NRAC 2009

Proceedings of the IJCAI-09 Workshop on Nonmonotonic Reasoning, Action and Change

11 July 2009; Pasadena, California
Edited by Andreas Herzig and Benjamin Johnston

ISBN 978-0-9802840-7-2
11 July 2009

Published by UTSePress
University of Technology, Sydney
Broadway NSW 2007 Australia

Contents

Preface

The biennial workshop on Nonmonotonic Reasoning, Action and Change (NRAC) has an active and loyal community. Since its inception in 1995, the workshop has been held seven times in conjunction with IJCAI, and has experienced growing success. We hope to build on this success again this eighth year with an interesting and fruitful day of discussion.

The areas of reasoning about action, non-monotonic reasoning and belief revision are among the most active research areas in Knowledge Representation, with rich inter-connections and practical applications including robotics, agent-systems, commonsense reasoning and the semantic web. This workshop provides a unique opportunity for researchers from all three fields to be brought together at a single forum with the prime objectives of communicating important recent advances in each field and the exchange of ideas. As these fundamental areas mature it is vital that researchers maintain a dialog through which they can cooperatively explore common links. The goal of this workshop is to work against the natural tendency of such rapidly advancing fields to drift apart into isolated islands of specialization.

This year, we have accepted ten papers authored by a diverse international community. Each paper has been subject to careful peer review on the basis of innovation, significance and relevance to NRAC. The high quality selection of work could not have been achieved without the invaluable help of the international Program Committee.

A highlight of the workshop will be our invited speaker Professor Hector Geffner from ICREA and UPF in Barcelona, Spain, discussing representation and inference in modern planning. Hector Geffner is a world leader in planning, reasoning, and knowledge representation; in addition to his many important publications, he is a Fellow of the AAAI, an associate editor of the Journal of Artificial Intelligence Research and won an ACM Distinguished Dissertation Award in 1990.

Whether you are part of the loyal community or if this is your first time to attend, we hope that you will enjoy the scholarship and the fun at NRAC 2009.

Sincerely,

Andreas Herzig and Benjamin Johnston
Workshop Chairs

Workshop Organization

Workshop Chairs

Andreas Herzig	Paul Sabatier University (France)
Benjamin Johnston	University of Technology, Sydney (Australia)

Program Committee

Eyal Amir	University of Illinois (USA)
Gerhard Brewka	University of Leipzig (Germany)
Xiaoping Chen	University of Science and Technology of China (China)
Jim Delgrande	Simon Fraser University (Canada)
Patrick Doherty	Linköping Universtiy (Sweden)
Jérôme Lang	Universite Paul Sabatier (France)
Fangzhen Lin	Hong Kong University of Science and Technology (China)
Wei Liu	University of Western Australia (Australia)
Thomas Meyer	Meraka Institute (South Africa)
Leora Morgenstern	IBM Research (USA)
Maurice Pagnucco	University of NSW (Australia)
Pavlos Peppas	University of Patras (Greece)
Erik Sandewall	Linköping University (Sweden)
Michael Thielscher	Dresden University of Technology (Germany)
Mary-Anne Williams	University of Technology, Sydney (Australia)

Other Reviewers

Ji Jianmin	University of Science and Technology of China (China)
Meghyn Bienvenu	Paul Sabatier University (France)

Steering Committee

Gerhard Brewka	University of Leipzig (Germany)
Leora Morgenstern	IBM Research (USA)
Maurice Pagnucco	University of NSW (Australia)
Pavlos Peppas	University of Patras (Greece)
Michael Thielscher	Dresden University of Technology (Germany)
Mary-Anne Williams	University of Technology, Sydney (Australia)

Invited Talk:
Representation and Inference in Modern Planning: From Classical Plans to Finite-State Controllers

Hector Geffner
ICREA and UPF
Barcelona, Spain

Abstract

Planning is concerned with the development of solvers for a range of models where actions must be executed to achieve goals. In these models, actions may be deterministic or not, states may be observable or not, and so on. The challenge in all cases is computational: how to scale up to large problems even if the models are all intractable. In the last 10–15 years significant progress has been made in the area resulting from novel inference techniques and transformations. In this talk, I'll review these techniques, as we consider the computation of classical and conformant plans for unobservable problems, and finite-state controllers for partially observable ones.

This talk describes work performed jointly with Blai Bonet, Hector Palacios, colleagues and students.

Biography

Hector Geffner got his Ph.D in UCLA with a dissertation that was co-winner of the 1990 Association for Computing Machinery (ACM) Dissertation Award. Then he worked as Staff Research Member at the IBM T.J. Watson Research Center in NY, USA and at the Universidad Simon Bolivar, in Caracas, Venezuela. He is currently a researcher at the ICREA and a professor at the Universitat Pompeu Fabra. Hector Geffner is an Associate Editor of the Journal of Artificial Intelligence Research, a Fellow of the AAAI, and author of the book "Default Reasoning: Causal and Conditional Theories" published by MIT Press in 1992. He is interested in models of reasoning, action, planning, and learning.

Putting ABox Updates into Action

Conrad Drescher, Hongkai Liu, Franz Baader, Peter Steinke, Michael Thielscher
Department of Computer Science,
Dresden University of Technology
Nöthnitzer Str. 46, 01187 Dresden, Germany

Abstract

When trying to apply recently developed approaches for updating Description Logic ABoxes in the context of an action programming language, one encounters two problems. First, updates generate so-called *Boolean* ABoxes, which cannot be handled by traditional Description Logic reasoners. Second, iterated update operations result in *very large* Boolean ABoxes, which, however, contain a huge amount of redundant information. In this paper, we address both issues from a practical point of view.

1 Introduction

Agent programming languages such as Golog [Levesque *et al.*, 1997] and Flux [Thielscher, 2005] employ actions whose effects are defined in a logic-based calculus to describe and implement the behaviour of intelligent agents. In the so-called progression approach, the agent starts with a (possibly incomplete) description of the initial state of the world. When an action is performed, it updates this description to take into account the effects of this action. Reasoning about the description of the current state of the world is then, for example, used in the control structures of the agent program to decide which action to apply. The calculi underlying Golog and Flux (situation calculus and fluent calculus, respectively) employ full first-order predicate logic, which makes the computation of exact updates as well as the use of decision procedures for reasoning about descriptions of the state of the world impossible. To overcome this problem, recent papers [Baader *et al.*, 2005; Liu *et al.*, 2006] have proposed to employ a decidable Description Logic (DL) [Baader *et al.*, 2003] in place of full first-order predicate logic. In particular, states of the world are then described using a DL ABox. In [Liu *et al.*, 2006], a method for updating DL ABoxes has been developed, and in [Drescher and Thielscher, 2007] it was shown that this notion of an update conforms with the semantics employed by Golog and Flux.

In practice, however, there are two obstacles towards employing the update approach from [Liu *et al.*, 2006] in the context of agent programs. First, using the update procedures in the form described in [Liu *et al.*, 2006] quickly leads to unmanageably large ABoxes. However, there is quite some room for optimizations since the updated ABoxes contain a lot of redundant information. The second problem is that the updated ABoxes are so-called Boolean ABoxes, which cannot be directly handled by traditional DL reasoners. The main contributions of this paper are, on the one hand, that we propose and evaluate different optimization approaches for computing more concise updated ABoxes. On the other hand, we compare different approaches for reasoning with Boolean ABoxes, among them one based on the DPLL(T) approach.

The rest of this paper is organized as follows. In Section 2, we recall the basic notions for DLs and ABox updates. In Sections 3 we present optimizations that enable the construction of more concise updated ABoxes, and in Section 4 we discuss reasoning with Boolean ABoxes. In Section 5, the approaches introduced in the previous two sections are empirically evaluated.

2 Preliminaries

In DLs, knowledge is represented with the help of concepts (unary predicates) and roles (binary predicates). Complex concepts and roles are inductively defined starting with a set $\mathsf{N_C}$ of *concept names*, a set $\mathsf{N_R}$ of *role names*, and a set $\mathsf{N_I}$ of *individual names*. The expressiveness of a DL is determined by the set of available *constructors* to build *concepts* and *roles*. The concept and role constructors of the DLs $\mathcal{ALCO}^@$ and $\mathcal{ALCO}^+$ that form the base of our work on ABox update are shown in Table 1, where C, D are concepts, q, r are roles, and a, b are individual names. The DL that allows only for negation, conjunction, disjunction, and universal and existential restrictions is called $\mathcal{ALC}$. By adding nominals $\mathcal{O}$, we obtain $\mathcal{ALCO}$, which is extended to $\mathcal{ALCO}^@$ by the @-constructor from hybrid logic [Areces *et al.*, 1999], and to $\mathcal{ALCO}^+$ by the Boolean constructors on roles and the nominal role [Liu *et al.*, 2006]. We will use $\top$ ($\bot$) to denote arbitrary tautological (unsatisfiable) concepts and roles. By $\mathsf{sub}(\phi)$ we denote the set of all subconcepts and subroles of a concept or role ϕ, respectively.

The semantics of concepts and roles is defined via *interpretations* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$. The *domain* $\Delta^\mathcal{I}$ is a non-empty set and the *interpretation function* $\cdot^\mathcal{I}$ maps each concept name $A \in \mathsf{N_C}$ to a subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$, each role name $r \in \mathsf{N_R}$ to a binary relation $r^\mathcal{I}$ on $\Delta^\mathcal{I}$, and each individual name $a \in \mathsf{N_I}$ to an individual $a^\mathcal{I} \in \Delta^\mathcal{I}$. The interpretation function $\cdot^\mathcal{I}$ is

Name	Syntax	Semantics
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
univ. res.	$\forall r.C$	$\{x \mid \forall y.((x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}})\}$
exist. res.	$\exists r.C$	$\{x \mid \exists y.((x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
@ constructor	$@_a C$	$\Delta^{\mathcal{I}}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and $\emptyset$ otherwise
role negation	$\neg r$	$(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus r^{\mathcal{I}}$
role conjunction	$q \sqcap r$	$q^{\mathcal{I}} \cap r^{\mathcal{I}}$
role disjunction	$q \sqcup r$	$q^{\mathcal{I}} \cup r^{\mathcal{I}}$
nominal role	$\{(a,b)\}$	$\{(a^{\mathcal{I}}, b^{\mathcal{I}})\}$

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

inductively extended to complex concepts and roles as shown in Table 1.

An *ABox assertion* is of the form $C(a)$, $r(a,b)$, or $\neg r(a,b)$ with r a role, C a concept and a, b individual names. A *classical ABox*, or an *ABox* for short, is a finite conjunction of ABox assertions. A *Boolean ABox* is a Boolean combination of ABox assertions. For convenience we will also sometimes represent classical and Boolean ABoxes as finite sets of assertions by breaking the toplevel conjunctions. An interpretation $\mathcal{I}$ is a *model* of an assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$. $\mathcal{I}$ is a model of an assertion $r(a,b)$ (resp. $\neg r(a,b)$) if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}$). A model of a (Boolean) ABox is defined in the obvious way. We use $M(\mathcal{A})$ to denote the set of models of a Boolean ABox $\mathcal{A}$. A (Boolean) ABox $\mathcal{A}$ is *consistent* if $M(\mathcal{A}) \neq \emptyset$. Two (Boolean) ABoxes $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent*, denoted by $\mathcal{A} \equiv \mathcal{A}'$, if $M(\mathcal{A}) = M(\mathcal{A}')$. An assertion α is *entailed* by a Boolean ABox $\mathcal{A}$, written as $\mathcal{A} \models \alpha$, if $M(\mathcal{A}) \subseteq M(\{\alpha\})$. Classical $\mathcal{ALCO}^@$-ABoxes can equivalently be compiled to Boolean $\mathcal{ALCO}$-ABoxes (and vice versa) — the translation in the first direction is exponential, in the other direction it is linear [Liu *et al.*, 2006]. *Consistency checking* and *entailment* for classical ABoxes are standard inference problems and supported by all DL reasoners[1], while, to the best of our knowledge, no state of the art reasoner directly supports these inferences for Boolean ABoxes. Reasoning in $\mathcal{ALCO}^+$ is NExpTime complete [Tobies, 2001]; for $\mathcal{ALCO}^@$ it is PSpace complete [Areces *et al.*, 1999].

ABox Update

An ABox can be used to represent knowledge about the state of some world. An *update* contains information on changes that have taken place in that world.

Definition 2.1 (Update) *An* update $\mathcal{U} = \{\delta(\bar{t})\}$ *contains a single literal, i.e.* $\delta(\bar{t})$ *is of the form* $A(a)$, $\neg A(a)$, $r(a,b)$, *or* $\neg r(a,b)$ *with* A *a concept name,* r *a role name, and* a, b *individual names.*[2] ⊣

Intuitively, an update literal $\delta(\bar{t})$ says that this literal holds after the change of the world state. The formal semantics of updates given in [Liu *et al.*, 2006] defines, for every interpretation $\mathcal{I}$, a successor interpretation $\mathcal{I}^{\mathcal{U}}$ obtained by changing this model according to the update. This is the Winslett semantics first introduced in [Winslett, 1988]. Given an ABox $\mathcal{A}$, all its models are considered to be possible current states of the world. The goals is then to find an updated ABox $\mathcal{A} * \mathcal{U}$ that has exactly the successor of the models of $\mathcal{A}$ as its models, i.e., $\mathcal{A} * \mathcal{U}$ must be such that $M(\mathcal{A} * \mathcal{U}) = \{\mathcal{I}^{\mathcal{U}} \mid \mathcal{I} \in M(\mathcal{A})\}$.[3] In general, such an updated ABox need not exists.

The minimal DLs that contain both the basic DL $\mathcal{ALC}$ and are closed under ABox updates are $\mathcal{ALCO}^@$ and Boolean $\mathcal{ALCO}$. For $\mathcal{ALCO}^@$, updated ABoxes are exponential in the size of the original ABox and the update. The DL $\mathcal{ALCO}^+$ admits updated ABoxes that are exponential in the size of the update, but polynomial in the size of the original ABox. This is the reason why, in this work, we focus on $\mathcal{ALCO}^+$ and $\mathcal{ALCO}^@$. The following two propositions, which are simplified and streamlined versions of the ones given in [Liu *et al.*, 2006], tell us how updated ABoxes can be computed for these two DLs:

Proposition 2.2 (Updated ABox for $\mathcal{ALCO}^+$) *Let* $\alpha^{\mathcal{U}}$ *be the concept (role) obtained by the construction defined in Figure 1. Let the ABox* $\mathcal{A}'$ *be defined as*

$$\mathcal{A}' = \bigwedge(\mathcal{A} \cup \mathcal{U}) \vee \bigwedge(\mathcal{A}^{\mathcal{U}} \cup \mathcal{U}), \tag{1}$$

where the ABox $\mathcal{A}^{\mathcal{U}}$ *is defined as* $\mathcal{A}^{\mathcal{U}} = \{\alpha^{\mathcal{U}}(\bar{t}) \mid \alpha(\bar{t}) \in \mathcal{A}\}$. *Then* $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$.

For the DL $\mathcal{ALCO}^@$, the part of the construction of updated concepts $C^{\mathcal{U}}$ that differs from the construction for $\mathcal{ALCO}^+$ is depicted in Figure 2. Here $\mathsf{Obj}(\mathcal{U})$ denotes all the individuals that occur in the update $\mathcal{U}$.

Proposition 2.3 (Updated ABox for $\mathcal{ALCO}^@$) *For* $\mathcal{ALCO}^@$ *the ABox* $\mathcal{A}^{\mathcal{D}}$ *is defined as*

$$\begin{aligned} \mathcal{A}^{\mathcal{D}} = &\{C^{\mathcal{D}}(a) \mid C(a) \in \mathcal{A}\} \cup \\ &\{r(a,b) \mid r(a,b) \in \mathcal{A} \wedge \neg r(a,b) \notin \mathcal{D}\} \cup \\ &\{\neg r(a,b) \mid \neg r(a,b) \in \mathcal{A} \wedge r(a,b) \notin \mathcal{D}\}. \end{aligned}$$

Let $\mathcal{A}'$ *be as defined in (1). Then* $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$.

To see how the construction of updated ABoxes works consider the following example:

Example 2.4 (Updated ABox) *Let the ABox* $\mathcal{A} = \{A(a)\}$ *be updated with* $\mathcal{U} = \{\neg A(a)\}$. *Following Propositions 2.2 and 2.3 we obtain the (highly redundant) updated ABox*

$$\{(A(a) \wedge \neg A(a)) \vee ((A \sqcup \{a\})(a) \wedge \neg A(a))\},$$

which can be simplified to $\{\neg A(a)\}$. *The first disjunct is for the case that the update was already true, whereas the second disjunct is for the case that it wasn't.*

[1]A list of DL reasoners is available at `http://www.cs.man.ac.uk/~sattler/reasoners.html`.

[2]In [Liu *et al.*, 2006], an update is defined as a consistent set of literals, not as a single literal. Updating an ABox $\mathcal{A}$ with a set of literals can in our setting be achieved by iteratively updating $\mathcal{A}$ with the individual literals.

