varphi \vee \psi & \text{if consistent, else} \\ \top & \text{otherwise.} \end{cases}$

Item (15) finally states that negation is involutive for the full logic (in particular, $\neg(\neg\langle \varphi_1, \dots, \varphi_n \rangle) \equiv_c \langle \varphi_1, \dots, \varphi_n \rangle$). No other link between negation and the comma connective can be established in the general case (for instance, $\neg$ does not distribute over the “comma connective”; for instance, $\neg\langle \perp \rangle \equiv_c \perp$ while $\neg\langle \neg\perp \rangle \equiv_c \top$).

5 Generality of the framework

Let us now show how several approaches for dealing with inconsistency can be encoded in this logic.

5.1 Skeptical inference

One of the most common method for reasoning from an inconsistent set of formulae consists in selecting the maximal (w.r.t. set inclusion) consistent subsets (also called *maxcons*) of formulae (see [Rescher and Manor, 1970] for example). Consequences w.r.t. skeptical inference (also called “universal” inference) are then defined as classical consequences of every maxcons:

Definition 4 $\Delta \vdash_{\forall} \varphi$ iff $\forall M \in \text{MaxCons}(\Delta), M \models \varphi$.

Such an inference relation can be easily recovered in our comma logic:

Proposition 5 $\{\varphi_1, \dots, \varphi_n\} \vdash_{\forall} \varphi$ iff $\langle \varphi_1, \dots, \varphi_n \rangle \models_c \varphi$.

5.2 Credulous inference

From a set of maxcons, one can also consider credulous inference (also called “existential” inference), that allows to infer a formula if at least one maxcons classically entails it.

Definition 5 $\Delta \vdash_{\exists} \varphi$ iff $\exists M \in \text{MaxCons}(\Delta), M \models \varphi$.

Again, such an inference relation can be easily recovered in our comma logic:

Proposition 6

$\{\varphi_1, \dots, \varphi_n\} \vdash_{\exists} \varphi$ iff $\langle \varphi_1, \dots, \varphi_n, \neg\varphi \rangle \wedge \varphi \not\models_c \perp$.

5.3 Supernormal defaults

Assumption-based theories and supernormal default logic [Poole, 1988; Brewka, 1989] can be described in the following way. Let $\langle B, B_d \rangle$ be a pair, where B and B_d are two finite sets of formulae. B is the set of facts (or hard constraints), that is supposed to be consistent. B_d is a set of defaults, i.e., a set of formulae that one wishes to add to facts if it does not result in an inconsistency. B_d is often not consistent with B .

An extension is a subset of formulae of $B \cup B_d$ that contains all the formulae of B and a maximum (w.r.t. set inclusion) of formulae of B_d . So, the set of all the extensions is defined as: $\text{Extens}(\langle B, B_d \rangle) = \{E \mid E \subseteq B \cup B_d \text{ and } B \subseteq E \text{ and } E \not\models \perp \text{ and } \forall E' \text{ s.t. } E \subset E' \subseteq B \cup B_d, E' \models \perp\}$.

Inference from such a supernormal default theory $\langle B, B_d \rangle$ is defined as:

Definition 6 $\langle B, B_d \rangle \vdash_D \varphi$ iff $\forall E \in \text{Extens}(\langle B, B_d \rangle), E \models \varphi$.

This can be also easily expressed within the comma logic framework (note that since the set B of facts is consistent, it can be considered conjunctively):

Proposition 7 $\langle B, \{\varphi_1, \dots, \varphi_n\} \rangle \vdash_D \varphi$ if and only if $\langle \varphi_1 \wedge \dots \wedge \varphi_n \wedge \bigwedge B, \dots, \varphi_n \wedge \bigwedge B \rangle \wedge \bigwedge B \models_c \varphi$.

Finally, note that since circumscription can be encoded as skeptical inference from supernormal defaults (see e.g., [Etherington, 1987]), it can also be encoded in the comma logic framework.

5.4 Belief revision

The basic revision operator proposed by Nebel [Nebel, 1991; 1998; Fagin *et al.*, 1983] is defined as follows:

Definition 7 Let $B \perp \varphi = \{M \mid M \subseteq B \text{ and } M \not\models \varphi \text{ and } \forall M' \text{ s.t. } M \subset M' \subseteq B, M' \models \varphi\}$. Basic revision is then defined as: $B \circ_N \varphi = (\bigvee_{M \in B \perp \neg \varphi} \bigwedge M) \wedge \varphi$.

In comma logic, this can be expressed as follows:

Proposition 8 $\{\varphi_1, \dots, \varphi_n\} \circ_N \varphi \models \mu$ if and only if $\langle \varphi_1 \wedge \varphi, \dots, \varphi_n \wedge \varphi \rangle \wedge \varphi \models_c \mu$.

5.5 Belief merging

Suppose that one wants to merge several belief (or goal) bases while allowing some of those bases to be inconsistent. If one wants to take into account inconsistent belief bases in the merging process, one can first perform an intra-source merging in order to extract useful information from those bases, before making the proper inter-source merging.

Let $E = \{B_1, \dots, B_n\}$ be a set of (possibly inconsistent) belief bases s.t. for all $i \in 1 \dots n$, we have $B_i = \{\varphi_1^i, \dots, \varphi_{k_i}^i\}$. Then, we can define the merging of those bases as :

$$\Delta(E) = \langle \langle \varphi_1^1, \dots, \varphi_{k_1}^1 \rangle, \dots, \langle \varphi_1^n, \dots, \varphi_{k_n}^n \rangle \rangle$$

This operator is not definable as a two aggregation steps merging operator (as defined in [Konieczny *et al.*, 2002]). Conversely, considering an alternative semantics for the comma connective (see Section 7), we can capture some of the two aggregation steps merging operators in comma logic.

6 Computational aspects

Let us investigate both translatability issues and complexity issues for comma logic. First of all, one can prove that the expressiveness of the comma logic is exactly the same as the expressiveness of classical logic (every piece of information which can be encoded in one of the logics can also be encoded in the other one):

Proposition 9 For any formula φ of $\mathcal{L}_{PS}^C$, there is a formula ψ of $\mathcal{L}_{PS}^{C[0]} = \mathcal{L}_{PS}$ that is equivalent to φ .

A simple induction on the depth of the formulae shows how each formula φ of $\mathcal{L}_{PS}^C$ can be equivalently (for $\models_c$) rewritten into a classical formula $C(\varphi)$. Let φ be a formula of $\mathcal{L}_{PS}^C$ of depth k . A translation of it into a classical formula can be obtained as follows. First, replace every subformula of φ

that is in $\mathcal{L}_{PS}^{C[s,1]}$ by an equivalent formula from $\mathcal{L}_{PS}^{C[0]}$ (the substitution theorem for the “comma logic” makes it sound); one gets then a formula of depth $n - 1$, so we can iterate the process until a formula of depth 0 is obtained. For example, let $\varphi = \langle a, b, \langle \neg b \vee c, \neg c \rangle \rangle$; we start by translating $\langle \neg b \vee c, \neg c \rangle$ in $\neg b \wedge \neg c$: after one iteration, we get the formula $\langle a, b, \neg b \wedge \neg c \rangle$, and a second iteration gives $(a \wedge b) \vee (a \wedge \neg b \wedge \neg c)$, that is equivalent to $a \wedge (b \vee \neg c)$. The problem with such a translation is that the size of $C(\varphi)$ is generally exponentially larger than the size of φ . Can we do better? Actually, the answer is negative:

Proposition 10 *Under the usual assumptions of complexity theory,² there is no polysize function that maps any formula from $\mathcal{L}_{PS}^C$ (and even from $\mathcal{L}_{PS}^{C[1]}$) to*

- *an equivalent classical formula.*
- *an equivalent formula from $\mathcal{L}_{PS}^C$ in which the “comma connective” is binary (i.e., whenever it is used, it may connect at most two formulae).*

The first item is a direct consequence of the non-compilability of circumscription [Cadoli *et al.*, 1997]. The second is a consequence of the first one, given that every formula φ from $\mathcal{L}_{PS}^C$ in which the comma connective is at most binary can be turned into a classical formula whose size is bounded by $|\varphi|$ (the proof is easy by structural induction given Corollary 1).