[3]It should be noted that for deterministic updates the Winslett semantics is uncontroversial, even though it does not extend to non-deterministic effects or ramifications.

$$
\begin{array}{llll}
A^{\mathcal{U}} = & (A \sqcup \bigsqcup_{\neg A(a)\in\mathcal{U}} \{a\}) \sqcap \neg(\bigsqcup_{A(a)\in\mathcal{U}} \{a\}) & & \\
r^{\mathcal{U}} = & (r \sqcup \bigsqcup_{\neg r(a,b)\in\mathcal{U}} \{(a,b)\}) \sqcap \neg(\bigsqcup_{r(a,b)\in\mathcal{U}} \{(a,b)\}) & & \\
\{a\}^{\mathcal{U}} = & \{a\} & \{(a,b)\}^{\mathcal{U}} = & \{(a,b)\} \\
(\neg C)^{\mathcal{U}} = & \neg C^{\mathcal{U}} & (\neg r)^{\mathcal{U}} = & \neg r^{\mathcal{U}} \\
(C \sqcap D)^{\mathcal{U}} = & C^{\mathcal{U}} \sqcap D^{\mathcal{U}} & (r \sqcap q)^{\mathcal{U}} = & r^{\mathcal{U}} \sqcap q^{\mathcal{U}} \\
(C \sqcup D)^{\mathcal{U}} = & C^{\mathcal{U}} \sqcup D^{\mathcal{U}} & (r \sqcup q)^{\mathcal{U}} = & r^{\mathcal{U}} \sqcup q^{\mathcal{U}} \\
(\exists r.C)^{\mathcal{U}} = & \exists r^{\mathcal{U}}.C^{\mathcal{U}} & (\forall r.C)^{\mathcal{U}} = & \forall r^{\mathcal{U}}.C^{\mathcal{U}}
\end{array}
$$

Figure 1: Constructing $C^{\mathcal{U}}$ and $r^{\mathcal{U}}$ for $\mathcal{ALCO}^{+}$

$$
\begin{array}{ll}
(@_i C)^{\mathcal{U}} = & @_i C^{\mathcal{U}} \\
(\exists r.C)^{\mathcal{U}} = & (\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\} \sqcap \exists r.C^{\mathcal{U}}) \sqcup \exists r.(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\} \sqcap C^{\mathcal{U}}) \\
 & \sqcup \bigsqcup_{a,b\in\mathsf{Obj}(\mathcal{U}),r(a,b)\notin\mathcal{U}} (\{a\} \sqcap \exists r.(\{b\} \sqcap C^{\mathcal{U}})) \sqcup \bigsqcup_{\neg r(a,b)\in\mathcal{U}} (\{a\} \sqcap @_b C^{\mathcal{U}}) \\
(\forall r.C)^{\mathcal{U}} = & (\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\} \rightarrow \forall r.C^{\mathcal{U}}) \sqcap \forall r.(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\} \rightarrow C^{\mathcal{U}}) \\
 & \sqcap \bigsqcap_{a,b\in\mathsf{Obj}(\mathcal{U}),r(a,b)\notin\mathcal{U}} (\{a\} \rightarrow \forall r.(\{b\} \rightarrow C^{\mathcal{U}})) \sqcap \bigsqcap_{\neg r(a,b)\in\mathcal{U}} (\{a\} \rightarrow @_b C^{\mathcal{U}})
\end{array}
$$

Figure 2: Constructing $C^{\mathcal{U}}$ for $\mathcal{ALCO}^{@}$

3 Optimizations for ABox Updates

It turns out that a naive implementation of the update algorithms based on Proposition 2.2 or 2.3 is not practical. Even for very simple update problems — where simple means e.g. small initial ABoxes containing only literals — after only a few updates we obtain ABoxes so huge and redundant that the reasoners cannot handle them anymore. In this section we propose a range of techniques for obtaining less redundant updated ABoxes.

In particular we are looking for ABoxes that are smaller than, but equivalent to, the updated ABoxes. In principle this could be done by enumerating ever bigger ABoxes, and checking for equivalence to the updated ABox. This is not likely to be practical, though. Instead we focus on logical transformations for obtaining smaller updated ABoxes. Since these transformations can be computationally expensive themselves, we also identify fragments of the transformations that we expect to be relatively cheap. The proposed techniques are each motivated by avoidable redundancy that we observed in practical examples. We present the various techniques for obtaining smaller updated ABoxes individually; they can be combined in a modular fashion.

Updating Boolean ABoxes

Updating an ABox according to Proposition 2.2 or 2.3 results in a Boolean ABox. In [Liu *et al.*, 2006] this updated ABox is transformed to a non-Boolean ABox using the @-constructor, before it is updated again. The following observation shows that Boolean ABoxes can directly be updated again by updating the individual assertions, avoiding the transformation.

Observation 3.1 (Distributivity of Update) *Update distributes over conjunction and disjunction in Boolean ABoxes; i.e.*

$$(\mathcal{A}_1 \boxtimes \mathcal{A}_2) * \mathcal{U} \equiv (\mathcal{A}_1 * \mathcal{U}) \boxtimes (\mathcal{A}_2 * \mathcal{U}),$$

where $\boxtimes$ denotes either $\wedge$ or $\vee$ (negation can be pushed inside the assertions).

By updating a Boolean ABox directly we also obtain a slightly more compact representation than the original one — the update $\mathcal{U}$ is no longer contained in two disjuncts:

Observation 3.2 (Updating Boolean ABoxes) *For a Boolean ABox $\mathcal{A}$ (we assume negation has been pushed inside the assertions), let the updated ABox $\mathcal{A}'$ be defined as*

$$\mathcal{A}' = (\mathcal{A} \circledast \mathcal{U}) \wedge \bigwedge \mathcal{U}.$$

Here $\mathcal{A} \circledast \mathcal{U}$ is defined recursively as

$$
\begin{array}{ll}
\alpha \circledast \mathcal{U} = & \alpha \vee \alpha^{\mathcal{U}} \\
(\alpha \boxtimes \mathcal{B}) \circledast \mathcal{U} = & (\alpha \circledast \mathcal{U}) \boxtimes (\mathcal{B} \circledast \mathcal{U})
\end{array}
$$

where $\boxtimes$ denotes $\wedge$ or $\vee$, α is an assertion, and $\{\alpha\}^{\mathcal{D}}$ is defined as in Proposition 2.2 (or 2.3) for $\mathcal{ALCO}^{+}$ (or for $\mathcal{ALCO}^{@}$, respectively). Then $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$.

Determinate Updates

Looking at the construction of updated ABoxes, we see that from an ABox $\mathcal{A}$ by an update we get a disjunction $\mathcal{A} \vee \mathcal{A}^{\mathcal{U}}$. This causes a rapid growth of the updated ABox. If, however, either the update or its negation is entailed by the ABox $\mathcal{A}$, then one of the disjuncts is inconsistent and can be removed:

Observation 3.3 (Determinate Updates) *For (Boolean) ABox $\mathcal{A}$, update $\mathcal{U} = \{\delta\}$, and updated ABox $\mathcal{A}'$ we have that $\mathcal{A}' \equiv \mathcal{A}$ if $\mathcal{A} \models \delta$; and $\mathcal{A}' \equiv \mathcal{U} \cup \mathcal{A}^{\mathcal{U}}$ if $\mathcal{A} \models \neg\delta$.*[4] *Otherwise, if neither $\mathcal{A} \models \delta$ nor $\mathcal{A} \models \neg\delta$, both $\mathcal{A}^{\emptyset}$ and $\mathcal{A}^{\mathcal{U}}$ are consistent with $\mathcal{U}$.*

Detecting this type of situation requires up to two reasoning steps: $\mathcal{A} \models \delta$ and $\mathcal{A} \models \neg\delta$, resulting in a tradeoff between time and space efficiency.

Exploiting the Unique Name Assumption

The common unique name assumption (UNA) means that no two individual names may denote the same object. The constructions depicted in Figure 1 and 2 do not take the UNA

[4]The latter of these two observations is from [Liu *et al.*, 2006].

into account; but we can construct simpler updated ABoxes by keeping track of the individuals $\bar{s}$ and $\bar{t}$ that an assertion $\gamma(\bar{s})$ refers to when updating it with $\delta(\bar{t})$:

Observation 3.4 (Updated Assertion with UNA) *Let $\mathcal{A}$ be an ABox, $\mathcal{U}$ an update, and $\mathcal{A}'$ the updated ABox. Further, let the ABox $\mathcal{B}$ be obtained by re-defining the ABox $\mathcal{A}^{\mathcal{U}}$ as $\mathcal{A}^{\mathcal{U}} = \{C^{\mathcal{U},a}(a) \mid C(a) \in \mathcal{A}\} \cup \{r^{\mathcal{U},a-b}(a,b) \mid r(a,b) \in \mathcal{A}\}$ for $\mathcal{ALCO}^+$,*

where $C^{\mathcal{U},i}$ and $r^{\mathcal{U},i-j}$ are given in Figure 3.[5] *For $\mathcal{ALCO}^@$ ABoxes we only use the modified construction of concept assertions. Then $\mathcal{A}' \equiv \mathcal{B}$.*

This UNA-based construction is not costly at all. It cannot identify all cases where the UNA admits a more concise updated ABox, though. The next example illustrates both its strength and limitations:

Example 3.5 (Exploiting UNA) *If we update the ABox $\mathcal{A} = \{C(i)\}$ with $\mathcal{U} = \{\neg C(j)\}$, using $C^{\mathcal{U},i}$ we obtain $C(i)$, instead of $C \sqcup \{j\}(i)$ using $C^{\mathcal{U}}$. Next consider the ABox $\mathcal{A} = \{\forall r.(\{j\} \sqcap C)\}(i)$, updated by $\mathcal{U} = \{C(k)\}$. As part of the update construction we obtain $\forall r.(\{j\} \sqcap (C \sqcap \neg\{k\}))(i)$ which can be simplified using UNA to $\forall r.(\{j\} \sqcap C)(i)$. Our method for exploiting UNA cannot detect this latter case.*

Omitting Subsuming Disjuncts and Entailed Assertions

Intuitively, in a disjunction we can omit the “stronger” of two disjuncts:

Observation 3.6 (Omitting Subsuming Disjuncts) *Let the disjunction $(\mathcal{A} \vee \mathcal{A}^{\mathcal{U}})$ be part of an updated ABox. If $\mathcal{A} \vDash \mathcal{A}^{\mathcal{U}}$ (or $\mathcal{A}^{\mathcal{U}} \vDash \mathcal{A}$) then $(\mathcal{A} \vee \mathcal{A}^{\mathcal{U}}) \equiv \mathcal{A}^{\mathcal{U}}$ (or $(\mathcal{A} \vee \mathcal{A}^{\mathcal{U}}) \equiv \mathcal{A}$).*

Detecting subsuming disjuncts in general requires reasoning. But by a simple, syntactic check we can detect beforehand some cases where one of the disjuncts $\mathcal{A}^{\mathcal{U}}$ and $\mathcal{A}$ will subsume the other. Then the computation of subsuming disjuncts can be avoided. We say that an occurrence of a concept or role name δ in an assertion is *positive*, if it is in the scope of an even number of negation signs, and *negative* otherwise; *δ occurs only positively (negatively)* in an assertion if every occurrence of δ is positive (negative).

Observation 3.7 (Detecting Subsuming Disjuncts) *If for an ABox $\mathcal{A}$, updated with update $\mathcal{U} = \{(\neg)\delta(\bar{t})\}$, we have that:*

(1) if the update is positive (i.e. $\delta(\bar{t})$) then
   - *if δ occurs only positively in $\mathcal{A}$ then $\mathcal{A}^{\mathcal{U}} \vDash \mathcal{A}$; and*
   - *if δ occurs only negatively in $\mathcal{A}$ then $\mathcal{A} \vDash \mathcal{A}^{\mathcal{U}}$.*

(2) if the update is negative (i.e. $\neg\delta(\bar{t})$) then
   - *if δ occurs only positively in $\mathcal{A}$ then $\mathcal{A} \vDash \mathcal{A}^{\mathcal{U}}$; and*
   - *if δ occurs only negatively in $\mathcal{A}$ then $\mathcal{A}^{\mathcal{U}} \vDash \mathcal{A}$.*

Conversely, we can also avoid updating entailed assertions:

Observation 3.8 (Omitting Entailed Assertions) *Let $\mathcal{A}$ be an ABox and $\mathcal{U}$ an update. If $\mathcal{U} \models \alpha$ or $\mathcal{A} \setminus \{\alpha\} \models \alpha$ for some assertion $\alpha \in \mathcal{A}$, then $\mathcal{A} * \mathcal{U} \equiv (\mathcal{A} \setminus \{\alpha\}) * \mathcal{U}$.*

[5]We omit the Boolean constructors.

Removing all entailed assertions might be too expensive in practice; one might try doing this periodically.

Propositional ABoxes

Sometimes we do not need the full power of DL reasoning, but propositional reasoning is enough:

Definition 3.9 (Propositional ABox) *We call a Boolean ABox $\mathcal{A}$* propositional *if it does not contain quantifiers.* $\dashv$

For propositional ABoxes we could in principle use progression algorithms for propositional logic [Amir and Russell, 2003] and efficient SAT-technology, since an updated propositional ABox is propositional, too.

Independent Assertions

Next we address the question under which conditions an assertion in an ABox is not affected by an update, i.e. independent. The more independent assertions we can identify, the more compact our ABox representation becomes.

Definition 3.10 (Independent Assertion) *Assertion α in an ABox $\mathcal{A}$ is independent from update $\mathcal{U} = \{\delta\}$ iff $\mathcal{A} * \mathcal{U} \equiv \alpha \wedge (\mathcal{B} * \mathcal{U})$ where $\mathcal{B} = \mathcal{A} \setminus \{\alpha\}$.* $\dashv$

Detecting this in all cases requires reasoning steps and thus is costly. It is easy, though, to syntactically detect some of the independent assertions:

Observation 3.11 (Independent Assertion) *For an ABox $\mathcal{A}$ in negation normal form and update $\mathcal{U} = \{(\neg)\delta(\bar{t}_1)\}$, the assertion $\alpha(\bar{t}_2) \in \mathcal{A}$ is independent if $\delta \notin \mathsf{sub}(\alpha)$. It is also independent if $\mathcal{A} \vDash \bar{t}_1 \neq \bar{t}_2$, δ occurs in α only outside the scope of a quantifier, and for all subconcepts $@_i C$ of α the assertion $C(i)$ is independent of $\mathcal{U}$.*

4 Reasoning with Boolean ABoxes

As we have seen in the previous sections, updated ABoxes are Boolean $\mathcal{ALCO}^@$- or $\mathcal{ALCO}^+$-ABoxes, so that an intelligent agent built on top of ABox update needs Boolean ABox reasoning. Reasoning with $\mathcal{ALC}$-LTL formulas [Baader *et al.*, 2008] requires Boolean ABox reasoning, too. However, Boolean ABox reasoning is not directly supported by DL reasoners. In this section, we present four different reasoning methods that can handle Boolean ABoxes:

- one where a DL reasoner operates on single disjuncts of a Boolean ABox in DNF;
- one which uses Otter, a first-order theorem prover;
- one which uses a consistency preserving reduction from a Boolean ABox to a non-Boolean ABox; and
- one which is based on propositional satisfiability testing modulo theories — the DPLL(T) approach.

Replacing every assertion in a Boolean ABox $\mathcal{A}$ with a propositional letter results in a propositional formula $F_{\mathcal{A}}$. The ABox $\mathcal{A}$ is a *Boolean ABox in CNF (resp. DNF)* if $F_{\mathcal{A}}$ is in CNF (resp. DNF). The first approach works on Boolean ABoxes in DNF while the other approaches are based on CNF.

$A^{\mathcal{U},i} =$	$\top$, if $\mathcal{U} = \{\neg A(i)\}$	$A^{\mathcal{U},i} =$	$\bot$, if $\mathcal{U} = \{A(i) \in \mathcal{U}\}$
$A^{\mathcal{U},i} =$	A, if $\mathcal{U} \neq \{\neg A(i)\}$ and $\mathcal{U} \neq \{A(i)\}$		
$r^{\mathcal{U},i-j} =$	$\top$, if $\mathcal{U} = \{\neg r(i,j)\}$	$r^{\mathcal{U},i-j} =$	$\bot$, if $\mathcal{U} = \{r(i,j)\}$
$r^{\mathcal{U},i-j} =$	r, if $\mathcal{U} \neq \{\neg r(i,j)\}$ and $\mathcal{U} \neq \{r(i,j)\}$		
$\{i\}^{\mathcal{U},i} =$	$\top$	$\{i\}^{\mathcal{U},j} =$	$\bot$
$\{(i,j)\}^{\mathcal{U},i-j} =$	$\top$	$\{(i,j)\}^{\mathcal{U},k-l} =$	$\bot$, if $k \neq i$ or $l \neq j$
$(\exists r.C)^{\mathcal{U},i} =$	$\exists r.(C^{\mathcal{U}})$, if $\mathcal{U} \neq \{q(i,x)\}$ for $q \in \mathsf{sub}(r)$	$(\forall r.C)^{\mathcal{U},i} =$	$\forall r.(C^{\mathcal{U}})$, if $\mathcal{U} \neq \{q(i,x)\}$ for $q \in \mathsf{sub}(r)$
$(\exists r.C)^{\mathcal{U},i} =$	$(\exists r.C)^{\mathcal{U}}$, otherwise	$(\forall r.C)^{\mathcal{U},i} =$	$(\forall r.C)^{\mathcal{U}}$, otherwise
$(@_j C)^{\mathcal{U},i} =$	$@_j C^{\mathcal{U},j}$	$(@_i C)^{\mathcal{U}} =$	$@_i C^{\mathcal{U},i}$

Figure 3: Constructing $C^{\mathcal{U},i}$ and $r^{\mathcal{U},i-j}$ for $\mathcal{ALCO}^+$ and $\mathcal{ALCO}^@$

In all approaches we do not use the equivalence-preserving, exponential transformation from [Liu *et al.*, 2006] for compiling the @ constructor away. Instead we simulate the @-operator by a universal role [Bong, 2007]; this consistency-preserving transformation is linear.