Let us now turn to the inference issue. From Definition 3, it is easy to show that $\varphi \models_c \psi$ if and only if $\varphi \wedge \neg \psi$ is inconsistent. Hence it is sufficient to focus on the satisfiability issue. The satisfiability problem in $\mathcal{L}_{PS}^C$ is, of course, at least as hard than for classical logic (so it is NP-hard). Can we say more than that? Let us begin with $\mathcal{L}_{PS}^{C[1]}$.

Proposition 11 *The satisfiability problem in $\mathcal{L}_{PS}^{C[1]}$ is Σ_2^p -complete.*

Membership to Σ_2^p comes from the fact that the corresponding model checking problem is in coNP: for any simple subformula $\psi = \langle \alpha_1, \dots, \alpha_n \rangle$ of φ from $\mathcal{L}_{PS}^{C[s,1]}$, deciding whether $M \models_c \psi$ simply consists in guessing an interpretation M' and to check in polynomial time that the set of α_i satisfied by M' is a proper superset of the set of α_i satisfied by M . Actually, the truth values of all such ψ in M can be established in nondeterministic polynomial time (guess in parallel several M' , one for each simple subformula ψ of depth 1 which is not satisfied by M). Once this has been done, the truth value of φ in M can be computed in deterministic polynomial time. Hardness comes from Σ_2^p -hardness of credulous inference from maximal consistent subbases [Nebel, 1998; Cayrol and Lagasque-Schiex, 1995].

Now, what about the satisfaction problem when commas are nested? An iterated use of the proof of the preceding result (membership part) easily leads to the following:

Proposition 12

- *The satisfiability problem in $\mathcal{L}_{PS}^{C[n]}$ is in Σ_{n+1}^p .*
- *The satisfiability problem in $\mathcal{L}_{PS}^C$ is in PSPACE.*

²Especially, the fact that the polynomial hierarchy does not collapse.

We do not have hardness proofs but we conjecture them.³ Thus, we do not know how to perform comma elimination (for the nested occurrences) in polynomial time (and we conjecture that it is not possible to do it); especially, introducing new symbols would not necessarily preserve equivalence over the original language; for instance, $\langle a, \langle \neg a, b \rangle \rangle \not\models_c \langle a, new \rangle \wedge (new \Leftrightarrow \langle \neg a, b \rangle)$.

Let us now give a few comments about those complexity results. First, the fact that satisfiability in $\mathcal{L}_{PS}^{C[1]}$ is at the second level of the polynomial hierarchy, and not higher are good news. We know (from Section 5) that the universal and existential inference problems can be reduced, in polynomial time, to satisfaction or validity problems of formulae of $\mathcal{L}_{PS}^{C[1]}$. Given complexity results from [Cayrol *et al.*, 1998; Nebel, 1998], it follows that the satisfiability problem in $\mathcal{L}_{PS}^{C[1]}$ cannot be lower than the second level of the polynomial hierarchy. The fact that it is not higher shows that the gain of flexibility we get does not result in a complexity shift.

7 Variation on the “comma semantics”

Among others, another semantics for the comma connective is obtained by considering as models of $\langle \varphi_1, \dots, \varphi_n \rangle$, the interpretations that maximize the *number* of satisfied formulae φ_i . Formally, this amounts to replacing in Definition 3, the item $M \models_c \langle \varphi_1, \dots, \varphi_n \rangle$ by the following one:

- $M \models_c^* \langle \varphi_1, \dots, \varphi_n \rangle$ if and only if there is no interpretation $M' \in \mathcal{M}_{PS}$ such that $|\{i \mid M' \models_c \varphi_i\}| > |\{i \mid M \models_c \varphi_i\}|$.

Properties reported in Section 4 are typically preserved by this modification, but idempotence: $\langle \varphi_1, \varphi_1, \varphi_2, \dots, \varphi_n \rangle$ is (generally) not equivalent to $\langle \varphi_1, \varphi_2, \dots, \varphi_n \rangle$. For example, $\langle a, a, \neg a \rangle$ is not equivalent to $\langle a, \neg a \rangle$ under the cardinality maximality semantics.

Contrariwise to the $\models_c$ case, we can show that:

Proposition 13 *For every fixed n , there is a polysize function that maps any formula of $\mathcal{L}_{PS}^{C[n]}$ into a query-equivalent formula of $\mathcal{L}_{PS}$ when we appeal to the $\models_c^*$ semantics.*

The proof is a constructive one and relies on a translation function C similar to the one given in Section 6: we first replace each occurrence of the formulae $\psi = \langle \alpha_1, \dots, \alpha_n \rangle$ of $\mathcal{L}_{PS}^{C[s,1]}$ in φ by the *cardinality formulae* $[= k_\psi] : \langle \alpha_1, \dots, \alpha_n \rangle$, meaning that exactly k_ψ formulae among the n formulae $\alpha_1, \dots, \alpha_n$ must be satisfied (see e.g. [Benhamou *et al.*, 1994]); such cardinality formulae can be translated in polynomial space as classical query-equivalent formulae. Such a translatability result is helpful in the objective of compiling formulae from $\mathcal{L}_{PS}^{C[n]}$ into classical formulae so as to improve inference.

Proposition 14 *The satisfiability problem in $\mathcal{L}_{PS}^C$ for $\models_c^*$ is in Δ_2^p .*

³That the comma nesting level goes seemingly pairwise with a complexity increase is not surprising (a similar phenomenon happens when conditionals are nested, see e.g. [Eiter and Gottlob, 1993]).

Membership to Δ_2^P comes from the fact that one can associate to any simple subformula $\psi = \langle \alpha_1, \dots, \alpha_n \rangle$ of $\varphi \in \mathcal{L}_{PS}^C$ the maximal number k_ψ of α_i which can be jointly satisfied. Simple subformulae of φ are processed by increasing depth: for any ψ of depth 1, the α_i are classical formulae, and using binary search, logarithmically many calls to an NP oracle are sufficient to compute the corresponding k_ψ . For any ψ of depth 2, again using binary search, logarithmically many calls to an NP oracle are sufficient to compute the corresponding k_ψ : the key observation is that the model checking issue for ψ can be done in polynomial time once the $k_{\psi'}$ for each simple subformula ψ' of ψ have been computed ($M \models_c \psi'$ if and only if M satisfies $k_{\psi'}$ elements in ψ'), and so on. Since φ does not contain more than $|\varphi|$ simple subformulae, linearly many calls to an NP oracle are sufficient to compute all the k_ψ ; since model checking is in P once they have been computed, a last call to an NP oracle is sufficient to determine whether φ is satisfiable. We conjecture that the problem is Δ_2^P -complete but have only a proof of Θ_2^P -hardness at that time.

One has to note that the complexity gap (between the first and the second level of the polynomial hierarchy) highlighted by [Cayrol *et al.*, 1998] when one goes from the cardinality criterion to the set-inclusion one, becomes here an amazing gap (from at most Δ_2^P to PSPACE) – under the assumption of completeness for PSPACE, of course.

8 Conclusion

In this paper, we have given a “logic” for paraconsistent reasoning where comma is considered as a genuine connective. We have presented some properties of this logic, and we have shown that it allows to express, in a unified language, several problems of reasoning with inconsistent belief bases, non-monotonic inference, belief revision and belief merging. We have also provided some translatability results and some complexity results. Finally, we have given an alternative semantics for the comma connective.

A distinction between a finite set of formulas and the conjunction of them is also made in many substructural logics, i.e. logics weaker than the classical one because of the absence of some structural rules [Restall, 1999]. The “comma connective” in such logics is not necessarily commutative or idempotent. Unlike in our comma logic, it is binary and cannot be in the scope of every other connective under consideration in the language.

Another closely related work is [Mengin, 2004], which defined a more general framework than ours: it can be shown indeed that any formula of our language can be translated in a formula of his (which is a consequence of the equivalence given in his Section 3.3 for formulas of depth one together with our Proposition 2). Therefore, our logical properties and complexity results carry on to the framework of [Mengin, 2004].

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. This work has been supported by the Université d’Artois, the IRCICA Consortium, the Région Nord/Pas-de-Calais and the European Community FEDER program.

References

- [Benhamou *et al.*, 1994] B. Benhamou, L. Saïs, and P. Siegel. Two proof procedures for a cardinality based language in propositional calculus. In *STACS’94*, pages 71–82. IRISA, 1994.
- [Bloch and Hunter, 2001] I. Bloch and A. Hunter, editors. *Fusion: General Concepts and Characteristics*, volume 16 (10) of *International Journal of Intelligent Systems*. 2001.
- [Brewka, 1989] G. Brewka. Preferred subtheories: an extended logical framework for default reasoning. In *IJCAI’89*, pages 1043–1048, 1989.
- [Cadoli *et al.*, 1997] M. Cadoli, F.M. Donini, M. Schaerf, and R. Silvestri. On compact representations of propositional circumscription. *Theoretical Computer Science*, 182:183–202, 1997.
- [Cayrol and Lagasquie-Schiex, 1995] C. Cayrol and M.C. Lagasquie-Schiex. Non-monotonic syntax-based entailment: a classification of consequence relations. In *ESQARU’95*, pages 107–114, 1995.
- [Cayrol *et al.*, 1998] C. Cayrol, M. Lagasquie-Schiex, and Th. Schiex. Nonmonotonic reasoning: from complexity to algorithms. *Annals of Mathematics and Artificial Intelligence*, 22(3-4):207–236, 1998.
- [Eiter and Gottlob, 1993] Th. Eiter and G. Gottlob. The complexity of nested counterfactuals and iterated knowledge base revisions. In *IJCAI’93*, pages 526–531, 1993.
- [Etherington, 1987] D.V. Etherington. Relating default logic and circumscription. In *IJCAI’87*, pages 489–494, 1987.
- [Fagin *et al.*, 1983] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. In *PODS’83*, pages 352–365, 1983.
- [Jaśkowski, 1969] S. Jaśkowski. Propositional calculus for contradictory deductive systems. *Studia Logica*, 24:143–167, 1969.
- [Konieczny *et al.*, 2002] S. Konieczny, J. Lang, and P. Marquis. Distance-based merging: a general framework and some complexity results. In *KR’02*, pages 97–108, 2002.
- [Mengin, 2004] J. Mengin. Logical connectives for non-monotonicity: A choice function-based approach. In *JELIA’2004*, pages 452–461, 2004.
- [Nebel, 1991] B. Nebel. Belief revision and default reasoning: Syntax-based approaches. In *KR’91*, pages 417–428, 1991.
- [Nebel, 1998] B. Nebel. *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, chapter How hard is it to revise a knowledge base? Kluwer Academics, 1998.
- [Poole, 1988] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.
- [Rescher and Manor, 1970] N. Rescher and R. Manor. On inference from inconsistent premises. *Theory and Decision*, 1:179–219, 1970.
- [Restall, 1999] Greg Restall. *An Introduction to Substructural Logics*. Routledge, 1999.

Semantics for a useful fragment of the situation calculus

Gerhard Lakemeyer

Dept. of Computer Science
RWTH Aachen
52056 Aachen
Germany
gerhard@cs.rwth-aachen.de

Hector J. Levesque

Dept. of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S 3A6
hector@cs.toronto.edu

Abstract

In a recent paper, we presented a new logic called $\mathcal{ES}$ for reasoning about the knowledge, action, and perception of an agent. Although formulated using modal operators, we argued that the language was in fact a dialect of the situation calculus but with the situation terms suppressed. This allowed us to develop a clean and workable semantics for the language without piggybacking on the generic Tarski semantics for first-order logic. In this paper, we reconsider the relation between $\mathcal{ES}$ and the situation calculus and show how to map sentences of $\mathcal{ES}$ into the situation calculus. We argue that the fragment of the situation calculus represented by $\mathcal{ES}$ is rich enough to handle the basic action theories defined by Reiter as well as Golog. Finally, we show that in the full second-order version of $\mathcal{ES}$, almost all of the situation calculus can be accommodated.

1 Introduction

In a recent paper, we [Lakemeyer and Levesque, 2004] presented a new logic called $\mathcal{ES}$ for reasoning about the knowledge, action, and perception of an agent. Our main justification for introducing yet another knowledge representation formalism was twofold. First, we claimed that the language was not a new formalism at all, but merely a situation-suppressed version of part of the situation calculus, as originally presented by McCarthy and Hayes [McCarthy and Hayes 1969] and subsequently formalized by Reiter [Reiter 2001a]. Second, we claimed that because the language was defined semantically rather than axiomatically, certain mathematical arguments about the formalism were considerably simpler than in the original situation calculus. We presented evidence for this by showing a very compact proof of the correctness of Reiter's regression operator, and another involving a property called the determinacy of knowledge.

However, our first claim, which was that $\mathcal{ES}$ is in fact just an alternate way of writing some formulas of the situation calculus, was left unsubstantiated. In this paper, we remedy this. The main result we prove is the correctness of a simple mapping between sentences of $\mathcal{ES}$ and their counterparts in the classical language of the situation calculus. In addition, we show that the fragment of the situation calculus expressed

by $\mathcal{ES}$ is rich enough to handle (among other things) basic action theories (as defined by Reiter) and the *Do* operator which is the basis of the Golog language [Levesque et al. 1997]. We also illustrate this richness more informally using an example involving the knowledge, action, and sensing of a simple robot. However, there are sentences of the situation calculus that cannot be expressed directly in $\mathcal{ES}$, and which may be useful in some contexts. In the final part of the paper, we show that if we are prepared to use second-order quantification, there is a way to encode (almost) every sentence of the situation calculus in $\mathcal{ES}$. This encoding is a bit of a trick, and not something we advocate for everyday use. But it does exist. So we gain the clarity that comes from having a real semantic basis, and the ability to present concise semantic arguments, without any significant loss of expressiveness.

The rest of the paper is organized as follows. In the next section, we define the full language of $\mathcal{ES}$, its syntax and semantics. This generalizes the version of $\mathcal{ES}$ presented in [Lakemeyer and Levesque, 2004] in a variety of ways, the most important of which is that it allows functions, predicates, and second-order variables that are *fluent* (vary from situation to situation) as well as *rigid* (fixed for all situations). In Section 3, we consider the mapping from $\mathcal{ES}$ to the situation calculus and prove that a sentence of $\mathcal{ES}$ is valid iff its mapping is a logical entailment of a suitable situation calculus theory. In Section 4, we consider the expressiveness of $\mathcal{ES}$, first with an example basic action theory and then with *Do*. In Section 5, we consider the inverse mapping from the situation calculus to $\mathcal{ES}$, followed by some remarks on related work and conclusions. Then we stop.

2 The language: syntax and semantics

The full language $\mathcal{ES}$ consists of formulas over symbols from the following vocabulary:

- first-order variables: $x_1, x_2, \dots, y_1, y_2, \dots, a_1, a_2, \dots$;
- fluent second-order variables of arity k : $P_1^k, P_2^k, \dots$;
- rigid second-order variables of arity k : $Q_1^k, Q_2^k, \dots$;
- standard names: $n_1, n_2, \dots$;
- fluent function symbols of arity k : $f_1^k, f_2^k, \dots$;
for example, *location*, *bestAction*;
- rigid function symbols of arity k : $g_1^k, g_2^k, \dots$;
for example, *block5*, *repair*;

- fluent predicate symbols of arity k : $F_1^k, F_2^k, \dots$; for example, *Broken*;¹
- rigid predicate symbols of arity k : $G_1^k, G_2^k, \dots$; for example, *Fragile*;
- connectives and other symbols: $=, \wedge, \neg, \forall, \text{Know}, \square$, round and square parentheses, period, comma.

We assume that first-order variables, standard names, and function symbols come in two sorts, *action* (like *repair* and *bestAction*) and *object* (like *block5* and *location*). Constants are function symbols of 0 arity.² We let $\mathcal{N}$ denote the set of all standard names and $\mathcal{Z}$ denote the set of all sequences of standard names for actions, including $\langle \rangle$, the empty sequence.