We use Pellet as a DL reasoner because it supports nominals, query-answering and pinpointing [Sirin *et al.*, 2007].

The DNF Approach

A Boolean ABox in DNF is consistent iff it contains a consistent disjunct. We can employ a DL reasoner to decide the consistency of each disjunct. We refer to this approach as Pellet-DNF. A drawback of this approach is that we will see that the less redundant updated ABoxes are in CNF, and thus require a costly translation to DNF (using de Morgan's laws).

The Theorem Prover Approach

The DL $\mathcal{ALCO}^+$ admits smaller updated ABoxes than $\mathcal{ALCO}^@$ [Liu *et al.*, 2006]; however, its role operators are not supported by current mature DL reasoners. Once we translate $\mathcal{ALCO}^+$ to first order logic [Borgida, 1996], we can use theorem provers that can cope with Boolean role constructors. We chose to use Otter [McCune, 2003] because it supports query-answering via answer literals [Green, 1969]; this is useful e.g. for parametric actions, which are to be instantiated to concrete actions. After a few experiments we chose to configure Otter to use hyperresolution combined with Knuth-Bendix-rewriting, plus the set-of-support strategy.

The Reduction Approach

We can linearly compile Boolean $\mathcal{ALCO}^@$-ABoxes to classical $\mathcal{ALCO}^@$-ABoxes [Liu *et al.*, 2006]. Then, simulating the @-operator by a universal role, we can directly use a standard DL reasoner; this approach is henceforth called Pellet-UR.

The DPLL(T) Approach

Most modern SAT-solvers [Een and Sörensson, 2003; de Moura and Bjørner, 2008] are variants of the Davis-Putnam-Logemann-Loveland (DPLL) procedure [Davis and Putnam, 1960; Davis *et al.*, 1962]. Such a SAT-solver exhaustively applies transition rules[6] to generate and extend a partial interpretation and thus decides satisfiability of a propositional formula in CNF. One of the strengths of the DPLL procedures is that they can efficiently prune the search space by building and learning backjump clauses [Zhang *et al.*, 2001].

The DPLL(T) approach combines a DPLL procedure with a theory solver that can handle conjunctions of literals in the theory to solve the satisfiability problem modulo theories (SMT) [Nieuwenhuis *et al.*, 2007]. In DPLL(T) a DPLL procedure works on the propositional formula obtained by replacing the theory atoms with propositional letters. Whenever the DPLL procedure extends the current partial interpretation by a new element the theory solver is invoked to check consistency of the conjunction of the theory atoms corresponding to the partial, propositional interpretation. If the theory solver reports an inconsistency, the DPLL procedure will backjump and thus the search space is pruned.

The consistency problem of Boolean ABoxes can be viewed as an instance of SMT where ABox assertions are the theory atoms and a DL reasoner serves as theory solver.

The non-standard DL inference of pinpointing [Schlobach, 2003; Baader and Peñaloza, 2008] is highly relevant to this approach. Explaining why an ABox is inconsistent is an instance of the pinpointing problem, where an explanation is a minimal sub-conjunction of the input ABox, containing only those assertions that are responsible for the inconsistency. Based on these explanations in the DPLL(T) approach we can build better backjump clauses [Nieuwenhuis *et al.*, 2007].

We implemented an algorithm based on the DPLL(T) approach with the strategy of MiniSat [Een and Sörensson, 2003]. Pellet was chosen as the theory solver because it supports pinpointing. This approach is called Pellet-DPLL.

Propositional Reasoning

For the case where we can identify propositional ABoxes we have developed and implemented a simple, specialized method. Reasoning there is reduced to efficient list operations. This reasoner is used to supplement the other reasoning approaches (if possible).

5 Experimental Results

In this section, we evaluate the efficiency of the different update and reasoning mechanisms. The relevant measures are the time needed for computing the updated ABox together with its size, and the efficiency of reasoning with it. We will see that choosing the right update and reasoning algorithms depends upon a problem's specifics.

An update algorithm based on Proposition 2.2 or 2.3 generates Boolean ABoxes in DNF, while an algorithm based on Proposition 3.2 outputs ABoxes in CNF. Of course, every Boolean ABox can equivalently be represented in CNF or in DNF; however, this transformation (using De Morgan's laws) is rather expensive. The performance of reasoning with updated ABoxes strongly depends on the choice of underlying representation. We use several types of testing data:

[6] See [Nieuwenhuis *et al.*, 2007] for the details.

- a set of randomly generated Boolean ABoxes in CNF;
- a set of random ABoxes, Updates, and Queries; and
- the Wumpus world [Russell and Norvig, 2003].

We distinguish two main types of update algorithms that we implemented:

- In one we compute updated ABoxes in DNF;
- alternatively, we compute updated ABoxes in CNF.

Both approaches are further parametrized by using different reasoners, and a different combination of optimization techniques. We have implemented the different ABox update algorithms in ECLiPSe-Prolog.

The reasoning methods have already been described in Section 4. We call a reasoning method *hybrid* if it resorts to our propositional reasoner whenever possible; for example, we then speak of hybrid Pellet-UR.

5.1 Representation: DNF or CNF?

We have used both the Wumpus world and the random update examples to compare DNF and CNF based update algorithms (with and without optimizations). CNF representation consistently proved to be superior: The DNF approach quickly drowns in redundant information. This is because to compute an updated ABox in DNF is to include both the update and all the non-affected information in both disjuncts. Detecting subsuming disjuncts and determinate updates alleviates this problem, but does not eliminate it. By avoiding this redundancy we immediately obtain an updated ABox in CNF. On DNF-based updated ABoxes Pellet-DNF performs best — the other methods suffer from the expensive conversion to CNF. In the following we only consider the CNF-based representation of updated ABoxes.

5.2 Consistency Checking for Boolean ABoxes in CNF

We implemented a random generator of Boolean $\mathcal{ALC}$-ABoxes, which randomly generates a propositional formula in CNF and then assigns a randomly generated assertion to each propositional letter. Several parameters are used to control the shape of the generated Boolean ABoxes (the numbers in parentheses indicate the upper bound on the parameters we used): the number n_1 of literals in a clause (53), the number n_2 of propositional letters (36), the number n_3 of clauses (83), the number d of nested roles in a concept assertion (23), the number ncs of the constructors in a concept assertion (106), the numbers nc, nr, and ni of concept names, role names, and individual names in an assertion (12 each), and the probability pr of generating a role assertion (0.2).

In Figure 4, we plot the runtimes of Pellet-DPLL and Pellet-UR on these testing data against the number of symbols in the Boolean ABox. The points plotted as $+$ indicate the runtime of Pellet-DPLL while those plotted as $\times$ indicate the runtime of Pellet-UR. We depict the performance on consistent and inconsistent Boolean ABoxes separately — there were more consistent than inconsistent Boolean ABoxes.

For Pellet-UR, the runtime linearly increases with the size of the input (the bar from the lower left to the upper right corner). On inconsistent ABoxes Pellet-DPLL also exhibits a linear increase in runtime, while on consistent ABoxes the runtime is less predictable. Pellet-DPLL performs better on all of the inconsistent Boolean ABoxes. On most of the consistent ABoxes, the Pellet-UR approach does better. This is due to the fact that in Pellet-DPLL the frequent invocations of the theory solver Pellet are more likely to pay off if inconsistency of the current, partial model can be detected often: We then can build a back-jump clause that helps to prune the search space. The runtimes of Pellet-UR are about the same on both consistent and inconsistent input data.

For Otter the conversion from ABoxes in CNF to full first order CNF proved to be a big obstacle, as did the conversion to DNF for Pellet-DNF.

5.3 Random Updates

We have extensively experimented with a set of randomly generated ABoxes and updates. Initial ABoxes were between two and thirty assertions in size.We were mostly interested in runtime and space consumption for iterated updates. We could make a number of interesting observations:

- The UNA-based concept update construction from Figure 3 always paid.
- The reasoning needed to identify determinate updates pays in the long run.
- Syntactically detecting subsuming disjuncts worked, too. Doing so using a reasoner proved too expensive.
- Identifying all entailed assertions to shrink the ABoxes proved to be too expensive, too.
- Resorting to our dedicated propositional reasoner whenever possible resulted in significantly better performance.
- We can keep updated ABoxes much smaller at a low cost by syntactically identifying independent assertions.

Updating an ABox according to [Liu *et al.*, 2006] is a purely syntactic procedure. But if we iteratively update ABoxes, then in the long run we get both a lower space and time consumption by calling a reasoner to identify determinate updates. Using our propositional reasoner whenever possible for this resulted in better performance. If identifying determinate updates required DL reasoning then Pellet-UR performed slightly better than Pellet-DPLL. This is due to the fact that less updates were determinate than not, and thus inconsistency was not detected often. On a subset of the random examples where there were more determinate updates Pellet-DPLL performed better than Pellet-UR. The runtimes for Otter widely varied: converting CNF-ABoxes to full first order CNF proved the bottleneck. Pellet-DNF was not competitive because of the expensive conversion to DNF.

We could also identify characteristics of initial ABoxes that allow to predict performance: If the initial ABox does not contain nested quantifiers then performance is acceptable; e.g. we can iteratively apply 300 singleton updates to a fifteen assertion ABox in 90 seconds, without a significant increase in size. If the initial ABox contains nested quantifiers space consumption quickly grows out of bounds. This is because

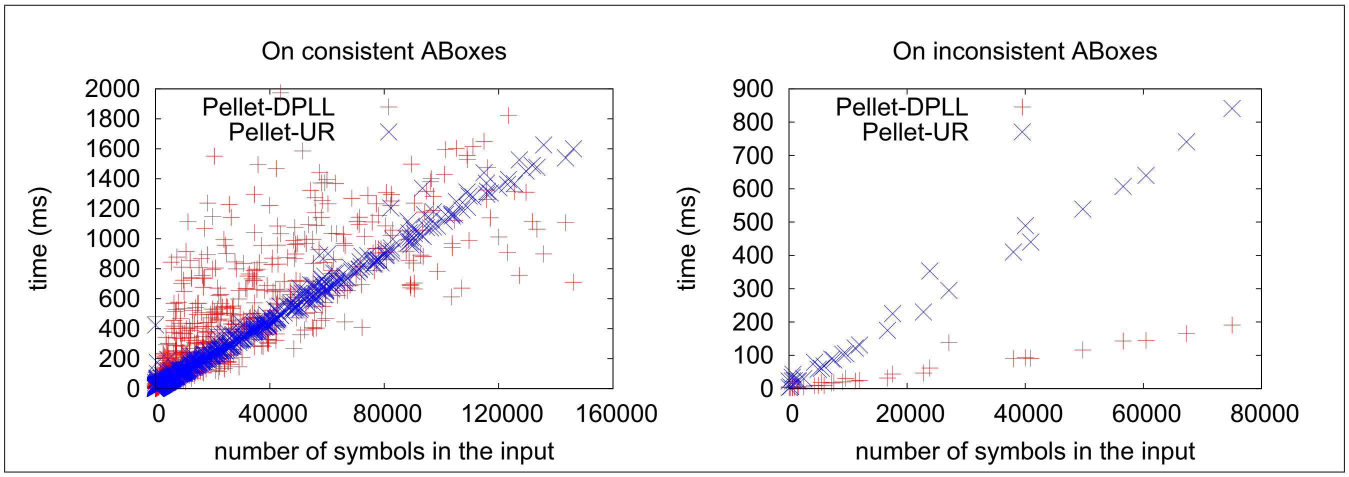


Figure 4: Benchmarks for Pellet-DPLL and Pellet-UR

we then cannot cheaply identify independent assertions and use the UNA-based concept update construction. For nested quantifiers using $\mathcal{ALCO}^+$ instead of $\mathcal{ALCO}^@$ helps to reduce space consumption; but this still does not result in satisfactory overall performance.

5.4 The Wumpus World

The Wumpus World [Russell and Norvig, 2003] is a well-known challenge problem in the reasoning about action community. It consists of a grid-like world: cells may contain pits, one cell contains gold, and one the fearsome Wumpus. The agent dies if she enters a cell containing a pit or the Wumpus. But she carries one arrow so that she can shoot the Wumpus from an adjacent cell. If the agent is next to a cell containing a pit (Wumpus), she can detect that one of the surrounding cells contains a pit (the Wumpus), but doesn't know which one. She knows the contents of the visited cells. Getting the gold without getting killed is the agent's goal.

At each step, the agent performs sensing to learn whether one of the adjacent cells contains a pit or the Wumpus. Since the sensing results are disjunctive, we cannot treat them via ABox updates. But since the properties sensed are static (i.e., cannot change once we know them), we can simply adjoin the sensing results to the ABox serving as the agent's current world model. The effects of the agent's (non-sensing) actions (like moving to another cell) are modelled as ABox update.

The Wumpus World can be modelled in different ways. In the simplest model, the initial ABox contains the connections between the cells, the agent's location, and the facts that the agent carries an arrow, and that the Wumpus is alive (Model PL1). For this, Boolean combinations of concept/role literals are enough. In Model PL2, we include the fact that the Wumpus is at exactly one location by enumerating all possible cases in a big disjunction. We turn PL1 into a DL problem by including the information $\exists \text{at}.\top(\text{wumpus})$ (Model DL1). Model DL2 is obtained from PL2 by adding this same assertion, which here is redundant. Table 2 shows the runtimes, where n/a stands for unavailable expressivity and * for non-termination in 15 minutes. For the propositional models we also used the action language Flux [Thielscher, 2005].

Pellet-DNF, and to a lesser extent also Otter, again had difficulties with the necessary input conversion. Pellet-UR

Model	Prop	hybrid Otter	hybrid Pellet-UR	Flux
4x4 PL1	0.008 s	0.008 s	0.008 s	0.6 s
8x8 PL1	0.26 s	0.26 s	0.26 s	14.9 s
8x8 PL2	16.9 s	16.9 s	16.9 s	n/a
4x4 DL1	n/a	36.4 s	5.5 s	n/a
4x4 DL2	n/a	*	23.93 s	n/a

Table 2: Runtimes for the Wumpus World.

proved to be the best DL reasoner in this setting. This is due to the fact that this domain requires query-answering: The agent e.g. needs to know for which values of x and y we have that $\text{at}(\text{agent}, x) \wedge \text{connected}(x, y)$. Pellet-DPLL is the only reasoner that lacks direct support for query-answering. Thus, for query $C(x)$, we iteratively check for every individual name $i \in \mathsf{N_I}$ whether $C(i)$ holds — but this results in bad performance for Pellet-DPLL.

We also see that the propositional reasoner performs quite well on the propositional models. Including more information wrt. the Wumpus' location results in worse performance. We used Model DL2 to see if it pays to identify all entailed assertions: after omitting the entailed $\exists \text{at}.\top(\text{wumpus})$ the model is propositional again. In practice this proved too costly. The other observations from Section 5.3 also hold in this domain. Sometimes removing assertions entailed by the update did help, though. In particular, once the Wumpus is found, we can remove the assertion $\exists \text{at}.\top(\text{wumpus})$ entailed by the respective update and then resort to efficient propositional reasoning.

6 Summary and Future Work

In this work, we have investigated implementation techniques for ABox update, and for reasoning with (updated) Boolean ABoxes. We have introduced and evaluated several optimizations of the ABox update algorithms in [Liu *et al.*, 2006]. The lessons learnt were: Using CNF-representation of updated ABoxes is strongly recommended. The (incomplete) syntactic techniques for exploiting the unique name assumption, and detecting subsuming disjuncts and independent assertions have also resulted in an improved performance. The benefit of identifying determinate updates made up for the associated reasoning costs. Other techniques requiring DL reasoning in general proved to be too expensive; but removing some entailed assertions helped in the Wumpus world.

Regarding the investigated reasoning methods for Boolean ABoxes, we have come to the following conclusions. Pellet-DNF is the best reasoner for Boolean ABoxes in DNF. For consistency checking of ABoxes in CNF, Pellet-DPLL and Pellet-UR worked best. Pellet-DPLL did better for detecting an actual inconsistency, while it performed worse than Pellet-UR on most of the consistent Boolean ABoxes. On the randomly generated update examples, Pellet-UR also performed slightly better than Pellet-DPLL because inconsistency was not detected often. On a subset where the updates were mostly determinate, Pellet-DPLL outperformed Pellet-UR. If query-answering is among the reasoning tasks, then Pellet-UR is to be preferred over Pellet-DPLL because of Pellet's direct support for this inference.

It would be interesting to develop heuristics for finding suitable individual names as well as other optimizations for query-answering in the DPLL(T) approach. The performance of the DPLL(T) approach also depends on the performance of the SAT solver and the pinpointing service. Thus Pellet-DPLL can benefit from more efficient implementation of these tasks as well.

The tests on the Wumpus world confirmed that resorting to our dedicated propositional reasoner whenever possible is useful. In the Wumpus world, removing entailed assertions helped a lot. In contrast, for the randomly generated update examples, finding entailed assertions proved to be too costly.

Using Otter as a theorem prover might be considered somewhat unfair (to the theorem proving approach), since it is no longer actively maintained and optimized. The conversion to full first order CNF proved to be the biggest obstacle for Otter. We chose to use Otter because it supports query-answering, which is not supported by most current provers [Waldinger, 2007], but vital in some domains. If this is to change,[7] we can try to resort to state-of-the art theorem provers for reasoning in $\mathcal{ALCO}^+$. This may allow us to really exploit the fact that $\mathcal{ALCO}^+$ admits smaller updated ABoxes than $\mathcal{ALCO}^@$. Alternatively, one could also try to use a more dedicated reasoning system for $\mathcal{ALCO}^+$ [Schmidt and Tishkovsky, 2007].

Acknowledgments: Many thanks to Albert Oliveras for his help regarding the construction of a backjump clause in the DPLL(T) approach.

References

[Amir and Russell, 2003] Eyal Amir and Stuart J. Russell. Logical Filtering. In *(IJCAI2003)*, 2003.

[Areces *et al.*, 1999] Areces, Blackburn, and Marx. A road-map on complexity for hybrid logics. In *(CSL1999)*, 1999.

[Baader and Peñaloza, 2008] Franz Baader and Rafael Peñaloza. Automata-Based Axiom Pinpointing. In *(IJCAR2008)*, 2008.

[Baader *et al.*, 2003] F. Baader, D. Calvanese, D. L. Mcguinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[Baader *et al.*, 2005] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating Description Logics and Action Formalisms: First Results. In *(AAAI2005)*, 2005.

[Baader *et al.*, 2008] Franz Baader, Silvio Ghilardi, and Carsten Lutz. LTL over Description Logic Axioms. In *(KR2008)*, 2008.

[Bong, 2007] Yusri Bong. Description Logic ABox Updates Revisited. Master thesis, TU Dresden, Germany, 2007.

[Borgida, 1996] Alexander Borgida. On the Relative Expressiveness of Description Logics and Predicate Logics. *Artificial Intelligence*, 1996.

[Davis and Putnam, 1960] Martin Davis and Hilary Putnam. A Computing Procedure for Quantification Theory. *Journal of the ACM*, 1960.

[Davis *et al.*, 1962] Martin Davis, George Logemann, and Donald Loveland. A Machine Program for Theorem-proving. *Communications of the ACM*, 1962.

[de Moura and Bjørner, 2008] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT Solver. In *(TACAS2008)*, 2008.

[Drescher and Thielscher, 2007] Conrad Drescher and Michael Thielscher. Integrating Action Calculi and Description Logics. In *(KI2007)*, 2007.

[Een and Sörensson, 2003] Niklas Een and Niklas Sörensson. An Extensible SAT-solver. In *(SAT2003)*, 2003.

[Green, 1969] Cordell Green. Theorem Proving by Resolution as a Basis for Question-answering Systems. *Machine Intelligence*, 1969.

[Levesque *et al.*, 1997] Hector Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 1997.

[Liu *et al.*, 2006] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating Description Logic ABoxes. In *(KR2006)*, 2006.

[McCune, 2003] William McCune. OTTER 3.3 Reference Manual. *Computing Research Repository*, 2003.

[Nieuwenhuis *et al.*, 2007] Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio. Challenges in Satisfiability Modulo Theories. In *(TRA2007)*, 2007.

[Russell and Norvig, 2003] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[Schlobach, 2003] Stefan Schlobach. Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies. In *(IJCAI-03)*, 2003.

[Schmidt and Tishkovsky, 2007] Renate A. Schmidt and Dmitry Tishkovsky. Using tableau to decide expressive description logics with role negation. In *(ISWC2007)*, 2007.

[Sirin *et al.*, 2007] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 2007.

[Thielscher, 2005] M. Thielscher. FLUX: A Logic Programming Method for Reasoning Agents. *Theory and Practice of Logic Programming*, 2005.

[Tobies, 2001] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH-Aachen, Germany, 2001.

[Waldinger, 2007] Richard J. Waldinger. Whatever happened to deductive question answering? In *(LPAR2007)*, 2007.

[Winslett, 1988] Marianne Winslett. Reasoning about Action Using a Possible Models Approach. In *(AAAI1988)*, 1988.

[Zhang *et al.*, 2001] Lintao Zhang, Conor F. Madigan, Matthew H. Moskewicz, and Sharad Malik. Efficient Conflict Driven Learning in a Boolean Satisfiability Solver. In *(ICCAD2001)*, 2001.

[7]cf. `www.cs.miami.edu/~tptp/TPTP/Proposals/AnswerExtraction.html`.

Modeling Multi-Agent Domains in an Action Languages: an Empirical Study Using $\mathcal{C}$

Chitta Baral
Dept. Computer Science & Engineering
Arizona State University
chitta@asu.edu

Tran Cao Son and **Enrico Pontelli**
Dept. Computer Science
New Mexico State University
tson | epontell @cs.nmsu.edu

Abstract

In this paper, we evaluate the expressiveness of the action language $\mathcal{C}$ in modeling multi-agent domains. Our investigation shows that, with minimal extensions, $\mathcal{C}$ can be adapted to model multi-agent domains in a natural way. We also show that the language is suitable for answering various types of queries that are interesting in modeling and analyzing multi-agent domains.

1 Introduction and Motivation

Representing and reasoning in multi-agent domains are two of the most active research areas in *multi-agent system (MAS)* research. The literature in this area is extensive, and it provides a plethora of logics for representing and reasoning about various aspects in multi-agent domains. For example, the authors of [Sauro *et al.*, 2006] combine an action logic and a cooperation logic to represent and reason about the capability and the cooperation between agents. [Gerbrandy, 2006] generalizes this framework to consider domains where an agent may control only a part of a proposition. In [van der Hoek *et al.*, 2005], an extension of Alternating-time Temporal Logic (ATL) is developed to facilitate strategic reasoning in multi-agent domains. The work in [Spaan *et al.*, 2006] suggests that decentralized partially observable Markov decision processes could be used to represent multi-agent domains and discusses the usefulness of agent communication in multi-agent planning. In [Herzig and Troquard, 2006], an extension of Alternating-time Temporal Epistemic Logic (ATEL) is proposed for reasoning about choices.

The rich collection of logics proposed in the literature for formalizing MAS reflects attempts to design solutions to address specific issues in modeling MAS, often justified by a specific application scenario. This makes such logics suitable to address specific subsets of the general features required to model real-world MAS domains. The task of generalizing some of these existing proposals to create a uniform and comprehensive framework for modeling several different aspects of MAS domains is, to the best of our knowledge, still an open problem. Although we do not dispute the possibility of extending several of these existing proposals in various directions, the task does not seem easy. It is worth noticing that, on the other hand, the need for a general language for MAS domains, with a formal and simple semantics, that allows for the verification of plan correctness has been extensively motivated, e.g., [Brenner, 2005].

The state of affairs in formalizing multi-agent systems reflects the same trend that occurred in the early nineties, regarding the formalization of *single agent* domains. Between the discovery of the frame problem [McCarthy and Hayes, 1969] and 1990, several formalisms for representing and reasoning in dynamic domains have been proposed. Frequently, new proposals have appeared to address shortcomings of previous approaches on specific example domains. For example, the well-known Yale Shooting problem [Hanks and McDermott, 1987] was invented to show that the early solutions to the frame problem fail. A simple solution to the Yale Shooting problem was proposed in [Baker, 1989], yet this solution was shown to fail in the Stolen Car example [Kautz, 1986], etc. Action languages [Gelfond and Lifschitz, 1998] were one of the outcomes of this development, and they have been proved to be very useful ever since.

Action description languages, first introduced in [Gelfond and Lifschitz, 1993] and further refined in [Gelfond and Lifschitz, 1998], are formal models used to describe dynamic domains, by focusing on the representation of effects of actions. Traditional action languages (e.g., $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$) have mostly focused on domains involving a single agent. In spite of different features offered by several of these languages (e.g., concurrent actions, sensing actions, non-deterministic behavior), there is a general consensus on what are the essential components of an action description language in single agent domains. In particular, an action specification focuses on the *direct effects* of each action on the state of the world; the semantics of the language takes care of all the other aspects concerning the evolution of the world (e.g., the ramification problem).

The analogy between the development of several formalisms for single agent domains and the development of several logics for formalizing multi-agent systems indicates the need for, and the usefulness of a formalism capable of dealing with multiple desiderata in the specification of MAS. A natural question that arises is whether single agent action languages can be adapted to describe MAS. *This is the main question that we explore in this paper.*

In this paper, we answer the above question by investigating whether an action language developed for single agent domains can be used, with minimal modifications, to model MAS domains with several features of interest. Our starting point is a well-studied and well-understood single agent language action language—the language $\mathcal{C}$ [Gelfond and Lifschitz, 1998]. This language is chosen as it provides a number of features that appear necessary to handle multi-agent domains, such as concurrent interacting actions. The language is employed to formalize examples drawn from the multi-agent literature, describing different types of problems that can arise when dealing with multiple agents. Whenever necessary, we identify weaknesses of $\mathcal{C}$ and introduce extensions that are adequate to model the domains. The resulting action language provides a unifying framework for modeling multi-agent domains. The language can be used as a foundation for different forms of reasoning in multi-agent domains (e.g., projection, validation of plans), which are formalized in the form of a query language.

Before we continue, let us discuss the desirable features MAS and the assumptions we will make in the rest of this work. In this paper, we consider a multi-agent systems (MAS) domain as an environment in which multiple agents can execute actions to change the environment. We assume that

- Agents can execute actions concurrently;
- Each agent knows its own capabilities;
- Actions executed by different agents can interact;
- Agents can communicate to exchange knowledge; and
- Knowledge can be private to an agent or shared among groups of agents.

The questions that we are interested to answer in a MAS domain involve

- *hypothetical reasoning*, e.g., what happens if agent A executes the action a; what happens if A executes a_1 and B executes b_1 at the same time; etc.
- *planning/capability*, e.g., can a specified group of agents achieves a certain goal from a state.

Variations of the above types of questions will also be considered. For example, what happens if agents do not have complete information, agents do not cooperate, agents have preferences, etc.

We would like to note that in the past there have been attempts to use action description languages to formalize multi-agent domains, e.g., [Boutilier and Brafman, 2001]. On the other hand, existing proposals address only some of the properties of the multi-agent scenarios mentioned earlier (e.g., concurrency).

To the best of our knowledge, this is the first investigation of how to adapt a standard singe-agent action language to meet the needs of MAS domains. It is also important to stress that the goal of this work is to create a framework for *modeling* MAS domains, with a query language that enables plan validation and various types of reasoning. In this work we do not deal with the issues of distributed plan generation—an aspect extensively explored in the existing literature. This is certainly an important research topic and worth pursuing but it is outside of the scope of this paper.

The paper is organized as follows. In the next section, we will review the basics of the action language $\mathcal{C}$. Section 3 describes a straightforwards adaptation of $\mathcal{C}$ for multi-agent domains. Each of the following sections (Sections 4–6) shows how minor additions to $\mathcal{C}$ can address several features in representing and reasoning about multi-agent domains. Section 7 presents the query language that can be used with the extended $\mathcal{C}$. Section 8 discusses further aspects of modeling MAS that the proposed extension of $\mathcal{C}$ cannot deal with easily. We conclude in Section 9.

2 Action Language $\mathcal{C}$

The starting point of our investigation is the action language $\mathcal{C}$ [Gelfond and Lifschitz, 1998]—an action description language originally developed to describe single agent domains, where the agent is capable of performing non-deterministic and concurrent actions. In this section we revise the basic definitions of $\mathcal{C}$.

A domain description in $\mathcal{C}$ builds on a language signature $\langle \mathcal{F}, \mathcal{A} \rangle$, where $\mathcal{F}$ is a finite collection of fluent names and $\mathcal{A}$ is a finite collection of action names. Both the elements of $\mathcal{F}$ and $\mathcal{A}$ are viewed as propositional variables, and they can be used in formulae constructed using the traditional propositional operators. A propositional formula over $\mathcal{F} \cup \mathcal{A}$ is referred to simply as a *formula*, while a propositional formula over $\mathcal{F}$ is referred to as a *state formula*. A fluent literal is of the form f or $\neg f$ for any $f \in \mathcal{F}$.

A domain description D in $\mathcal{C}$ is a finite collection of axioms of the following forms:

caused ℓ **if** F	*static causal law*
caused ℓ **if** F **after** G	*dynamic causa laws*

where ℓ is a fluent literal, F is a state formula, while G is a formula. The language also allows the ability to declare properties of fluents; in particular **non_inertial** ℓ declares that the fluent literal ℓ is to be treated as a non-inertial literal (i.e., ℓ returns true if no action affects it).

A problem specification is obtained by adding an initial state description $\mathcal{I}$ to a domain D, composed of axioms of the form **initially** ℓ, where ℓ is a fluent literal.

The semantics of the language can be summarized using the following concepts. An *interpretation* is described simply by a set of $\mathcal{F}$-literals, such that $I \cap \{f, \neg f \mid f \in \mathcal{F}\} = \emptyset$. Given an interpretation I and a fluent f (literal $\neg f$), we say that $I \models f$ ($I \models \neg f$) if $f \in I$ ($\neg f \in I$). The entailment relation $\models$ can be easily generalized to arbitrarily propositional formulae. An interpretation I is complete if, for each $f \in \mathcal{F}$, we have that $f \in I$ or $\neg f \in I$. An interpretation is *closed* w.r.t. a set of static causal laws $\mathcal{SC}$ if, for each static causal law **caused** ℓ **if** F, if $I \models F$ then $\ell \in I$. Given an interpretation I and a set of static causal laws $\mathcal{SC}$, we denote with $Cl_{\mathcal{SC}}(I)$ the smallest set of literals containing I and closed w.r.t. $\mathcal{SC}$.

A *state* s is a complete interpretation which is closed w.r.t. the static causal laws. Given a state s, a set of actions $A \subseteq \mathcal{A}$, and a collection of dynamic causal laws $\mathcal{DC}$, we define

$$Eff_{\mathcal{DC}}(s, A) = \left\{ \ell \;\middle|\; \begin{array}{l} (\textbf{ caused } \ell \textbf{ if } F \textbf{ after } G) \in \mathcal{DC}, \\ s \cup A \models G, s \models F \end{array} \right\}$$

Let us consider a domain $D = \langle \mathcal{SC}, \mathcal{DC}, \mathcal{IN} \rangle$, where $\mathcal{SC}$ are the static causal laws, $\mathcal{DC}$ are the dynamic causal laws and $\mathcal{IN}$ are the non-inertial axioms. Let us also denote with if the set $if = \{f, \neg f \mid f \in \mathcal{IN} \ or \ \neg f \in \mathcal{IN}\}$. The semantics of D is given by a transition system (N_D, E_D), where N_D is the set of all states and the transitions in E_D are of the form $\langle s, A, s' \rangle$, where s, s' are states, $A \subseteq \mathcal{A}$, and s' satisfies the property

$$s' = Cl_{\mathcal{SC}}(Eff_{\mathcal{DC}}(s, A) \cup ((s \setminus if) \cap s') \cup (\mathcal{IN} \cap s')).$$

The original $\mathcal{C}$ language supports a query language (called $\mathcal{P}$ in [Gelfond and Lifschitz, 1998]). This language allows queries of the form **necessarily** F **after** $A_1, \ldots, A_k$ where F is a state formula and $A_1, \ldots, A_k$ are sets of actions (called a *plan*). Intuitively, the query asks whether each state s reached after executing $A_1, \ldots, A_k$ from the initial state has the property $s \models F$.

Let us denote with $State_D$ the set of all possible states for the domain D. Formally, an initial state s_0 w.r.t. an initial state description $\mathcal{I}$ and a domain D is an element of $State_D$ such that $\{\ell \mid$ **initially** $\ell \in \mathcal{I}\} \subseteq s_0$. The transition function $\Phi_D : 2^{\mathcal{A}} \times State_D \rightarrow 2^{State_D}$ is defined as

$$\Phi_D(A, s) = \{s' \mid \langle s, A, s' \rangle \in E_D\}$$

where (N_D, E_D) is the transition system describing the semantics of D. This function can be extended to define Φ_D^* which considers sequences of sets of actions (i.e., plans), where $\Phi_D^*([\,], s) = \{s\}$ and

$$\Phi_D^*([A_1, \ldots, A_n], s) = \bigcup_{s' \in \Phi_D^*([A_1, \ldots, A_{n-1}], s)} \Phi_D(A_n, s').$$

Let us consider an action domain D and an initial state description $\mathcal{I}$. A query **necessarily** F **after** $A_1, \ldots, A_k$ is *entailed* by $(D, \mathcal{I})$, denoted by

$$(D, \mathcal{I}) \models \textbf{necessarily } F \textbf{ after } A_1, \ldots, A_k$$

if for every s_0 initial state w.r.t. $\mathcal{I}$, we have that $\Phi_D^*([A_1, \ldots, A_k], s_0) \neq \emptyset$, and for each $s \in \Phi_D^*([A_1, \ldots, A_k], s_0)$ we have that $s \models F$.

3 $\mathcal{C}$ for Multi-agent Domains

In this section, we will discuss a number of small modifications of $\mathcal{C}$ necessary to enable modeling MAS domains. We will describe each domain from the perspective of someone (the modeler) who has knowledge of everything, including the capabilities and knowledge of each agent. Note that this is *only a modeling perspective*—it does not mean that we expect agents to have knowledge of everything, we only expect the *modeler* to have such knowledge.