2.1 Terms and formulas

The *terms* of the language are of sort *action* or *object*, and form the least set of expressions such that

1. Every standard name and first-order variable is a term of the corresponding sort;
2. If $t_1, \dots, t_k$ are terms and h is a k -ary function symbol then $h(t_1, \dots, t_k)$ is a term of the same sort as h .

By a *primitive term* we mean one of the form $h(n_1, \dots, n_k)$ where h is a (fluent or rigid) function symbol and all of the n_i are standard names.

The *well-formed formulas* of the language form the least set such that

1. If $t_1, \dots, t_k$ are terms, and H is a k -ary predicate symbol then $H(t_1, \dots, t_k)$ is an (atomic) formula;
2. If $t_1, \dots, t_k$ are terms, and V is a k -ary second-order variable, then $V(t_1, \dots, t_k)$ is an (atomic) formula;
3. If t_1 and t_2 are terms, then $(t_1 = t_2)$ is a formula;
4. If t is an action term and α is a formula, then $[t]\alpha$ is a formula;
5. If α and β are formulas, v is a first-order variable, and V is a second-order variable, then the following are also formulas: $(\alpha \wedge \beta)$, $\neg\alpha$, $\forall v. \alpha$, $\forall V. \alpha$, $\square\alpha$, $\text{Know}(\alpha)$.

We read $[t]\alpha$ as “ α holds after action t ”, and $\square\alpha$ as “ α holds after any sequence of actions.” As usual, we treat $(\alpha \vee \beta)$, $(\alpha \supset \beta)$, $(\alpha \equiv \beta)$, $\exists v. \alpha$, and $\exists V. \alpha$ as abbreviations. We call a formula without free variables a *sentence*. By a *primitive sentence* we mean a formula of the form $H(n_1, \dots, n_k)$ where H is a (fluent or rigid) predicate symbol and all of the n_i are standard names.

2.2 The semantics

The language contains both fluent and rigid expressions. The former vary as the result of actions and have values that may be unknown, but the latter do not. Intuitively, to determine whether or not a sentence α is true after a sequence of actions z has been performed, we need to specify two things: a world w and an epistemic state e . We write $e, w, z \models \alpha$. A

¹We assume this list includes the predicates *Poss* and *SF*.

²The standard names can be thought of as special extra constants that satisfy the unique name assumption and an infinitary version of domain closure.

world determines truth values for the primitive sentences and co-referring standard names for the primitive terms after any sequence of actions. An epistemic state is defined by a set of worlds, as in possible-world semantics. More precisely:

- a world $w \in W$ is any function from the primitive sentences and $\mathcal{Z}$ to $\{0, 1\}$, and from the primitive terms and $\mathcal{Z}$ to $\mathcal{N}$ (preserving sorts), and satisfying the rigidity constraint: if r is a rigid function or predicate symbol, then $w[r(n_1, \dots, n_k), z] = w[r(n_1, \dots, n_k), z']$, for all z and z' in $\mathcal{Z}$.
- an epistemic state $e \subseteq W$ is any set of worlds.

We extend the idea of co-referring standard names to arbitrary ground terms as follows. Given a term t without variables, a world w , and an action sequence z , we define $|t|_w^z$ (read: the co-referring standard name for t given w and z) by:

1. If $t \in \mathcal{N}$, then $|t|_w^z = t$;
2. $|h(t_1, \dots, t_k)|_w^z = w[h(n_1, \dots, n_k), z]$, where $n_i = |t_i|_w^z$.

To interpret formulas with free variables, we proceed as follows. First-order variables are handled substitutionally using the standard names. To handle the quantification over second-order variables, we use second-order *variable maps* defined as follows:

The *second-order primitives* are formulas of the form $V(n_1, \dots, n_k)$ where V is a (fluent or rigid) second-order variable and all of the n_i are standard names. A *variable map* u is a function from worlds, second-order primitives, and $\mathcal{Z}$ to $\{0, 1\}$, satisfying the rigidity constraint: if Q is a rigid second-order variable, then for all w and w' in W , and all z and z' in $\mathcal{Z}$, $u[w, Q(n_1, \dots, n_k), z] = u[w', Q(n_1, \dots, n_k), z']$.

Let u and u' be variable maps, and let V be a (fluent or rigid) second-order variable; we write $u' \sim_V u$ to mean that u and u' agree except perhaps on the second-order primitives involving V . Finally, to interpret what is known after a sequence of actions possibly including sensing has taken place, we define $w' \simeq_z w$ (read: w' and w agree on the sensing throughout action sequence z) inductively by the following:

1. $w' \simeq_{\langle \rangle} w$ iff w' and w agree on the value of every primitive rigid term and sentence;
2. $w' \simeq_{z \cdot n} w$ iff $w' \simeq_z w$ and $w'[SF(n), z] = w[SF(n), z]$.

Putting all these together, here is the semantic definition of truth. Given $e \subseteq W$ and $w \in W$, we define $e, w \models \alpha$ (read: α is true) as $e, w, \langle \rangle \models \alpha$, where for any $z \in \mathcal{Z}$ and any second-order variable map u :

1. $e, w, z, u \models H(t_1, \dots, t_k)$ iff $w[H(n_1, \dots, n_k), z] = 1$, where $n_i = |t_i|_w^z$;
2. $e, w, z, u \models V(t_1, \dots, t_k)$ iff $u[w, V(n_1, \dots, n_k), z] = 1$, where $n_i = |t_i|_w^z$;
3. $e, w, z, u \models (t_1 = t_2)$ iff n_1 and n_2 are identical, where $n_i = |t_i|_w^z$;
4. $e, w, z, u \models [t]\alpha$ iff $e, w, z \cdot n, u \models \alpha$, where $n = |t|_w^z$;

5. $e, w, z, u \models (\alpha \wedge \beta)$ iff
 $e, w, z, u \models \alpha$ and $e, w, z, u \models \beta$;
6. $e, w, z, u \models \neg \alpha$ iff $e, w, z, u \not\models \alpha$;
7. $e, w, z, u \models \forall v. \alpha$ iff $e, w, z, u \models \alpha_n^x$,
for every standard name n (of the same sort as v);
8. $e, w, z, u \models \forall V. \alpha$ iff
 $e, w, z, u' \models \alpha$, for every $u' \sim_V u$;
9. $e, w, z, u \models \Box \alpha$ iff
 $e, w, z \cdot z', u \models \alpha$, for every $z' \in \mathcal{Z}$;
10. $e, w, z, u \models \text{Know}(\alpha)$ iff
 $e, w', z, u \models \alpha$, for every $w' \in e$ such that $w' \simeq_z w$.

When α is *objective* (has no *Know* operators), we can leave out the e and write $w \models \alpha$. When Σ is a set of sentences and α is a sentence, we write $\Sigma \models \alpha$ (read: Σ logically entails α) to mean that for every e and w , if $e, w \models \alpha'$ for every $\alpha' \in \Sigma$, then $e, w \models \alpha$. Finally, we write $\models \alpha$ (read: α is valid) to mean $\{\} \models \alpha$.

3 Mapping to the situation calculus

How do we know that the semantics of $\mathcal{ES}$ is correct? In this section, we argue that it is indeed correct by showing how formulas α in $\mathcal{ES}$ can be translated in a direct way to formulas α^* in the situation calculus as defined by Reiter.³

The most desirable and simplest outcome of this translation would be that $\models \alpha$ iff $\Sigma \models_{\text{FOL}} \alpha^*$, where $\models$ is validity in $\mathcal{ES}$, Σ is the set of foundational axioms of the situation calculus (see [Levesque *et al.*, 1998], for example), and $\models_{\text{FOL}}$ is ordinary classical logical consequence. Unfortunately, we do not get exactly this correspondence for a variety of reasons we will discuss below. But we do get something close:

$$\models \alpha \text{ iff } \Sigma \cup \Upsilon \models_{\text{FOL}} \alpha^*,$$

where Υ is a set of four axioms that we will justify separately.

To prove this result it will be necessary to work with ordinary Tarski models of sentences of the situation calculus. As argued in [Lakemeyer and Levesque, 2004], this is difficult and painstaking, and is indeed one of the main reasons to prefer $\mathcal{ES}$ over the situation calculus. So while the proof of the theorem is quite laborious, we remind the reader that this can be thought of as a final reckoning for a formalism that is unworkable semantically. For space reasons, we do not review the conventional notation used for talking about Tarski interpretations, denotations, extensions, variable maps. See, for example, [Enderton 1972].