We uniquely associate to each agent an element from a set of *agent identifiers*, $\mathcal{AG}$. We will describe a MAS domain over a set of signatures $\langle \mathcal{F}_i, \mathcal{A}_i \rangle$ for each $i \in \mathcal{AG}$, with the assumption that $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ for $i \neq j$. Observe that $\bigcap_{i \in S} \mathcal{F}_i$ could be not empty for some $S \subseteq \mathcal{AG}$. This represents common knowledge between the agents in the group S of agents.

The result is a $\mathcal{C}$ domain over the signature $\langle \bigcup_{i=1}^n \mathcal{F}_i, \bigcup_{i=1}^n \mathcal{A}_i \rangle$. We will require the following condition to be met: if **caused** ℓ **if** F **after** G is a dynamic law and $a \in \mathcal{A}_i$ appears in G, then the literal ℓ belongs to $\mathcal{F}_i$. This condition summarizes the fact that agents are aware of the direct effects of their actions.

We will now illustrate the use of $\mathcal{C}$ in modeling various examples from the literature.

3.1 The Prison Domain

This domain has been originally presented in [Sauro *et al.*, 2006]. In this example, we have two prison guards, 1 and 2, who control two gates, the inner gate and the outer gate, by operating the four buttons a_1, b_1, a_2, and b_2. Agent 1 controls a_1 and b_1 and agent 2 controls a_2 and b_2. If either a_1 or a_2 is pressed, then the state of the inner gate is toggled. The outer gate, on the other hand, toggles only if both b_1 and b_2 are pressed. In $\mathcal{C}$, this domain can be represented as follows.

The set of agents is $\mathcal{AG} = \{1, 2\}$. For agent 1, we have that

$$\mathcal{F}_1 = \{in\_open, out\_open, pressed(a_1), pressed(b_1)\}.$$

Here, in_open and out_open represent the fact that the inner gate and outer gate are open, respectively. The fluent $pressed(X)$ indicates that the button X is pressed, where $X \in \{a_1, b_1\}$. We have $\mathcal{A}_1 = \{push(a_1), push(b_1)\}$. This indicates that guard 1 can push buttons a_1 and b_1. Similarly, for agent 2, we have that

$$\begin{array}{lcl} \mathcal{F}_2 & = & \{in\_open, out\_open, pressed(a_2), pressed(b_2)\} \\ \mathcal{A}_2 & = & \{push(a_2), push(b_2)\} \end{array}$$

We assume that the buttons do not stay pressed—thus, $pressed(X)$, for $X \in \{a_1, b_1, a_2, b_2\}$, is a non-inertial fluent with the default value *false*.

The domain specification (D_{prison}) contains:

non_inertial $\neg pressed(X)$
caused $pressed(X)$ **after** $push(X)$
caused in_open **if** $pressed(a_1), \neg in\_open$
caused in_open **if** $pressed(a_2), \neg in\_open$
caused $\neg in\_open$ **if** $pressed(a_1), in\_open$
caused $\neg in\_open$ **if** $pressed(a_2), in\_open$
caused out_open **if** $pressed(b_1), pressed(b_2), \neg out\_open$
caused $\neg out\_open$ **if** $pressed(b_1), pressed(b_2), out\_open$

where $X \in \{a_1, b_1, a_2, b_2\}$. The first statement declares that $pressed(X)$ is non-inertial and has *false* as its default value. The second statement describes the effect of the action $push(X)$. The remaining laws are static causal laws describing relationships between properties of the environment.

Let us now consider the queries that were asked in [Sauro *et al.*, 2006] and see how they can be answered by using the domain specification D_{prison}. In the first situation, both gates are closed, 1 presses a_1 and b_1, and 2 presses b_2. The question is whether the gates are open or not after the execution of these actions

The initial situation is specified by the initial state description $\mathcal{I}_1$ containing

$$\mathcal{I}_1 = \{ \ \textbf{initially} \ \neg in\_open, \quad \textbf{initially} \ \neg out\_open \ \}$$

In this situation, there is only one initial state $s_0 = \{\neg \ell \mid \ell \in \mathcal{F}_1 \cup \mathcal{F}_2\}$. We can show that

$$(D_{prison}, \mathcal{I}_1) \models \textbf{necessarily } out\_open \wedge in\_open \\ \textbf{after } \{push(a_1), push(b_1), push(b_2)\}$$

On the other hand, if the outer gate is initially closed, i.e., $\mathcal{I}_2 = \{$ **initially** $\neg out\_open\}$, then the set of actions $A = \{push(b_1), push(b_2)\}$ is both necessary and sufficient to open it:

$$(D_{prison}, \mathcal{I}_2) \models \textbf{ necessarily } out\_open \textbf{ after } X$$
$$(D_{prison}, \mathcal{I}_2) \models \textbf{ necessarily } \neg out\_open \textbf{ after } Y$$

where $A \subseteq X$ and $A \setminus Y \neq \emptyset$.

3.2 The Credit Rating Domain

We will next consider an example from [Gerbrandy, 2006] in which an effect on certain properties of the world cannot be changed by a single agent.

We have two agents, $\mathcal{AG} = \{w, t\}$, denoting the website and the telephone operator respectively. Both agents can set/reset the credit of a customer. The credit rating can only be set to be ok (i.e., the fluent $credit\_ok$ set to *true*) if both agents agree. Whether the customer is a webcustomer (is_web fluent) or not is set only by the website agent w.

The signatures of the two agents are as follows:

$$\begin{array}{lcl} \mathcal{F}_w & = & \{is\_web, credit\_ok\} \\ \mathcal{A}_w & = & \{set\_web, reset\_web, set\_credit(w), reset\_credit(w)\} \\ \mathcal{F}_t & = & \{credit\_ok\} \\ \mathcal{A}_t & = & \{set\_credit(t), reset\_credit(t)\} \end{array}$$

The domain specification D_{bank} consists of:

caused is_web **after** set_web
caused $\neg is\_web$ **after** $reset\_web$
caused $credit\_ok$ **after** $set\_credit(w) \wedge set\_credit(t)$
caused $\neg credit\_ok$ **after** $reset\_credit(w)$
caused $\neg credit\_ok$ **after** $reset\_credit(t)$

We can show that

$$(D_{bank}, \mathcal{I}_3) \models \textbf{ necessarily } credit\_ok \textbf{ after } \{set\_credit(w), set\_credit(t)\}$$

where $\mathcal{I}_3 = \{$ **initially** $\neg \ell \mid \ell \in \mathcal{F}_w \cup \mathcal{F}_t\}$. This entailment also holds if $\mathcal{I}_3 = \emptyset$.

4 Adding Priority between Actions

The previous examples show that $\mathcal{C}$ is sufficiently expressive to model the basic aspects of agents executing actions within a MAS, focusing on agents' capabilities and actions interaction. This is in itself not a big surprise and has been discussed by [Boutilier and Brafman, 2001]. We will now present a small extension of $\mathcal{C}$ that facilitates strategic reasoning. For each domain specification D, we assume the presence of a function $Pr_D : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$. Intuitively, $Pr_D(A)$ denotes the set of actions whose effects will be accounted for when A is executed. This function allows, for example, to prioritize certain sets of actions. The new transition function $\Phi_{D,P}$ will be modified as follows:

$$\Phi_{D,P}(A, s) = \Phi_D(Pr_D(A), s)$$

where Φ_D is defined as in the previous section.

4.1 The Rocket Domain

This domain was originally proposed in [van der Hoek *et al.*, 2005]. We have a rocket, a cargo, and three agents 1, 2, and 3. The rocket or the cargo are either in $london$ or $paris$. The rocket can be moved by 1 and 2 between the two locations. The cargo can be loaded (resp. unloaded) into the rocket by 1 and 3 (resp. 2 and 3). Agent 3 can refill the rocket if the tank is not full.

We will use the fluents $rocket(london)$ and $rocket(paris)$ to denote the location of the rocket. Likewise, $cargo(london)$ and $cargo(paris)$ denote the location of the cargo. The fluent in_rocket says that the cargo is inside the rocket and $tank\_full$ states that the tank is full. We will also have a non-inertial fluent literal, $moving$, denoting that the rocket is in the state of moving. The signatures for the agents can be defined as follows.

$$\begin{array}{lcl} \mathcal{F}_1 & = & \left\{ \begin{array}{l} in\_rocket, rocket(london), rocket(paris), \\ cargo(london), cargo(paris) \end{array} \right\} \\ \mathcal{A}_1 & = & \{\ load(1), unload(1), move(1)\ \} \\ \mathcal{F}_2 & = & \left\{ \begin{array}{l} in\_rocket, rocket(london), rocket(paris), \\ cargo(london), cargo(paris) \end{array} \right\} \\ \mathcal{A}_2 & = & \{\ unload(2), move(2)\ \} \\ \mathcal{F}_3 & = & \left\{ \begin{array}{l} in\_rocket, rocket(london), rocket(paris), \\ cargo(london), cargo(paris), tank\_full \end{array} \right\} \\ \mathcal{A}_3 & = & \{\ load(3), refill\ \} \end{array}$$

This domain has a special feature—there are priorities among actions. The domain states that $load$ or $unload$ will have no effect if $move$ is executed. The effects of two $load$ actions is the same as that of a single $load$ action. Likewise, two $unload$ actions have the same result as one action.

To account for the priority of actions, we define Pr_D as follows:

- $Pr_D(X) = \{move(a)\}$ if $\exists a.\ move(a) \in X$.
- $Pr_D(X) = \{load(a)\}$ if $move(x) \notin X$ for every $x \in \{1, 2, 3\}$ and $load(a) \in X$.
- $Pr_D(X) = \{unload(a)\}$ if $move(x) \notin X$ and $load(x) \notin X$ for every $x \in \{1, 2, 3\}$ and $unload(a) \in X$.
- $Pr_D(X) = X$ otherwise.

It is easy to see that Pr_D defines priorities among the actions: if the rocket is moving then load/unload are ignored; load has higher priority than unload; etc. The domain specification consists of the following laws:

caused in_rocket **after** $load(i)$ $\quad (i \in \{1, 3\})$
caused $\neg in\_rocket$ **after** $unload(i)$ $\quad (i \in \{1, 2\})$
caused $tank\_full$ **if** $\neg tank\_full$ **after** $refill$
caused $\neg tank\_full$ **if** $tank\_full$
after $move(i)$ $\quad (i \in \{1, 2\})$
caused $rocket(london)$ **if** $rocket(paris), tank\_full$
after $move(i)$ $\quad (i \in \{1, 2\})$
caused $rocket(paris)$ **if** $rocket(london), tank\_full$
after $move(i)$ $\quad (i \in \{1, 2\})$
caused $cargo(paris)$ **if** $rocket(paris), in\_rocket$
caused $cargo(london)$ **if** $rocket(london), in\_rocket$
non_inertial $\neg moving$

Let $\mathcal{I}_4$ consist of the following facts:

initially $tank\_full$	**initially** $rocket(paris)$
initially $cargo(london)$	**initially** $\neg in\_rocket$

We can show the following

$$(D_{rocket}, \mathcal{I}_4) \models \textbf{necessarily } cargo(paris) \\ \textbf{after } \{move(1)\}, \{load(3)\}, \{refill\}, \{move(3)\}$$

5 Reasoning with Agent Knowledge

In this section, we will consider some examples from [Spaan *et al.*, 2006; Herzig and Troquard, 2006] which address another aspect of modeling MAS, i.e., the exchange of knowledge between agents and the reasoning in presence of incomplete knowledge. We will show that using $\mathcal{C}$, we can model this aspect without the introduction of additional features.

5.1 Heaven and Hell Domain

This example has been drawn from [Spaan *et al.*, 2006]. This example has two agents 1 and 2, a priest p, and three rooms r_1, r_2, r_3. Each of the two rooms r_2 and r_3 is either heaven or hell. If r_2 is heaven then r_3 is hell and vice versa. The priest has knowledge of where the heaven and hell are located. Neither 1 nor 2 know where heaven/hell is, but, by visiting the priest, they can receive the information that tells them where heaven is. 1 and 2 want to meet in heaven.

The signatures for the three agents are as follows ($k, h \in \{1,2,3\}$):

$$\begin{array}{lcl} \mathcal{F}_1 & = & \{heaven_1^2, heaven_1^3, hell_1^2, hell_1^3, at_1^k\} \\ \mathcal{A}_1 & = & \{m_1(k,h), ask_1\} \\ \mathcal{F}_2 & = & \{heaven_2^2, heaven_2^3, hell_2^2, hell_2^3, at_2^k\} \\ \mathcal{A}_2 & = & \{m_2(k,h), ask_2\} \\ \mathcal{F}_p & = & \{heaven_p^2, heaven_p^3, hell_p^2, hell_p^3\} \\ \mathcal{A}_p & = & \emptyset \end{array}$$

The domain specification D_{heav} contains the following laws:

$$\begin{array}{ll} \textbf{caused } heaven_1^j \textbf{ if } heaven_p^j \textbf{ after } ask_1 & (j \in \{2,3\}) \\ \textbf{caused } heaven_2^j \textbf{ if } heaven_p^j \textbf{ after } ask_2 & (j \in \{2,3\}) \\ \textbf{caused } at_i^j \textbf{ if } at_i^k \textbf{ after } m_i(k,j) (i \in \{1,2,p\}, j,k \in \{1,2,3\}) & \\ \textbf{caused } \neg at_i^j \textbf{ if } at_i^k & (i \in \{1,2,p\}, j,k \in \{1,2,3\}, j \neq k) \\ \textbf{caused } hell_i^j \textbf{ if } heaven_i^k & (i \in \{1,2,p\}, j,k \in \{2,3\}, j \neq k) \\ \textbf{caused } heaven_i^j \textbf{ if } hell_i^k & (i \in \{1,2,p\}, j,k \in \{2,3\}, j \neq k) \\ \textbf{caused } \neg hell_i^j \textbf{ if } heaven_i^j & (i \in \{1,2,p\}, j \in \{2,3\}) \\ \textbf{caused } \neg heaven_i^j \textbf{ if } hell_i^j & (i \in \{1,2,p\}, j \in \{2,3\}) \end{array}$$

Let us consider an instance that has initial state described by $\mathcal{I}_5$ ($j \in \{2,3\}$):

$$\begin{array}{lll} \textbf{initially } at_1^1 & \textbf{initially } at_2^2 & \textbf{initially } heaven_p^2 \\ \textbf{initially } \neg heaven_1^j & \textbf{initially } \neg hell_1^j & \textbf{initially } \neg heaven_2^j \\ \textbf{initially } \neg hell_2^j & & \end{array}$$

We can show that

$$(D_{heav}, \mathcal{I}_5) \models \textbf{necessarily } at_1^2 \wedge heaven_1^2 \textbf{ after } \{ask_1\}, \{m_1(1,2)\}.$$

5.2 Blind Agents & Lamp Domain

This next example is drawn from [Herzig and Troquard, 2006]. There are two blind agents and two switches; the light is on only when both switches are in the same position. Agent 1 can toggle the first switch and agent 2 the second one.

$$\begin{array}{lcl} \mathcal{F}_1 & = & \{switch\_on_1, switch\_off_1, on\} \\ \mathcal{A}_1 & = & \{t_1\} \\ \mathcal{F}_2 & = & \{switch\_on_2, switch\_off_2, on\} \\ \mathcal{A}_2 & = & \{t_2\} \end{array}$$

Intuitively, $switch\_on_i$ (resp. $switch\_off_i$) represents the fact that i knows that switch i is at the on (resp. off) position. The domain D_{blind} is composed of the laws:

$$\begin{array}{l} \textbf{caused } \neg switch\_on_i \textbf{ if } switch\_off_i \\ \textbf{caused } \neg switch\_off_i \textbf{ if } switch\_on_i \\ \textbf{caused } on \textbf{ if } switch\_on_1 \wedge switch\_on_2 \\ \textbf{caused } on \textbf{ if } switch\_off_1 \wedge switch\_off_2 \\ \textbf{caused } switch\_on_i \textbf{ if } switch\_off_i \textbf{ after } t_i \\ \textbf{caused } switch\_off_i \textbf{ if } switch\_on_i \textbf{ after } t_i \end{array}$$

Let $\mathcal{I} = \{$ **initially** $switch\_on_1$, **initially** $switch\_off_2$, **initially** $\neg on\}$). We can show that

$$(D_{blind}, \mathcal{I}) \models \textbf{necessarily } on \textbf{ after } t_2$$

On the other hand, if $\mathcal{I} = \{$ **initially** $\neg on\}$, we can show that there is no plan α (i.e., a *conformant plan*) such that

$$(D_{blind}, \mathcal{I}) \not\models \textbf{necessarily } on \textbf{ after } \alpha$$

6 Adding Reward Strategies

The next example illustrates the need to handle numbers and optimization to represent reward mechanisms. The extension of $\mathcal{C}$ is simply the introduction of *numerical fluents*—i.e., fluents that, instead of being simply true or false, have a numeric value. For this purpose, we introduce a new variant of the necessity query

$$\textbf{necessarily max } F \textbf{ for } \varphi \textbf{ after } A_1, \dots, A_n$$

where F is a numerical expressions involving only numerical fluents, φ is a state formula, and $A_1, \dots, A_n$ is a plan. Given a domain specification D and an initial state description $\mathcal{I}$, we can define for each fluent numerical expression F and plan α:

$$value(F, \alpha) = \max \left\{ s(F) \middle| \begin{array}{l} s \in \Phi^*(\alpha, s_0), \\ s_0 \textit{ is an initial state w.r.t. } \mathcal{I}, D \end{array} \right\}$$

where $s(F)$ denotes the value of the expression F in state s. This allows us to define the following notion of entailment of a query:

$$(D, \mathcal{I}) \models \textbf{necessarily max } F \textbf{ for } \varphi \textbf{ after } A_1, \dots, A_n$$

if:

- ◦ $(D, \mathcal{I}) \models$ **necessarily** φ **after** $A_1, \dots, A_n$
- ◦ for every other plan $B_1, \dots, B_m$ such that

$$(D, \mathcal{I}) \models \textbf{necessarily } \varphi \textbf{ after } B_1, \dots, B_m$$

we have that

$$value(F, [A_1, \dots, A_n]) \geq value(F, [B_1, \dots, B_m]).$$

6.1 Social Laws Domain

The following example has been derived from [Boella and van der Torre, 2005]. There are three agents. Agent 0 is a normative system that can play one of two strategies—either st_0 or $\neg st_0$. Agent 1 plays a strategy st_1, while agent 2 plays the strategy st_2. The reward system is described in the following tables (the first is for st_0 and the second one is for $\neg st_0$).

st_0	st_1	$\neg st_1$
st_2	$1,1$	$0,0$
$\neg st_2$	$0,0$	$-1,-1$

$\neg st_0$	st_1	$\neg st_1$
st_2	$1,1$	$0,0$
$\neg st_2$	$0,0$	$1,1$

The signatures used by the agents are

$$\begin{array}{lll}
\mathcal{F}_0 & = & \{st_0, reward\} \\
\mathcal{A}_0 & = & \{play\_0, play\_not\_0\} \\
\mathcal{F}_1 & = & \{st_1, reward_1\} \\
\mathcal{A}_1 & & \{play\_1, play\_not\_1\} \\
\mathcal{F}_2 & = & \{st_2, reward_2\} \\
\mathcal{A}_2 & = & \{play\_2, play\_not\_2\}
\end{array}$$

The domain specification D_{gam} consists of:

caused st_0 **after** $play\_0$
caused $\neg st_0$ **after** $play\_not\_0$
caused st_1 **after** $play\_1$
caused $\neg st_1$ **after** $play\_not\_1$
caused st_2 **after** $play\_2$
caused $\neg st_2$ **after** $play\_not\_2$
caused $reward\_1 = 1$ **if** $\neg st_0 \wedge st_1 \wedge st_2$
caused $reward\_2 = 1$ **if** $\neg st_0 \wedge st_1 \wedge st_2$
caused $reward\_1 = 0$ **if** $\neg st_0 \wedge st_1 \wedge \neg st_2$
caused $reward\_2 = 0$ **if** $\neg st_0 \wedge st_1 \wedge \neg st_2$
...
caused $reward = a + b$ **if** $reward_1 = a \wedge reward_2 = b$

Assuming that $\mathcal{I} = \{$ **initially** $st_0\}$ we can show that

$(D_{game}, \mathcal{I}) \models$ **necessarily max** $reward$ **after** $\{play_1, play_2\}$

7 Reasoning and Properties

In this section we discuss various types of reasoning that are directly enabled by the semantics of $\mathcal{C}$.

7.1 Capability Queries

Let us explore another range of queries, that are aimed at capturing the capabilities of agents. We will use the generic form **can** X **do** φ, where φ is a state formula and $X \subseteq \mathcal{AG}$. The intuition is to validate whether the group of agents X can guarantee that φ is satisfied.

If $X = \mathcal{AG}$ then the semantics of the capability query is simply expressed as $(D, \mathcal{I}) \models$ **can** X **do** φ iff $\exists k.\ \exists A_1, \dots, A_k$ such that

$(D, \mathcal{I}) \models$ **necessarily** φ **after** $A_1, \dots, A_k$.

If $X \neq \mathcal{AG}$, then we can envision different variants of this query.

Capability query with non-interference and complete knowledge: Intuitively, the goal is to verify whether the agents X can achieve φ when operating in an environment that includes *all* the agents, but the agents $\mathcal{AG} \setminus X$ are simply providing their knowledge and not performing actions or interfering. We will denote this type of queries as **can**^n_g X **do** φ (n: not interference, g: availability of all knowledge).

The semantics of this type of queries can be formalized as follows: $(D, \mathcal{I}) \models$ **can**^n_g X **do** φ if there is a sequence of sets of actions $A_1, \dots, A_k$ with the following properties:

- ○ for each $1 \leq i \leq k$ we have that $A_i \subseteq \bigcup_{j \in X} \mathcal{A}_j$ (we perform only actions of agents in X)
- ○ $(D, \mathcal{I}) \models$ **necessarily** φ **after** $A_1, \dots, A_k$

Capability query with non-interference and projected knowledge: Intuitively, the query with projected knowledge assumes that not only the other agents ($\mathcal{AG} \setminus X$) are passive, but they also are not willing to provide knowledge to the active agents. We will denote this type of queries as **can**^n_l X **do** φ.

Let us refer to the *projection* of $\mathcal{I}$ w.r.t. X (denoted by $proj(\mathcal{I}, X)$) as the set of all the **initially** declarations that build on fluents of $\bigcup_{j \in X} \mathcal{F}_j$. The semantics of **can**^n_l type of queries can be formalized as follows: $(D, \mathcal{I}) \models$ **can**^n_l X **do** φ if there is a sequence of sets of actions $A_1, \dots, A_k$ such that:

- for each $1 \leq i \leq k$ we have that $A_i \subseteq \bigcup_{j \in X} \mathcal{A}_j$
- $(D, proj(\mathcal{I}, X)) \models$ **necessarily** φ **after** $A_1, \dots, A_k$ (i.e., the objective will be reached irrespective of the initial configuration of the other agents)

Capability query with interference: The final version of capability query takes into account the possible interference from other agents in the system. Intuitively, the query with interference, denoted by **can**^{i} X **do** φ, implies that the agents X will be able to accomplish X in spite of other actions performed by the other agents.

The semantics is as follows: $(D, \mathcal{I}) \models$ **can**^{i} X **do** φ if there is a sequence of sets of actions $A_1, \dots, A_k$ such that:

- for each $1 \leq i \leq k$ we have that $A_i \subseteq \bigcup_{j \in X} \mathcal{A}_j$
- for each sequence of sets of actions $B_1, \dots, B_k$, where $\bigcup_{j=1}^k B_j \subseteq \bigcup_{j \notin X} \mathcal{A}_j$, we have that
  $(D, \mathcal{I}) \models$ **necessarily** φ **after** $(A_1 \cup B_1), \dots, (A_k \cup B_k)$

7.2 Inferring Properties of the Theory

The form of queries explored above allows us to investigate some basic properties of a multi-agent action domain.

Agent Redundancy: agent redundancy is a property of $(D, \mathcal{I})$ which indicates the ability to remove an agent in accomplishing a goal. Formally, agent i is redundant w.r.t. a state formula φ and an initial state $\mathcal{I}$ if

$(D, \mathcal{I}) \models$ **can** $X \setminus \{i\}$ **do** φ.

The "level" of necessity can be refined, by adopting different levels of **can** (e.g., **can**^n_l implies that the knowledge of agent i is not required); it is also possible to strengthen it by enabling the condition to be satisfied for *any* $\mathcal{I}$.

Agent Necessity: agent necessity is symmetrical to redundancy—it denotes the inability to accomplish a property φ if an agent is excluded. Agent i is *necessary* w.r.t. φ and $(D, \mathcal{I})$ if, for all sequences of sets of actions $A_1, \ldots, A_k$, such that $A_j \cap \mathcal{A}_i = \emptyset$ for all $1 \leq j \leq k$, we have that it is not the case that $(D, \mathcal{I}) \models$ **necessarily** φ **after** $A_1, \ldots, A_k$.

We can also define different degrees of necessity, depending on whether the knowledge of i is available (or it should be removed from $\mathcal{I}$) and whether i is allowed to interfere.

7.3 Compositionality

The formalization of multi-agent systems in $\mathcal{C}$ enables exploring the effects of composing domains; this is an important property, that allows us to model dynamic MAS systems (e.g., where new agents can join an existing coalition).

Let D_1, D_2 be two domains and let us indicate with $\langle \mathcal{F}_i^1, \mathcal{A}_i^1 \rangle_{i \in \mathcal{AG}_1}$ and $\langle \mathcal{F}_i^2, \mathcal{A}_i^2 \rangle_{i \in \mathcal{AG}_2}$ the agent signatures of D_1 and D_2. We assume that all actions sets are disjoint, while we allow $(\bigcup_{i \in \mathcal{AG}_1} \mathcal{F}_i^1) \cap (\bigcup_{i \in \mathcal{AG}_2} \mathcal{F}_i^2) \neq \emptyset$.

We define the two instances $(D_1, \mathcal{I}_1)$ and $(D_2, \mathcal{I}_2)$ to be *composable* w.r.t. a state formula φ if $(D_1, \mathcal{I}_1) \models$ **can** $\mathcal{AG}_1$ **do** φ or $(D_2, \mathcal{I}_2) \models$ **can** $\mathcal{AG}_2$ **do** φ implies

$$(D_1 \cup D_2, \mathcal{I}_1 \cup \mathcal{I}_2) \models \textbf{can } \mathcal{AG}_1 \cup \mathcal{AG}_2 \textbf{ do } \varphi$$

Two instances are composable if they are composable w.r.t. all state formulae φ. Two domains D_1, D_2 are composable if all the instances $(D_1, \mathcal{I}_1)$ and $(D_2, \mathcal{I}_2)$ are composable.

8 Discussion

This section discusses an aspect of modeling MAS that cannot be easily dealt with in $\mathcal{C}$, i.e., representing and reasoning about knowledge of agents. In the domains 5.1 and 5.2, we use two different fluents to model the knowledge of an agent about properties of the world, similar to the approach in [Palacios and Geffner, 2007; Son and Baral, 2001]. This approach is adequate for several situations. Nevertheless, the same approach could become quite cumbersome if complex reasoning about knowledge of other agents is involved.

Example 1 (Muddy Children, [Fagin *et al.*, 1995])**.** Two children are playing outside the house. Their father comes and tells them that at least one of them has mud on his/her forehead. He then repeatedly asks "do you know whether your forehead is muddy or not?". The first time, both answer "no" and the second time, both say 'yes'. It is known that the father and the children can see and hear each other.

The representation of this domain in $\mathcal{C}$ is possible, but it would require a large number of fluents (that describe the knowledge of each child, the knowledge of each child about the other child, etc.) as well as a formalization of the axioms necessary to express how knowledge should be manipulated.

A more effective approach is to introduce explicit knowledge operators (with manipulation axioms implicit in their semantics—e.g., as operators in a S5 modal logic) and use them to describe agents state. Let us consider a set of modal operators $\mathbf{K}_i$, one for each agent. A formula such as $\mathbf{K}_i\varphi$ denotes that agent i knows property φ. Knowledge operators can be nested; in particular, $\mathbf{K}^*{}_G\psi$ denotes all formulae with arbitrary nesting of $\mathbf{K}_G$ operators (G being a set of agents).

In our example, let us denote the children with 1 and 2. We use m_i as a fluent denoting whether i is muddy or not. The initial state of the world can then be described as follows:

$$\textbf{initially } m_1 \wedge m_2 \quad (1)$$

$$\textbf{initially } \neg\mathbf{K}_i m_i \wedge \neg\mathbf{K}_i \neg m_i \quad (2)$$

$$\textbf{initially } \mathbf{K}^*(m_1 \vee m_2) \quad (3)$$

$$\textbf{initially } \mathbf{K}^*{}_{\{1,2\}\setminus\{i\}} m_i \quad (4)$$

$$\textbf{initially } \mathbf{K}^*(\mathbf{K}^*{}_{\{1,2\}\setminus\{i\}} m_i \vee \mathbf{K}^*{}_{\{1,2\}\setminus\{i\}} \neg m_i) \quad (5)$$

where $i \in \{1, 2\}$. (1) states that all the children are muddy. (2) says that i does not know whether he/she is muddy. (3) encodes the fact that the children share the common knowledge that at least one of them is muddy. (4) captures the fact that each child can see the other child. Finally, (5) represents the common knowledge that each child knows the muddy status of the other one.

The actions used in this domain would enable agents to gain knowledge; e.g., the 'no' answer of child 1 allows child 2 to learn $\mathbf{K}_1(\neg\mathbf{K}_1 m_1 \wedge \neg\mathbf{K}_1 \neg m_1)$. This, together with the initial knowledge, would be sufficient for 2 to conclude $\mathbf{K}_2 m_2$.

9 Conclusion

In this paper, we presented an investigation of the use of the $\mathcal{C}$ action language to model MAS domains. $\mathcal{C}$, as several other action languages, is interesting as it provides well studied foundations for knowledge representation and for performing several types of reasoning tasks. Furthermore, the literature provides a rich infrastructure for the implementation of action languages (e.g., through translational techniques [Son *et al.*, 2006]). The results presented in this paper identify several interesting features that are necessary for modeling MAS, and they show how such features can be encoded in $\mathcal{C}$—either directly or with simple extensions of the action language. We also report on different forms of reasoning that are naturally supported by the proposed language.

The natural next steps in this line of work consist of *(1)* exploring how more complex domains can be captured, especially domains requiring complex reasoning about knowledge of other agents (as discussed in Sect. 8); *(2)* adapting the more advanced forms of reasoning and implementation proposed for $\mathcal{C}$ to the case of MAS domains; *(3)* investigating the use of the proposed extension of $\mathcal{C}$ in formalizing distributed systems.

References

[Baker, 1989] A. Baker. A simple solution to the Yale Shooting Problem. In R. Brachman, H. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 11–20. Morgan Kaufmann, 1989.

[Boella and van der Torre, 2005] Guido Boella and Leendert W. N. van der Torre. Enforceable social laws. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *4rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, pages 682–689. ACM, 2005.

[Boutilier and Brafman, 2001] Craig Boutilier and Ronen I. Brafman. Partial-order planning with concurrent interacting actions. *J. Artif. Intell. Res. (JAIR)*, 14:105–136, 2001.

[Brenner, 2005] M. Brenner. Planning for Multiagent Environments: From Individual Perceptions to Coordinated Execution. In *Proceedings of Workshop on Multiagent Planning and Scheduling, ICAPS*, pages 80–88. 2005.

[Fagin *et al.*, 1995] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT press, 1995.

[Gelfond and Lifschitz, 1993] M. Gelfond and V. Lifschitz. Representing actions and change by logic programs. *Journal of Logic Programming*, 17(2,3,4):301–323, 1993.

[Gelfond and Lifschitz, 1998] M. Gelfond and V. Lifschitz. Action languages. *ETAI*, 3(6), 1998.

[Gerbrandy, 2006] Jelle Gerbrandy. Logics of propositional control. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 193–200. ACM, 2006.

[Hanks and McDermott, 1987] S. Hanks and D. McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, 1987.

[Herzig and Troquard, 2006] Andreas Herzig and Nicolas Troquard. Knowing how to play: uniform choices in logics of agency. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 209–216, 2006.

[Kautz, 1986] H. Kautz. The logic of persistence. In *Proceedings of AAAI-86*, pages 401–405, 1986.

[McCarthy and Hayes, 1969] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[Palacios and Geffner, 2007] H. Palacios and H. Geffner. From Conformant into Classical Planning: Efficient Translations that may be Complete Too. In *Proceedings of the 17th International Conference on Planning and Scheduling*, 2007.

[Sauro *et al.*, 2006] Luigi Sauro, Jelle Gerbrandy, Wiebe van der Hoek, and Michael Wooldridge. Reasoning about action and cooperation. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 185–192, New York, NY, USA, 2006. ACM.

[Son and Baral, 2001] T.C. Son and C. Baral. Formalizing sensing actions - a transition function based approach. *Artificial Intelligence*, 125(1-2):19–91, January 2001.

[Son *et al.*, 2006] Tran Cao Son, Chitta Baral, Nam Tran, and Sheila McIlraith. Domain-dependent knowledge in answer set planning. *ACM Trans. Comput. Logic*, 7(4):613–657, 2006.

[Spaan *et al.*, 2006] Matthijs T. J. Spaan, Geoffrey J. Gordon, and Nikos A. Vlassis. Decentralized planning under uncertainty for teams of communicating agents. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 249–256, 2006.

[van der Hoek *et al.*, 2005] Wiebe van der Hoek, Wojciech Jamroga, and Michael Wooldridge. A logic for strategic reasoning. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *4rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, pages 157–164. ACM, 2005.

A General Family of Preferential Belief Removal Operators

Richard Booth
Mahasarakham University
Faculty of Informatics
Mahasarakham 44150, Thailand
richard.b@msu.ac.th

Thomas Meyer
Meraka Institute, CSIR and
School of Computer Science
University of Kwazulu-Natal
South Africa
tommie.meyer@meraka.org.za

Chattrakul Sombattheera
Mahasarakham University
Faculty of Informatics
Mahasarakham 44150, Thailand
chattrakul.s@msu.ac.th

Abstract

Most belief change operators in the AGM tradition assume an underlying plausibility ordering over the possible worlds which is transitive and *complete*. A unifying structure for these operators, based on supplementing the plausibility ordering with a second, guiding, relation over the worlds was presented in [Booth *et al.*, 2004]. However it is not always reasonable to assume completeness of the underlying ordering. In this paper we generalise the structure of [Booth *et al.*, 2004] to allow incomparabilities between worlds. We axiomatise the resulting class of belief removal functions, and show that it includes an important family of removal functions based on *finite prioritised belief bases*. We also look at some alternative notions of *epistemic entrenchment* which become distinguishable once we allow incomparabilities.