3.1 Definition of the translation

Before describing Υ , we present the translation from $\mathcal{ES}$ into the situation calculus. In the simplest case, the idea is that a formula like *Broken*(c), where *Broken* is a fluent, will

³We assume that this language has functional and relational fluents, functions and predicates that are not fluents, the distinguished constant S_0 , function *do*, predicates $\sqsubseteq$, *Poss* and *SF*, and a two-place predicate *K* for knowledge. We take *Knows*(α, σ) in the situation calculus as an abbreviation for the formula $\forall s(K(s, \sigma) \supset \alpha_s^{\text{now}})$, where α_s^{now} is the result of replacing by s in α every occurrence of *now* that is not within the scope of a further *Knows*.

be mapped to the situation calculus formula *Broken*(c, S_0), where we have restored the distinguished situation term S_0 for the fluent. Similarly, $[\text{repair}(c)]\neg\text{Broken}(c)$ will be mapped to $\neg\text{Broken}(c, \text{do}(\text{repair}(c), S_0))$. So $\mathcal{ES}$ formulas can be thought of as “situation-suppressed” (in situation-calculus terminology) and the $*$ mapping restores the situation argument to the fluents, leaving the rigids unchanged.

More precisely, we have the following:

Definition 3.1 Let α be any term or formula of $\mathcal{ES}$ without standard names. The expression α^* is defined as $\alpha[S_0]$ where, for any situation term σ , $\alpha[\sigma]$ is defined inductively by:

1. $v[\sigma]$, where v is a first-order variable, is v ;
2. $g(t_1, \dots, t_k)[\sigma]$, where g is a rigid function, predicate, or second-order variable, is $g(t_1[\sigma], \dots, t_k[\sigma])$;
3. $f(t_1, \dots, t_k)[\sigma]$, where f is a fluent function, predicate, or second-order variable is $f(t_1[\sigma], \dots, t_k[\sigma], \sigma)$;
4. $(t_1 = t_2)[\sigma]$ is $(t_1[\sigma] = t_2[\sigma])$;
5. $([t]\alpha)[\sigma]$ is $\alpha[\text{do}(t[\sigma], \sigma)]$;
6. $(\alpha \wedge \beta)[\sigma]$ is $(\alpha[\sigma] \wedge \beta[\sigma])$;
7. $(\neg \alpha)[\sigma]$ is $\neg \alpha[\sigma]$;
8. $(\forall v. \alpha)[\sigma]$ is $\forall v. \alpha[\sigma]$;
9. $(\forall V. \alpha)[\sigma]$ is $\forall V. \alpha[\sigma]$;
10. $(\Box \alpha)[\sigma]$ is $\forall s'(\sigma \sqsubseteq s' \supset \alpha[s'])$;
11. $\text{Know}(\alpha)[\sigma]$ is $\text{Knows}(\alpha[\text{now}], \sigma)$.

Note that the translation of $\Box \alpha$ introduces quantification over situations, where the introduced variable s' is assumed to be one that does not appear in situation term σ .

3.2 The situation-calculus axioms Υ

The axioms we assume in Υ are the following:

1. domain of objects is countably infinite;⁴
2. domain of actions is countably infinite (as above);
3. equality is the identity relation:
 $\forall x \forall y. (x = y) \equiv \forall Q(Q(x) \equiv Q(y))$.
4. the *K* predicate:⁵ $\forall s(K(s, S_0) \supset \text{Ini}(s)) \wedge$
 $\forall s' \forall s. K(s', s) \equiv \forall P(\dots \supset P(s', s))$,
where the ellipsis stands for the universal closure of
 $[K(s_1, S_0) \wedge \text{Ini}(s_2) \supset P(s_1, s_2)] \wedge$
 $[P(s_1, s_2) \wedge \text{SF}(a, s_1) \equiv \text{SF}(a, s_2) \supset$
 $P(\text{do}(a, s_1), \text{do}(a, s_2))]$.

Axioms (1) and (2) talk about the cardinality of the set of objects and actions respectively: they are both countable and infinite. The countability aspect is not very controversial. In the first-order case, every satisfiable set of sentences is satisfiable in a countable domain, and we do not expect users of the situation calculus to use second-order logic to defeat this. Note that this does not rule out having theories that talk about

⁴For space reasons, we omit the formula of second-order logic that fixes the cardinality of the domain of objects and actions.

⁵We let *Ini*(t) be an abbreviation for the situation calculus formula $\forall a \forall s(t \neq \text{do}(a, s))$. In this version of the axiom, we ignore the correspondence between *K* and *Poss*.

real numbers or other continuous phenomena; it simply rules out using second-order logic to force the interpretations of these theories to be uncountable. We can, however, imagine contexts where finiteness might be desirable. In such cases, we can introduce a new predicate O and instead of asserting that there are finitely many objects, assert that there are finitely many objects in O .

As for axiom (3), it is hard imagining anyone taking the negation of this one seriously. The usual first-order axiomatization of equality is often enough, but the intent is invariably for the equality symbol to be understood as the identity relation, which this second-order axiom ensures.

Finally axiom (4) is a second order definition of the K predicate in terms of the value it has at S_0 . This is just another way of capturing the successor state axiom for K introduced by Scherl and Levesque [2003], and the added machinery to make $Knows$ be a weak-S5 operator [Hughes and Cresswell, 1968]. Other knowledge operators are possible in the situation calculus, but weak-S5 and its extensions (such as strong-S5) are the most often used.

3.3 The embedding theorem

What we show is that provided certain properties hold between a Tarski structure on one side and an epistemic state, and a collection of worlds on the other, sentences will be true in $\mathcal{ES}$ iff their translations are true in the Tarski structure.

Let $\mathcal{M}$ be a Tarski structure for the situation calculus over the domain $\mathcal{D} = \mathcal{D}_{sit} \cup \mathcal{D}_{act} \cup \mathcal{D}_{obj}$, with $\mathcal{D}_{ini} \subseteq \mathcal{D}_{sit}$ as the set of initial situations, and with $\iota_0 = S_0^{\mathcal{M}} \in \mathcal{D}_{ini}$. Let e be an epistemic state, and assume that we are given three mappings $\omega \in [\mathcal{D}_{ini} \rightarrow W]$, $\theta \in [\mathcal{N} \rightarrow \mathcal{D}_{act} \cup \mathcal{D}_{obj}]$, and $\pi \in [\mathcal{Z} \times \mathcal{D}_{ini} \rightarrow \mathcal{D}_{sit}]$. These mappings need to satisfy various consistency properties that we cannot enumerate here. But we can at least quote the lemma that uses them:

Lemma 3.2 *Let α be a formula of $\mathcal{ES}$ with no standard names and whose free variables are among $x_1, \dots, x_m$. Then, given the consistency properties, for any variable map μ , the $\mathcal{ES}$ variable map u_μ (defined in terms of μ), any variable s , any $z \in \mathcal{Z}$, any $\iota \in \mathcal{D}_{ini}$, and $w = \omega(\iota)$,*

$$e, w, z, u_\mu \models \alpha_{n_1 \dots n_m}^{x_1 \dots x_m} \text{ iff}$$

$$M, \mu\{x_1/\theta(n_1), \dots, x_m/\theta(n_m), s/\pi(z, \iota)\} \models \alpha[s].$$

This is proved by a (long) induction on the structure of α . The main result is then the following:

Theorem 3.3 *Let α be any sentence of $\mathcal{ES}$ without standard names. Then α is valid iff $\Sigma \cup \Upsilon \models_{\text{FOL}} \alpha^*$.*