1 Introduction

The problem of *belief removal* [Alchourrón *et al.*, 1985; Booth *et al.*, 2004; Rott and Pagnucco, 1999], i.e., the problem of what an agent, hereafter $\mathcal{A}$, should believe after being directed to remove some sentence from his stock of beliefs, has been well studied in philosophy and in AI over the last 25 years. During that time many different families of removal functions have been studied. A great many of them are based on constructions employing *total preorders* over the set of possible worlds which is meant to stand for some notion $\leq$ of relative *plausibility* [Katsuno and Mendelzon, 1992]. A unifying construction for these families was given in [Booth *et al.*, 2004], in which a general construction was proposed which involved supplementing the relation $\leq$ with a second, guiding, relation $\preceq$ which formed a subset of $\leq$. By varying the conditions on $\preceq$ and its interaction with $\leq$ many of the different families can be captured as instances.

The construction in [Booth *et al.*, 2004] achieves a high level of generality, but one can argue it fails to be general enough in one important respect: the underlying plausibility order $\leq$ is *always* assumed to be a total preorder which by definition implies it is *complete*, i.e., for any two worlds x, y, we have either $x \leq y$ or $y \leq x$. This implies that agent $\mathcal{A}$ is *always* able to decide which of x, y is more plausible. This is not always realistic, and so it seems desirable to study belief removal based on plausibility orderings which allow *incomparabilities*. A little work been done on this [Bochman, 2001; Cantwell, 2003; Katsuno and Mendelzon, 1992; Rott, 1992] but not much. This is in contrast to work in nonmonotonic reasoning (NMR), the research area which is so often referred to as the "other side of the coin" to belief change. In NMR, semantic models based on incomplete orderings are the norm, with work dating back to the seminal papers on *preferential models* of [Kraus *et al.*, 1991; Shoham, 1987]. Our aim in this paper is to relax the completeness assumption from [Booth *et al.*, 2004] and to investigate the resulting, even more general class of removal functions.

The plan of the paper is as follows. In Section 2 we give our generalised definition of the construction from [Booth *et al.*, 2004], which we call *(semi-modular) contexts.* We describe their associated removal functions, as well as mention the characterisation from [Booth *et al.*, 2004]. Then in Section 3 we present an axiomatic characterisation of the family of removal functions generated by semi-modular contexts. In Section 4 we discuss some different notions of epistemic entrenchment which collapse into the same notion for the removals from [Booth *et al.*, 2004], but which differ for the more general family. Then, in Section 5 we mention a couple of further restrictions on contexts, leading to two corresponding extra postulates. In Section 6 we mention an important subfamily of the general family, i.e., those removals which may be generated by a finite prioritised base of *defaults,* before moving on to AGM style removal in Section 7. We conclude in Section 8.

1.1 Preliminaries

We work in a finitely-generated propositional language L. The set of non-tautologous sentences in L is denoted by L_*. The set of propositional worlds/models is W. For any set of sentences $X \subseteq L$, the set of worlds which satisfy every sentence in X is denoted by $[X]$. Classical logical consequence and equivalence are denoted by $\vdash$ and $\equiv$ respectively.

As above, we let $\mathcal{A}$ denote some agent whose beliefs are subject to change. A *belief set* for $\mathcal{A}$ is represented by a single sentence which is meant to stand for all its logical consequences. A *belief removal function* (hereafter just *removal function*) belonging to $\mathcal{A}$ is a unary function $\circledast$ which takes any non-tautologous sentence $\lambda \in L_*$ as input and returns a new belief set $\circledast(\lambda)$ for $\mathcal{A}$ such that $\circledast(\lambda) \nvdash \lambda$. For any removal function $\circledast$ we can always derive an associated belief set. It is just the belief set obtained by removing the contradiction, i.e., $\circledast(\bot)$.

1.2 Orderings

The following definitions about orderings will be useful in what follows. A binary relation R over W is:

- *reflexive* iff $\forall x : xRx$
- *transitive* iff $\forall x, y, z : xRy \;\&\; yRz \rightarrow xRz$
- *complete* iff $\forall x, y : xRy \vee yRx$
- a *preorder* iff it is reflexive and transitive
- a *total preorder* iff it is a complete preorder

The above notions are used generally when talking of "weak" orderings, where xRy is meant to stand for something like "x is *at least as good as* y". However in this paper, following the lead of [Rott, 1992], we will find it more natural to work under a *strict* reading, where xRy denotes "x is *strictly better than* y". In this setting, the following notions will naturally arise. R is:

- *irreflexive* iff $\forall x : \mathrm{not}(xRx)$
- *modular* iff $\forall x, y, z : xRy \rightarrow (xRz \vee zRy)$
- a *strict partial order (spo)* iff it is both irreflexive and transitive
- the *strict part of* another relation R' iff $\forall x, y : xRy \leftrightarrow (xR'y \;\&\; \mathrm{not}(yR'x))$
- the *converse complement* of R' iff $\forall x, y : xRy \leftrightarrow \mathrm{not}\,(yR'x)$

We have that R is a modular spo iff it is the strict part of a total preorder [Maynard-Zhang and Lehmann, 2003]. So in terms of *strict* relations, much of the previous work on belief removal, including [Booth *et al.*, 2004], assumes an underlying strict order which is a modular spo. It is precisely the modularity condition which we want to relax in this paper.

Given any ordering R and $x \in W$, let $\nabla_R(x) = \{z \in W \mid zRx\}$ be the set of all worlds below x in R. Then we may define a new binary relation $\sqsubseteq^R$ from R by setting:

$$x \sqsubseteq^R y \text{ iff } \nabla_R(x) \subseteq \nabla_R(y).$$

That is, $x \sqsubseteq^R y$ iff every element below x in R is also below y in R. It is easy to check that if R is a modular spo then $x \sqsubseteq^R y$ iff not (yRx), i.e., $\sqsubseteq^R$ is just the converse complement of R.

2 Contexts, modular contexts and removals

In this section we set up our generalised definition of a context, show how each such context yields a removal function and vice versa, and recap the main results from [Booth *et al.*, 2004].

2.1 Contexts

We assume our agent $\mathcal{A}$ has in his mind *two* binary relations $(<, \prec)$ over the set W. The relation $<$ is a *strict* plausibility relation which forms the basis for $\mathcal{A}$'s actionable beliefs, i.e., $x < y$ means that, to $\mathcal{A}$'s mind, and on the basis of all available evidence, *world x is strictly more plausible than y*. We assume $<$ is a strict partial order. In addition to this there is a second binary relation $\prec$. This relation is open to several different interpretations, but the one we attach is as follows: $x \prec y$ means *"$\mathcal{A}$ has **an explicit reason** to hold x more plausible than y (or to treat x more favourably than y)"*. We will use $\preceq$ to denote the converse complement of $<$, i.e.,

$$x \preceq y \text{ iff } y \not\prec x.$$

Thus $x \preceq y$ iff $\mathcal{A}$ has no reason to treat y more favourably than x. Note $\preceq$ and $\prec$ are interdefinable, and we find it convenient to switch between them freely.

What are the properties of $\prec$? We assume only two things, at least to begin with: *(i)* an agent can never possess a reason to hold a world strictly more plausible than itself, and *(ii)* an agent does not hold a world x to be more plausible than another world y, i.e., $x < y$, *without* being in possession of some reason for doing so. (Note this latter property lends a certain "foundationalist" flavour to our construction.) All this is formalised in the following definition:

Definition 2.1. A *context* $\mathcal{C}$ is a pair of binary relations $(<, \prec)$ over W such that:

($\mathcal{C}$**1**) $<$ is a strict partial order

($\mathcal{C}$**2**) $\prec$ is irreflexive

($\mathcal{C}$**3**) $< \subseteq \prec$

If $<$ is modular then we call $\mathcal{C}$ a *modular context.*

We will later have grounds for strengthening ($\mathcal{C}$**3**).

How does $\mathcal{A}$ use his context $\mathcal{C}$ to construct a removal function $\circledast_{\mathcal{C}}$? In terms of models, the set $[\circledast_{\mathcal{C}}(\lambda)]$ of models of his new belief set, when removing a sentence λ, *must* include some $\neg\lambda$-worlds. Following the usual practice in belief revision, he should take the most plausible ones according to $<$, i.e., the $<$-minimal ones. But which, if any, of the λ-worlds should be included? The following principle was proposed by [Rott and Pagnucco, 1999]:

> **Principle of Weak Preference**
> If one object is held in equal or higher regard than another, the former should be treated no worse than the latter.

[Rott and Pagnucco, 1999] use this principle to argue that the new set of worlds following removal should contain all worlds x which are not less plausible than a $<$-minimal $\neg\lambda$-world y, i.e., $y \not< x$. We propose to apply a tempered version of this principle using the second ordering $\prec$. We include x if there is *no explicit reason to believe* that y is more plausible than x, i.e., if $y \not\prec x$.

Definition 2.2. **($\circledast$ from $\mathcal{C}$)** Given a context $\mathcal{C}$ we define the removal function $\circledast_{\mathcal{C}}$ by setting, for each $\lambda \in L_*$, $[\circledast_{\mathcal{C}}(\lambda)] = \bigcup\{\nabla_{\preceq}(y) \mid y \in \min_<([\neg\lambda])\}$.

It can be shown that different contexts give rise to different removal functions, i.e., the mapping $\mathcal{C} \mapsto \circledast_{\mathcal{C}}$ is injective.

The case of modular contexts was the one which was studied in detail in [Booth *et al.*, 2004], where it was shown how, by placing various restrictions on the interaction between $<$ and $\prec$, this family captures a wide range of removal operations which have been previously studied, for example both AGM contraction *and* AGM revision [Alchourrón *et al.*, 1985], severe withdrawal [Rott and Pagnucco, 1999], systematic withdrawal [Meyer *et al.*, 2002] and belief liberation [Booth *et al.*, 2005]. For the general family in that paper the following representation result was proved.

Theorem 2.3. *[Booth* et al.*, 2004] Let $\mathcal{C}$ be a modular context. Then $\circledast_{\mathcal{C}}$ satisfies the following rules:*[1]

$(\circledast\mathbf{1})$ $\circledast(\lambda) \nvdash \lambda$

$(\circledast\mathbf{2})$ *If* $\lambda_1 \equiv \lambda_2$ *then* $\circledast(\lambda_1) \equiv \circledast(\lambda_2)$

$(\circledast\mathbf{3})$ *If* $\circledast(\lambda \wedge \chi) \vdash \chi$ *then* $\circledast(\lambda \wedge \chi \wedge \psi) \vdash \chi$

$(\circledast\mathbf{4})$ *If* $\circledast(\lambda \wedge \chi) \vdash \chi$ *then* $\circledast(\lambda \wedge \chi) \vdash \circledast(\lambda)$

$(\circledast\mathbf{5})$ $\circledast(\lambda \wedge \chi) \vdash \circledast(\lambda) \vee \circledast(\chi)$

$(\circledast\mathbf{6})$ *If* $\circledast(\lambda \wedge \chi) \nvdash \lambda$ *then* $\circledast(\lambda) \vdash \circledast(\lambda \wedge \chi)$

Furthermore if $\circledast$ is any removal function satisfying the above 6 rules, there exists a unique modular context $\mathcal{C}$ such that $\circledast = \circledast_{\mathcal{C}}$.

All these rules are familiar from the literature on belief removal. Rule $(\circledast\mathbf{1})$ is the Success postulate which says the sentence to be removed is no longer implied by the new belief set, while $(\circledast\mathbf{2})$ is a syntax-irrelevance property. Rule $(\circledast\mathbf{3})$ is sometimes known as Conjunctive Trisection [Hansson, 1993; Rott, 1992]. A slight reformulation of it can be found already in [Alchourrón *et al.*, 1985] under the name Partial Antitony. It says if χ is believed after removing the conjunction $\lambda \wedge \chi$, then it should also be believed when removing the longer conjunction $\lambda \wedge \chi \wedge \psi$. Rule $(\circledast\mathbf{4})$ is closely-related to the rule Cautious Monotony from the area of non-monotonic reasoning [Kraus *et al.*, 1991], while $(\circledast\mathbf{5})$ and $(\circledast\mathbf{6})$ are the two AGM supplementary postulates for contraction [Alchourrón *et al.*, 1985].

Note the non-appearance in this list of the AGM contraction postulates Vacuity ($\circledast(\bot) \nvdash \lambda$ implies $\circledast(\lambda) \equiv \circledast(\bot)$),Inclusion ($\circledast(\bot) \vdash \circledast(\lambda)$) and Recovery ($\circledast(\lambda) \wedge \lambda \vdash \circledast(\bot)$), none of which are valid in general for removal functions generated from modular contexts. Vacuity has been argued against as a general principle of belief removal in [Booth *et al.*, 2004; Booth and Meyer, 2008]. Inclusion has been questioned in [Booth *et al.*, 2005], while Recovery has long been regarded as controversial (see, e.g., [Hansson, 1991]). Nevertheless we will see in Section 7 how each of these three rules may be captured within our general framework.

The second part of Theorem 2.3 was proved using the following construction.

[1]The appearance of the rules is changed from [Booth *et al.*, 2004] due to the fact that we now take removal functions to be unary. Also one redundant rule from the list in [Booth *et al.*, 2004] is removed (see [Booth and Meyer, 2008]).

Definition 2.4. **($\mathcal{C}$ from $\circledast$)** Given any removal function $\circledast$ we define the context $\mathcal{C}(\circledast) = (<, \prec)$ as follows: $x < y$ iff $y \notin [\circledast(\neg x \wedge \neg y)]$ and $x \prec y$ iff $y \notin [\circledast(\neg x)]$.[2]

[Booth *et al.*, 2004] showed that if $\circledast$ satisfies $(\circledast\mathbf{1})$-$(\circledast\mathbf{6})$ then $\mathcal{C}(\circledast)$ is a modular context and $\circledast = \circledast_{\mathcal{C}(\circledast)}$.

3 Characterising the general family

Now we want to drop the assumption that $<$ is modular and assume only it is a strict partial order. How can we characterise the resulting class of removal functions? We focus first on establishing which of the postulates from Theorem 2.3 are sound for the general family, modifying our initial construction as and when necessary. Clearly we cannot expect that all the rules remain sound. In particular rule $(\circledast\mathbf{6})$ is known to depend on the modularity of $<$ and so might be expected to be the first to go. However we might expect to retain weaker versions of it, for instance:

$(\circledast\mathbf{6a})$ If $\circledast(\lambda \wedge \chi) \vdash \chi$ then $\circledast(\lambda) \vdash \circledast(\lambda \wedge \chi)$.

Indeed we have:

Proposition 3.1. If $\mathcal{C}$ is a general context then $\circledast_{\mathcal{C}}$ satisfies $(\circledast\mathbf{1})$, $(\circledast\mathbf{2})$, $(\circledast\mathbf{4})$, $(\circledast\mathbf{5})$ and $(\circledast\mathbf{6a})$ but not $(\circledast\mathbf{6})$ in general.

Surprisingly, we lose $(\circledast\mathbf{3})$, as the following counterexample shows:

Example 3.2. Assume $L = \{p, q\}$ and let the four valuations of L be represented as $W = \{00, 11, 01, 10\}$, where the first and second numbers denote the truth-values of p, q respectively. Let $<= \{(00, 10)\}$ and $\preceq= \{(10, 01)\}$ (strictly speaking the reflexive closure of this). We have $[\circledast_{\mathcal{C}}(p \wedge q)] = \{00, 10, 01\}$ and $[\circledast_{\mathcal{C}}(q)] = \{00\}$. Hence $10 \in [\neg q \wedge \circledast_{\mathcal{C}}(p \wedge q)]$ but $10 \notin [\circledast_{\mathcal{C}}(q)]$.

This leaves us with a problem, since whereas $(\circledast\mathbf{6})$ is to be considered dispensible, $(\circledast\mathbf{3})$ is a very reasonable property for removal functions. Is there some way we can capture it? It turns out we can capture it if we strengthen the basic property $(\mathcal{C}\mathbf{3})$ to:

$(\mathcal{C}\mathbf{3a})$ $\preceq \subseteq \sqsubseteq^{<}$

In other words if $z < x$ and $x \preceq y$ then $z < y$. On a contrapositive reading, $(\mathcal{C}\mathbf{3a})$ is saying that if there is a world z which $\mathcal{A}$ judges to be more plausible than x but not to y then $\mathcal{A}$ must have a reason to treat y more favourably than x. Note that for modular contexts $(\mathcal{C}\mathbf{3})$ and $(\mathcal{C}\mathbf{3a})$ are equivalent, but in the general case they are not.

Proposition 3.3. Let $\mathcal{C}$ be any context which satisfies $(\mathcal{C}\mathbf{3a})$ then $\circledast_{\mathcal{C}}$ satisfies $(\circledast\mathbf{3})$.

Thus $(\mathcal{C}\mathbf{3a})$ seems necessary. And in fact without it we don't get the following important technical result, which provides the means to describe $<$-minimal λ-worlds purely in terms of the removal function:

Proposition 3.4. Let $\mathcal{C}$ be any context which satisfies $(\mathcal{C}\mathbf{3a})$. Then for all λ such that $\neg\lambda \in L_*$ we have $[\circledast_{\mathcal{C}}(\neg\lambda) \wedge \lambda] = \min_<([\lambda])$.