Proof: (Sketch) First assume that α is not valid. Then there is an e, w_0 such that $e, w_0 \not\models \alpha$. We define a Tarski structure $\mathcal{M}$ whose domain is $\mathcal{D} = \mathcal{D}_{obj} \cup \mathcal{D}_{act} \cup \mathcal{D}_{sit}$, where $\mathcal{D}_{obj}$ (resp. $\mathcal{D}_{act}$) is the set of standard names of objects (resp. actions), and $\mathcal{D}_{sit} = \mathcal{Z} \times W$. Then we define $=^{\mathcal{M}}$, $S_0^{\mathcal{M}}$, $do^{\mathcal{M}}$, $\sqsubseteq^{\mathcal{M}}$, $K^{\mathcal{M}}$, as well as the extension of every rigid and fluent predicate and function symbol using e and w_0 in a way that ensures that $\mathcal{M}$ satisfies $\Sigma \cup \Upsilon$. Next, define the mappings θ , π , and ω by letting $\theta(n) = n$, and for any initial $\iota = (\langle \rangle, w)$, letting $\pi(z, \iota) = (z, w)$, and $\omega(\iota) = w$. This ensures that the

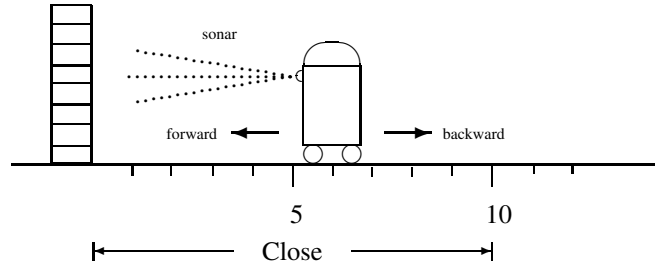


Figure 1: A simple robot

properties needed for Lemma 3.2 are satisfied, and therefore $\mathcal{M} \not\models \alpha^*$. Consequently, $\Sigma \cup \Upsilon \not\models_{\text{FOL}} \alpha^*$.

Conversely, assume that $\Sigma \cup \Upsilon \not\models_{\text{FOL}} \alpha^*$. Then there is a Tarski structure $\mathcal{M}$ that satisfies $\Sigma \cup \Upsilon$ but such that $\mathcal{M} \not\models \alpha^*$. The domain $\mathcal{D}$ must be $\mathcal{D}_{sit} \cup \mathcal{D}_{act} \cup \mathcal{D}_{obj}$, with $\mathcal{D}_{ini} \subseteq \mathcal{D}_{sit}$ as the set of initial situations, and with $\iota_0 = S_0^{\mathcal{M}} \in \mathcal{D}_{ini}$. Since $\mathcal{M} \models \Upsilon$, both $\mathcal{D}_{obj}$ and $\mathcal{D}_{act}$ are countably infinite, $\mathcal{D}_{obj} = \{\delta_1, \delta_2, \dots\}$, and $\mathcal{D}_{act} = \{\lambda_1, \lambda_2, \dots\}$. We define θ to map the i -th standard name for objects to δ_i and analogously for actions. We then define π using $do^{\mathcal{M}}$, and ω using the extensions of the function and predicate symbols given by $\mathcal{M}$. Finally, we let $e = \{\omega(\iota) \mid (\iota, \iota_0) \in K^{\mathcal{M}}\}$, and $w_0 = \omega(\iota_0)$. These ensure that the properties needed for Lemma 3.2 are satisfied, and so $e, w_0 \not\models \alpha$. Consequently, α is not valid. ■

4 The expressiveness of the language

Now that we have established that $\mathcal{ES}$ is actually a disguised fragment of the situation calculus, we next consider the expressiveness of this fragment, starting with a simple example problem, adapted from [Levesque and Lakemeyer 2001].

Imagine a robot that lives in a 1-dimensional world, and that can move towards or away from a fixed wall. The robot also has a sonar sensor that tells it when it gets close to the wall, say, less than 10 units away. See Figure 1. So we might imagine three actions, *forward* and *backward* which move the robot one unit towards and away from the wall, and a *sonar* sensing action which tells the robot if it is close to the wall. Each of these is a rigid constant, but for simplicity, we will simply assume that they are standard names. We have a single fluent, *distance*, which gives the actual distance from the robot to the wall. We can use *Close* as an abbreviation for the formula “*distance* < 10.”⁶ We begin our formalization by writing preconditions for the three actions:

$$\begin{aligned} \forall a \Box \text{Poss}(a) &\equiv \\ a = \text{forward} &\wedge \text{distance} > 0 \quad \vee \\ a = \text{backward} &\wedge \text{TRUE} \quad \vee \\ a = \text{sonar} &\wedge \text{TRUE}. \end{aligned}$$

Next, we define the sensing results for the actions:

$$\begin{aligned} \forall a \Box \text{SF}(a) &\equiv \\ a = \text{forward} &\wedge \text{TRUE} \quad \vee \\ a = \text{backward} &\wedge \text{TRUE} \quad \vee \end{aligned}$$

⁶Here and below, we use simple arithmetic involving $<$, $+$, and $-$, which can easily be defined in second-order terms with the standard names acting as natural numbers. We omit the details.

$$a = \text{sonar} \wedge \text{Close}.$$

Finally, we write a successor state axiom for our only fluent:

$$\begin{aligned} \forall a, x \square [a](\text{distance} = x) \equiv \\ a = \text{forward} \wedge \text{distance} = x + 1 \quad \vee \\ a = \text{backward} \wedge \text{distance} = x - 1 \quad \vee \\ a \neq \text{forward} \wedge a \neq \text{backward} \wedge \text{distance} = x. \end{aligned}$$

Now we are ready to consider some specifics having to do with what is true initially by defining an action theory Σ . Let ϕ denote the conjunction of the sentences above. We assume that ϕ is true and the robot knows it. We also assume the robot is located initially 6 units away from the wall, but that the robot has no idea where it is. So, we let Σ be

$$\{\phi, \text{Know}(\phi), \text{distance} = 6, \forall x \neg \text{Know}(\text{distance} \neq x)\},$$

and we get this:

Theorem 4.1 *The following are logical entailments of Σ :*

1. $\text{Close} \wedge \neg \text{Know}(\text{Close}) \wedge [\text{forward}] \neg \text{Know}(\text{Close})$
the robot is close to the wall, but does not know it, and continues not to know it after moving forward;
2. $[\text{sonar}] (\text{Know}(\text{Close}) \wedge [\text{forward}] \text{Know}(\text{Close}))$
after reading the sonar, the robot knows it is close, and continues to know it after moving forward;
3. $[\text{sonar}] [\text{backward}] \neg \text{Know}(\text{Close})$
after reading the sonar and then moving backward, the robot no longer knows that it is close to the wall;
4. $[\text{backward}] [\text{sonar}] \text{Know}(\text{Close})$
after moving backward and then reading the sonar, the robot knows that it is close to the wall;
5. $[\text{sonar}] [\text{forward}] [\text{backward}] \text{Know}(\text{Close})$
after reading the sonar, moving forward, and then backward, the robot knows that it is still close to the wall;
6. $[\text{sonar}] \text{Know}([\text{forward}] \text{Close})$
after reading the sonar, the robot knows that it will remain close after moving forward;
7. $\neg \text{Know}([\text{sonar}] \text{Know}(\text{Close}))$
the robot does not know initially that it will know that it is close after reading the sonar;
8. $\text{Know}([\text{sonar}] (\text{Know}(\text{Close}) \vee \text{Know}(\neg \text{Close})))$
the robot does know initially that after reading the sonar, it will then know whether or not it is close to the wall;
9. $\text{Know}([\text{sonar}] [\text{backward}] \neg \text{Know}(\text{Close}))$
the robot knows initially that it will not know that it is close after reading the sonar and moving backwards.

Proof: The proofs of these are similar. Here we will only do item 3. Let $z = \langle \text{sonar} \cdot \text{backward} \rangle$, and suppose that $e, w \models \Sigma$; we must show that $e, w, z \models \neg \text{Know}(\text{Close})$. Because $e, w \models \neg \text{Know}(\text{distance} \neq 9)$, there exists $w' \in e$ such that $w' \simeq_{\langle \rangle} w$ and $w'[\text{distance}, \langle \rangle] = 9$. Since $9 < 10$, we also have that $w' \simeq_z w$. However, $w'[\text{distance}, z] = 10$. So there exists $w' \in e$ such that $w' \simeq_z w$ and $w', z \models \neg \text{Close}$. Therefore, $e, w, z \models \neg \text{Know}(\text{Close})$. ■

Although we will not attempt to formulate a theorem here, it should be clear from this example that any basic action theory [Reiter 2001a] including those involving the Scherl-Levesque knowledge operator can be expressed in $\mathcal{ES}$.

4.1 The Do operator

We now turn our attention to the *Do* operator which is the basis of the Golog language [Levesque et al. 1997], and show that a variant of *Do* can be represented in $\mathcal{ES}$. We cannot encode $Do(\delta, s, s')$ directly since we do not have situations as terms. Instead we will use $D\tilde{o}(\delta, \alpha)$ which intuitively means that α holds after doing Golog program δ . There are two possible readings: one says that α holds in *all* final states, and the other says that α holds in *some* final state. They can be inter-defined, so we consider only the latter. We treat $D\tilde{o}(\delta, \alpha)$ as an abbreviation for a formula of $\mathcal{ES}$, defined recursively on the δ as follows:

1. for any action a , $D\tilde{o}(a, \alpha) \stackrel{\text{def}}{=} (\text{Poss}(a) \wedge [a]\alpha)$;
2. $D\tilde{o}(\phi?, \alpha) \stackrel{\text{def}}{=} (\phi \wedge \alpha)$;
3. $D\tilde{o}(\delta; \delta', \alpha) \stackrel{\text{def}}{=} D\tilde{o}(\delta, D\tilde{o}(\delta', \alpha))$;
4. $D\tilde{o}(\delta \mid \delta', \alpha) \stackrel{\text{def}}{=} (D\tilde{o}(\delta, \alpha) \vee D\tilde{o}(\delta', \alpha))$;
5. $D\tilde{o}(\pi x. \delta, \alpha) \stackrel{\text{def}}{=} \exists x. D\tilde{o}(\delta, \alpha)$;
6. $D\tilde{o}(\delta^*, \alpha) \stackrel{\text{def}}{=} \forall P. \{\square(\alpha \supset P) \wedge \square(D\tilde{o}(\delta, P) \supset P)\} \supset P$.

As usual, we can define while-loops and if-then-else as abbreviations. The main theorem that we state here is that this account of $D\tilde{o}$ is correct relative to the original account of *Do* and the mapping from $\mathcal{ES}$ defined above:

Theorem 4.2 *Let δ be any program, α be any sentence of $\mathcal{ES}$, and σ be any situation term of the situation calculus. Then $\models_{\text{FOL}} D\tilde{o}(\delta, \alpha)[\sigma] \equiv \exists s. Do(\delta, \sigma, s) \wedge \alpha[s]$.*

This is proved by induction over δ , the only troublesome case being for the nondeterministic iteration, δ^* .

5 Mapping from the situation calculus

When mapping the situation calculus into $\mathcal{ES}$, the main issue is the treatment of quantified situations. While simple formulas can be translated using the inverse of $*$ from Section 3, a sentence like $\forall s \exists s'. s \sqsubseteq s' \wedge (s \neq s') \wedge F(s) \equiv F(s')$, which says that from every situation another situation is reachable that agrees on the truth value of F , has no counterpart in $\mathcal{ES}$.

To deal with situation-calculus sentences like these, our proposal is to encode *action sequences* in second-order $\mathcal{ES}$. Let *nil* be a (rigid) constant and *seq* a (rigid) binary function symbol of $\mathcal{ES}$. We define *ActionSeq*(x) as an abbreviation for

$$\forall Q. Q(\text{nil}) \wedge \forall y, a. (Q(a) \supset Q(\text{seq}(a, y))) \supset Q(x).$$

In the situation calculus, a fluent like $F(\text{do}(\alpha, \text{do}(\beta, S_0)))$ actually means that F holds after doing β then α . So the actions are in reverse order in a situation term. It is therefore useful to be able to reverse a sequence as follows:

Definition 5.1 $\text{Rev}(x, y) \stackrel{\text{def}}{=} \forall R. (\dots \supset R(x, \text{nil}, y))$, where the ellipsis stands for the universal closure of

$$R(\text{nil}, x, x) \wedge [R(x, \text{seq}(a, y), z) \supset R(\text{seq}(a, x), y, z)].$$

Next we define what it means for a formula α to be true after a sequence of actions.

Definition 5.2 $After(s, \alpha) \stackrel{def}{=} \forall P. (\dots \supset P(s))$ where the ellipsis stands for the universal closure of

$$\Box\{(\alpha \supset P(nil)) \wedge ([a]P(x) \wedge Rev(seq(a, x), y) \supset P(y))\}.$$

The fluent $\neg Broken(do(repair, do(drop, S_0)))$ will then be translated as $After(seq(repair, seq(drop, nil)), \neg Broken)$. In order to deal with quantification over situations, we define $\leq$ as a relation over action sequences, similar to the situation calculus relation $\sqsubseteq$.

Definition 5.3 $(x \leq x') \stackrel{def}{=} \forall Q. (\dots \supset Q(x, x'))$ where the ellipsis stands for the universal closure of

$$Q(nil, x) \wedge (Q(x, y) \supset Q(seq(a, x), seq(a, y))).$$

The formula $\forall s \exists s'. s \sqsubseteq s' \wedge (s \neq s') \wedge F(s) \equiv F(s')$ of the situation calculus will then be translated as

$$\forall x. ActionSeq(x) \supset \exists x'. ActionSeq(x') \wedge (x \leq x') \wedge (x \neq x') \wedge [After(x, F) \equiv After(x', F)].$$

The final problem we must deal with concerns unrestricted quantification over initial situations. To simplify things, we restrict ourselves to SC^r , the *rooted* situation calculus, where all situation quantification is of the form $\forall s. \sigma \sqsubseteq s \supset \alpha$, for some situation term σ . Note that this is no restriction at all for the non-epistemic situation calculus, since $\forall s. \alpha$ is equivalent to $\forall s. S_0 \sqsubseteq s \supset \alpha$. We handle the *Knows* operator separately.

To ease the translation, we assume that situation-calculus formulas are in the following normal form: all arguments of predicates and functions are variables; equality expressions have the form $(x = t)$, where x is a variable and t is a term mentioning at most 1 function symbol. It is easy to show that every formula of SC^r is equivalent to one in this normal form.

The mapping from the situation calculus into $\mathcal{ES}$ begins with a mapping $\bullet$ from situation terms into sequences of actions in $\mathcal{ES}$. We assume that there is a countably infinite set V_s of extra object variables in $\mathcal{ES}$ for this purpose. For any situation variable s , we assume that $s^\bullet \in V_s$ and that $\bullet$ is 1-to-1 and onto wrt V_s when restricted to situation variables. We also have that $S_0^\bullet = now^\bullet = nil$, and that $do(a, s)^\bullet = seq(a, s^\bullet)$. We extend $\bullet$ to formulas as follows:

Definition 5.4 Let α be any formula of SC^r in normal form. Then its translation into $\mathcal{ES}$, $\alpha^\bullet$, is defined inductively by:

1. $G(x_1, \dots, x_k)^\bullet$, where G is a rigid predicate or rigid second-order variable, is $G(x_1, \dots, x_k)$;
2. $F(x_1, \dots, x_k, s)^\bullet$, where F is a fluent predicate or fluent second-order variable, is $After(s^\bullet, F(x_1, \dots, x_k))$;
3. $(x = g(x_1, \dots, x_k))^\bullet$, where g is a rigid function, is $(x = g(x_1, \dots, x_k))$;
4. $(x = f(x_1, \dots, x_k, s))^\bullet$, where f is a fluent function, is $After(s^\bullet, x = f(x_1, \dots, x_k))$;
5. $(s = \sigma)^\bullet$ is $(s^\bullet = \sigma^\bullet)$;
6. $(\sigma_1 \sqsubseteq \sigma_2)^\bullet$ is $(\sigma_1^\bullet \leq \sigma_2^\bullet)$;
7. $(\neg \alpha)^\bullet$ is $\neg \alpha^\bullet$;
8. $(\alpha \wedge \beta)^\bullet$ is $\alpha^\bullet \wedge \beta^\bullet$;
9. $(\forall x. \alpha)^\bullet$ is $\forall x. \alpha^\bullet$ for an object or action variable x ;

$$10. (\forall s. \sigma \sqsubseteq s \supset \alpha)^\bullet \text{ is } \forall s^\bullet. ActionSeq(s^\bullet) \wedge \sigma^\bullet \leq s^\bullet \supset \alpha^\bullet;$$

$$11. (\forall V. \alpha)^\bullet \text{ is } \forall V. \alpha^\bullet \text{ for a second-order variable } V;$$

$$12. Knows(\alpha, s)^\bullet \text{ is } After(s^\bullet, Know(\alpha^\bullet)).$$

To show correctness of the translation, we need two more axioms Ψ , which are the universal closure of the following:

$$seq(a, x) \neq nil;$$

$$seq(a, x) = seq(a', x') \supset a = a' \wedge x = x'.$$

Theorem 5.5 Let α be a sentence of SC^r in normal form. Then $\Sigma \cup \Upsilon \models_{FOL} \alpha$ iff $\Psi \models \alpha^\bullet$.

6 Related Work

The situation calculus, which has been the sole focus of this paper, is of course not the only language for reasoning about action. Over the years, there have been many other proposals such as the event calculus [Kowalski and Sergot 1986], Sandewall's features and fluents [Sandewall 1994], the language $\mathcal{A}$ [Gelfond and Lifschitz 1993], or the fluent calculus [Hölldobler and Schneeberger 1990], to name but a few. What distinguishes the situation calculus perhaps most from its alternatives is that it admits an elegant *monotonic* solution to the frame problem [Reiter 1991], which is shared by its close relative, the fluent calculus [Thielscher 1999]. However, the fluent calculus also has situation terms as part of the language and hence suffers from the same shortcomings which $\mathcal{ES}$ tries to address.

$\mathcal{ES}$ itself has much in common with dynamic logic [Pratt 1976; Harel 1984], which has also been applied to reasoning about action [Castilho, Gasquet, and Herzig 1999]. A new feature of $\mathcal{ES}$, compared to dynamic logic, are quantified modalities as in $\forall a. [a]\alpha$, which are key in expressing successor state axioms, among other things.

Dynamic logic has been combined with epistemic logic [Herzig et al. 2000; Demolombe 2003]. In the language of [Herzig et al. 2000], it is possible to express things like *[backward] [sonar] Know(Close)* using an almost identical syntax and where *Know* also has a possible-world semantics. Despite such similarities, there are significant differences, however. In particular, their language is propositional. Consequently, there is no quantification over actions. Demolombe (2003) proposes an axiomatic translation of parts of the epistemic situation calculus into modal logic. While his modal language is first-order, he does not consider quantified modalities.

Although they do not consider epistemic notions, the work by [Blackburn et al. 2001] is relevant as it reconstructs a version of the situation calculus in Hybrid Logic [Blackburn et al. 2001], a variant of modal logic which was inspired by the work on tense logic by Prior [Prior 1967]. In a sense, though, this work goes only part of the way as an explicit reference to situations within the logic is retained. To us this presents a disadvantage when moving to an epistemic extension. The problem is that the epistemic situation calculus requires us to consider uncountably many situations, which precludes a substitutional interpretation of quantification.

7 Conclusion

In this paper, we have substantiated the claim made informally in a previous paper that $\mathcal{ES}$ is a fragment of the situation calculus that admits an intuitive possible-world semantics. We argued that this fragment is expressive enough to represent basic action theories by presenting a simple example involving the action, knowledge, and sensing of a robot. We also showed how a version of the *Do* operator of the Golog language could be accommodated within the language. Finally, we proved that by using second-order encodings of sequences, virtually all of the situation calculus can be represented in $\mathcal{ES}$. So what we get out of $\mathcal{ES}$ is a tool with practically the same expressive power as the standard situation calculus, but which is much more amenable to analysis due to its simple semantic basis.

In work currently under way [Claßen 2005], $\mathcal{ES}$ is used to simplify and extend results by Reiter on knowledge-based programs [Reiter 2001b]. There are also a number of directions for future research, but perhaps the most immediate would be to implement a new version of Golog based on $\mathcal{ES}$ and put it to work.

Acknowledgments

We thank the anonymous reviewers for their helpful comments.

References

- [Blackburn et al. 2001] P. Blackburn, J. Kamps, and M. Marx, Situation calculus as hybrid logic: First steps. In P. Brazdil and A. Jorge (Eds.) *Progress in Artificial Intelligence*, Lecture Notes in Artificial Intelligence 2258, Springer Verlag, 253–260, 2001.
- [Castilho, Gasquet, and Herzig 1999] M. A. Castilho, O. Gasquet, and A. Herzig, Formalizing Action and Change in Modal Logic I: the frame problem. *Journal of Logic and Computation*, 9(5), 701–735, 1999.
- [Claßen 2005] J. Claßen, *Knowledge-Based Programming in the Logic $\mathcal{ES}$* , MSc. Thesis, RWTH Aachen, in progress.
- [Demolombe 2003] R. Demolombe, Belief change: from Situation Calculus to Modal Logic. *IJCAI Workshop on Nonmonotonic Reasoning, Action, and Change (NRAC'03)*, Acapulco, Mexico, 2003.
- [Enderton 1972] H. Enderton, *A Mathematical Introduction to Logic*, Academic Press, New York, 1972.
- [Gelfond and Lifschitz 1993] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–321, 1993.
- [Harel 1984] D. Harel, Dynamic Logic, in D. Gabbay and F. Guenther (Eds.), *Handbook of Philosophical Logic*, Vol. 2, D. Reidel Publishing Company, 497–604, 1984.
- [Herzig et al. 2000] A. Herzig, J. Lang, D. Longin, and T. Polacsek, A logic for planning under partial observability. In *Proc. AAAI-2000*, AAAI Press.
- [Hölldobler and Schneeberger 1990] S. Hölldobler and J. Schneeberger. A new deductive approach to planning. *New Generation Computing*, 8:225–244, 1990.
- [Kowalski and Sergot 1986] R. Kowalski and M. Sergot. A logic based calculus of events. *New Generation Computing*, 4:67–95, 1986.
- [Hughes and Cresswell, 1968] G. Hughes, and M. Cresswell, *An Introduction to Modal Logic*, Methuen and Co., London, 1968.
- [Lakemeyer and Levesque, 2004] G. Lakemeyer and H. J. Levesque, Situations, si! Situation Terms, no!. In *Ninth Conf. on Principles of Knowledge Representation and Reasoning*, AAAI Press, 2004.
- [Levesque et al, 1998] H. J. Levesque, F. Pirri, and R. Reiter, Foundations for the situation calculus, *Electronic Transactions of AI (ETAI)*, 2(3-4):159–178, 1998.
- [Levesque and Lakemeyer 2001] H. J. Levesque and G. Lakemeyer, *The Logic of Knowledge Bases*, MIT Press, 2001.
- [Levesque et al. 1997] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl., Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31, 59–84, 1997.
- [McCarthy and Hayes 1969] J. McCarthy and P. J. Hayes, Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer, D. Mitchie and M. Swann (Eds.) *Machine Intelligence 4*, Edinburgh University Press, 463–502, 1969.
- [Pratt 1976] V. R. Pratt, Semantical considerations on Floyd-Hoare logic. In *Proc. 17th Ann. IEEE Symp. on Foundations of Computer Science*, IEEE Press, 109–121, 1976.
- [Prior 1967] A. Prior, *Past, Present and Future*. Oxford University Press, 1967.
- [Reiter 1991] Ray Reiter, The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, 359–380. Academic Press, 1991.
- [Reiter 2001a] R. Reiter, *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press, 2001.
- [Reiter 2001b] Ray Reiter, On knowledge-based programming with sensing in the situation calculus. *ACM Transactions on Computational Logic*, 2(4):433–457, 2001.
- [Sandewall 1994] Erik Sandewall. *Features and Fluents. The Representation of Knowledge about Dynamical Systems*. Oxford University Press, 1994.
- [Scherl and Levesque, 2003] R. B. Scherl and H. J. Levesque, Knowledge, action, and the frame problem. *Artificial Intelligence* 144(1-2), 1–39, 2003.
- [Thielscher 1999] Michael 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.