[2]When a world appears in the scope of a propositional connective, it should be understood as denoting any sentence which has that world as its only model.

Example 3.2 provides a counterexample showing this might not be possible in general, for there we have $[\circledast_{\mathcal{C}}(p \wedge q) \wedge \neg(p \wedge q)] = \{00, 10, 01\}$ but $\min_{<}([\neg(p \wedge q)]) = \{00, 01\}$.

Note rule ($\mathcal{C}\mathbf{3a}$) may also be interpreted as a restricted form of modularity for $<$, since it may be re-written as

$$\forall x, y, z\, (z < x \rightarrow (y \prec x \vee z < y)) .$$

For this reason we make the following definition:

Definition 3.5. A *semi-modular context* is any context $\mathcal{C}$ satisfying ($\mathcal{C}\mathbf{3a}$).

In the rest of the paper we will work only with semi-modular contexts.

3.1 Going the other direction

So far we have a list of sound properties for the removal functions defined from semi-modular contexts. They are the same as the rules which characterise modular removal, but with ($\circledast\mathbf{6}$) replaced by the weaker ($\circledast\mathbf{6a}$). It might be hoped that this list is complete, i.e., that *any* removal function $\circledast$ satisfying these 6 rules is equal to $\circledast_{\mathcal{C}}$ for some semi-modular context $\mathcal{C}$. Indeed we might expect to be able to show $\circledast = \circledast_{\mathcal{C}(\circledast)}$, where $\mathcal{C}(\circledast)$ is the context defined via Definition 2.4. The following result gives us a good start.

Proposition 3.6. Let $\circledast$ be any removal function satisfying ($\circledast\mathbf{1}$)-($\circledast\mathbf{5}$) and ($\circledast\mathbf{6a}$). Then $\mathcal{C}(\circledast)$ is a context, i.e., satisfies ($\mathcal{C}\mathbf{1}$)-($\mathcal{C}\mathbf{3}$).

However to get ($\mathcal{C}\mathbf{3a}$) it seems an extra property is needed:

($\circledast\mathbf{C}$) If $\circledast(\lambda) \wedge \neg\lambda \vdash \circledast(\chi) \wedge \neg\chi$ then $\circledast(\lambda) \vdash \circledast(\chi)$

We can rephrase this using the *Levi Identity* [Levi, 1991].

Definition 3.7. Given any removal function $\circledast$ we may define the function $\circledast^{\mathrm{R}}$ by setting, for each consistent sentence $\lambda \in L$, $\circledast^{\mathrm{R}}(\lambda) = \circledast(\neg\lambda) \wedge \lambda$.

The function $\circledast^{\mathrm{R}}$ is the *revision function* obtained from $\circledast$. Then rule ($\circledast\mathbf{C}$) may be equivalently written as:

($\circledast\mathbf{C}'$) If $\circledast^{\mathrm{R}}(\neg\lambda) \vdash \circledast^{\mathrm{R}}(\neg\chi)$ then $\circledast(\lambda) \vdash \circledast(\chi)$

Thus ($\circledast\mathbf{C}'$) is effectively saying that if revising by $\neg\lambda$ leads to a stronger belief set than revising by $\neg\chi$, then removing λ leads to a stronger belief set than removing χ. The next result confirms that this rule is sound for the removal functions generated by semi-modular contexts, and that this property is enough to show that $\mathcal{C}(\circledast)$ satisfies ($\mathcal{C}\mathbf{3a}$).

Proposition 3.8. Let $\mathcal{C}$ be a semi-modular context. Then $\circledast_{\mathcal{C}}$ satisfies ($\circledast\mathbf{C}$). Furthermore if $\circledast$ is any removal function satisfying ($\circledast\mathbf{C}$) then the context $\mathcal{C}(\circledast)$ satisfies ($\mathcal{C}\mathbf{3a}$).

Rule ($\circledast\mathbf{C}$) is actually quite strong. In the presence of ($\circledast\mathbf{3}$) it implies ($\circledast\mathbf{4}$):

Proposition 3.9. Any removal function which satisfies ($\circledast\mathbf{3}$) and ($\circledast\mathbf{C}$) also satisfies ($\circledast\mathbf{4}$).

This means that, in the axiomatisation of $\circledast_{\mathcal{C}}$ we can replace ($\circledast\mathbf{4}$) with ($\circledast\mathbf{C}$).

To show that the list of rules is complete, it remains to prove $\circledast = \circledast_{\mathcal{C}(\circledast)}$. It turns out that here we need one more additional property which does not seem to follow from the list we have so far:

($\circledast\mathbf{E}$) $\neg(\lambda \wedge \chi) \wedge \circledast(\lambda) \wedge \circledast(\chi) \vdash \circledast(\lambda \wedge \chi)$

This rule may be reformulated as "$\circledast(\lambda) \wedge \circledast(\chi) \vdash (\lambda \wedge \chi) \vee \circledast(\lambda \wedge \chi)$". In this reformulation, the right hand side of the turnstile may be thought of as standing for all those consequences of the conjunction $\lambda \wedge \chi$ which are *believed* upon its removal. The rule is saying that any such surviving consequence must be derivable from the *combination* of $\circledast(\lambda)$ and $\circledast(\chi)$.

Proposition 3.10. Let $\mathcal{C}$ be a semi-modular context. Then $\circledast_{\mathcal{C}}$ satisfies ($\circledast\mathbf{E}$).

Theorem 3.11. *Let $\circledast$ be any removal function satisfying* ($\circledast\mathbf{1}$),($\circledast\mathbf{2}$), ($\circledast\mathbf{3}$),($\circledast\mathbf{C}$), ($\circledast\mathbf{5}$), ($\circledast\mathbf{6a}$) *and* ($\circledast\mathbf{E}$). *Then* $\circledast_{\mathcal{C}(\circledast)} = \circledast$.

Thus, to summarise, we have arrived at the following rules which completely chatacterise the family of removal functions defined from semi-modular contexts:

($\circledast\mathbf{1}$) $\circledast(\lambda) \nvdash \lambda$

($\circledast\mathbf{2}$) If $\lambda_1 \equiv \lambda_2$ then $\circledast(\lambda_1) \equiv \circledast(\lambda_2)$

($\circledast\mathbf{3}$) If $\circledast(\lambda \wedge \chi) \vdash \chi$ then $\circledast(\lambda \wedge \chi \wedge \psi) \vdash \chi$

($\circledast\mathbf{C}$) If $\circledast(\lambda) \wedge \neg\lambda \vdash \circledast(\chi) \wedge \neg\chi$ then $\circledast(\lambda) \vdash \circledast(\chi)$

($\circledast\mathbf{5}$) $\circledast(\lambda \wedge \chi) \vdash \circledast(\lambda) \vee \circledast(\chi)$

($\circledast\mathbf{6a}$) If $\circledast(\lambda \wedge \chi) \vdash \chi$ then $\circledast(\lambda) \vdash \circledast(\lambda \wedge \chi)$

($\circledast\mathbf{E}$) $\neg(\lambda \wedge \chi) \wedge \circledast(\lambda) \wedge \circledast(\chi) \vdash \circledast(\lambda \wedge \chi)$

We will later look at a few more reasonable postulates which are not covered by the above list. But before that we take a look at some different notions of *epistemic entrenchment* which can be defined within this general family.

4 Notions of entrenchment

In this section we want to point out that widening investigation from modular to semi-modular contexts uncovers different notions of the *entrenchment* of a sentence. These distinctions were hidden in the previous case of modular removal, in that for modular removals they collapse into the same notion. Given a removal function $\circledast$ we may define three notions of strict entrenchment relation as follows:

- $\lambda \lhd_1 \chi$ iff $\circledast(\lambda \wedge \chi) \vdash \chi$
This is the usual definition [Gärdenfors, 1988; Rott, 1992]. χ is strictly more entrenched than λ iff, when faced with a choice of giving up at least one of λ, χ, $\mathcal{A}$ will give up λ and hold on to χ.
- $\lambda \lhd_2 \chi$ iff $\exists\psi[\circledast(\lambda \wedge \chi \wedge \psi) \nvdash \lambda \,\&\, \circledast(\lambda \wedge \chi \wedge \psi) \vdash \chi]$
In other words χ is strictly more entrenched than λ iff, there exists some choice situation in which both λ and χ are up for selection, and in which χ, but not λ is chosen. This is similar to the "revealed preference" relation introduced in the theory of rational choice in [Arrow, 1959].
- $\lambda \lhd_3 \chi$ iff $\exists\psi\,[\circledast(\lambda \wedge \psi) \nvdash \lambda \,\&\, \circledast(\chi \wedge \psi) \vdash \chi]$
This one says χ is strictly more entrenched iff there is some ψ such that χ, but not λ is chosen over ψ.

Note $\lhd_2$ could actually be defined in terms of $\lhd_1$ as follows:

$$\lambda \lhd_2 \chi \text{ iff } \exists\psi\,[\text{not}\,(\chi \wedge \psi \lhd_1 \lambda) \ \& \ \lambda \wedge \psi \lhd_1 \chi]\,.$$

while $\lhd_3$ obviously corresponds to:

$$\lambda \lhd_3 \chi \text{ iff } \exists\psi\,[\text{not}\,(\psi \lhd_1 \lambda) \ \& \ \psi \lhd_1 \chi]$$

For each i we will say $\lhd_i$ is *generated by the context* $\mathcal{C}$ if it is obtained from the removal function $\circledast_{\mathcal{C}}$. Our first observation is that, for semi-modular contexts, $\lhd_2$ and $\lhd_3$ coincide.

Proposition 4.1. If $\mathcal{C}$ is a semi-modular context and $\lhd_2, \lhd_3$ are both generated from $\mathcal{C}$ then $\lhd_2 = \lhd_3$.

Next, we show how if $\lhd_1$ and $\lhd_2$ are generated by a semi-modular context, then they may be described directly in terms of that context.

Proposition 4.2. Let $\mathcal{C} = (<, \prec)$ be a semi-modular context and let $\lhd_1$ and $\lhd_2$ be generated from $\mathcal{C}$. Then
(i). $\lambda \lhd_1 \chi$ iff $\min_<([\neg\lambda \vee \neg\chi]) \subseteq [\chi]$.
(ii). $\lambda \lhd_2 \chi$ iff it is not the case that $\forall x \in \min_<([\neg\lambda])\,, \exists y \in \min_<([\neg\chi])$ s.t. $y \sqsubseteq^< x$.

Note how both $\lhd_1$ and $\lhd_2$ are independent of $\prec$. Given this we can establish the following:

Proposition 4.3. $\lhd_1 \subseteq \lhd_2$. The converse is not true in general but is true for modular removals.

[Rott, 1992] proposed the following postulates for any strict entrenchment relation $\lhd$:[3]

(GEE1) **not**$(\lambda \lhd \lambda)$

(GEE2↑) If $\lambda \lhd \chi$ and $\chi \vdash \psi$ then $\lambda \lhd \psi$

(GEE2↓) If $\lambda \lhd \chi$ and $\psi \vdash \lambda$ then $\psi \lhd \chi$

(GEE3↑) If $\lambda \lhd \chi$ and $\lambda \lhd \psi$ then $\lambda \lhd \chi \wedge \psi$

(GEE3↓) If $\lambda \wedge \chi \lhd \chi$ then $\lambda \lhd \chi$

Proposition 4.4. $\lhd_1$ satisfies all the above rules for strict entrenchment relations, while $\lhd_2$ satisfies all but **(GEE3↑)** in general.

As was shown in [Rott, 1992], any strict entrenchment relation satisfying the above rules is transitive and so forms a strict partial order. However $\lhd_2$ fails to be transitive. In fact it fails to be asymmetric, as the next example shows:

Example 4.5. Assume $L = \{p, q\}$ and let $\mathcal{C} = (<, \prec)$ be such that $<= \{(10, 11), (01, 00)\}$. Let $\lambda = \neg(p \wedge q)$ and $\chi = p \vee q$. Then $\min_<([\neg\lambda]) = \{11\}$ and $\min_<([\neg\chi]) = \{00\}$. We obtain both $\lambda \lhd_2 \chi$ and $\chi \lhd_2 \lambda$ via Proposition 4.2(ii), using the fact that $11 \not\sqsubseteq^< 00$ and $00 \not\sqsubseteq^< 11$.

5 Transitivity and Priority

In this section we look at imposing an extra couple of properties on semi-modular contexts $\mathcal{C} = (<, \prec)$, both of which were investigated in the case of modular contexts in [Booth *et al.*, 2004]. There it was shown how the resulting classes of removal functions still remain general enough to include a great many of the classes of removal functions which have been previously proposed in the context of modular removal.

[3]Actually Rott's **(GEE1)** was "not$(\top \lhd \top)$", which given **(GEE2↑)**, **(GEE2↓)** and **(GEE3↓)** is equivalent to the version here. We use this version because, unlike Rott, we do not allow removal of $\top$.

5.1 Transitivity

The first property is the transitivity of $\preceq$, thus making $\preceq$ a preorder. (Recall $\preceq$ is the converse complement of $\prec$.) According to our above interpretation of $\preceq$ this means *if there is no reason to treat y more favourably than x, and no reason to treat z more favourably than y then there is no reason to treat z more favourably than x.*

Proposition 5.1. *(i).* If $\preceq$ is transitive then $\circledast_{\mathcal{C}}$ satisfies the following strengthening of $(\circledast\mathbf{C})$:

$(\circledast\mathbf{C}+)$ If $\circledast(\lambda) \wedge \neg\lambda \vdash \circledast(\chi)$ then $\circledast(\lambda) \vdash \circledast(\chi)$

(ii). If $\circledast$ satisfies $(\circledast\mathbf{C}+)$ then the relation $\preceq$ in $\mathcal{C}(\circledast)$ is transitive.

Note this property is a great deal simpler than the one used to characterise transitivity of $\preceq$ in the modular context in [Booth *et al.*, 2004]. It can be re-written as: If $\circledast^{\mathrm{R}}(\neg\lambda) \vdash \circledast(\chi)$ then $\circledast(\lambda) \vdash \circledast(\chi)$. It says that if the belief set following removal of χ is contained in the belief set following the *revision* by $\neg\lambda$, then it must be contained also in the belief set following the removal of λ. This seems like a reasonable property.

Corollary 5.2. For any removal function $\circledast$, the following are equivalent:
(i). $\circledast$ is generated by a semi-modular context $\mathcal{C} = (<, \prec)$ such that $\preceq$ is transitive.
(ii). $\circledast$ satisfies the list of rules given at the end of Section 3, with $(\circledast\mathbf{C})$ replaced by $(\circledast\mathbf{C}+)$.

5.2 Priority

Now consider the following property of a context $\mathcal{C} = (<, \prec)$:

$(\mathcal{C}\mathbf{P})$ If $x \prec y$ and $y \not\prec x$ then $x < y$

This, too, looks reasonable: if $\mathcal{A}$ has an explicit reason to hold x more plausible than y, but not vice versa, then in the final reckoning he should hold x to be strictly more plausible than y. Consider the following property of removal functions:

$(\circledast\mathbf{P})$ If $\circledast(\lambda) \vdash \chi$ and $\circledast(\chi) \nvdash \lambda$ then $\circledast(\lambda \wedge \chi) \vdash \chi$

This property is briefly mentioned as *Priority* in [Bochman, 2001], and is also briefly mentioned right at the end of [Cantwell, 1999]. It can be read as saying that if λ is excluded following removal of χ, but not vice versa, then χ is strictly more entrenched than λ (using the first, usual, notion of entrenchment from the previous subsection). **For the case of modular removal**, we can obtain the following exact correspondence between $(\mathcal{C}\mathbf{P})$ and $(\circledast\mathbf{P})$:

Proposition 5.3. *(i).* If $\mathcal{C}$ is a modular context satisfying $(\mathcal{C}\mathbf{P})$ then $\circledast_{\mathcal{C}}$ satisfies $(\circledast\mathbf{P})$. *(ii).* If $\circledast$ satisfies $(\circledast\mathbf{P})$ then $\mathcal{C}(\circledast)$ satisfies $(\mathcal{C}\mathbf{P})$.

We remark that in [Booth *et al.*, 2004] the combination of $\preceq$-transitivity and $(\mathcal{C}\mathbf{P})$ was shown to be equivalent to the following single property:

$(\circledast\mathbf{Conserv})$ If $\circledast(\lambda) \nvdash \circledast(\chi)$ then there exists $\psi \in L_*$ such that $\lambda \vdash \psi$ and $\circledast(\psi) \wedge \circledast(\chi) \vdash \lambda$

The above results prove that, in the presence of rules $(\circledast\mathbf{1})$-$(\circledast\mathbf{6})$ from Theorem 2.3, $(\circledast\mathbf{Conserv})$ is *equivalent* to the conjunction of $(\circledast\mathbf{C}+)$ and $(\circledast\mathbf{P})$.

The proof of Proposition 5.3(i) makes critical use of the modularity of $<$. It turns out that $(\circledast\mathbf{P})$ is *not* sound for general semi-modular contexts, even if we insist on $(\mathcal{C}\mathbf{P})$.

Example 5.4. Suppose $L = \{p, q\}$ and that $<= \{(01, 11)\}$ while $\preceq= \{(01, 11)\}$ (strictly speaking the reflexive closure of this). One can verify that $\mathcal{C}$ is a semi-modular context and that $(\mathcal{C}\mathbf{P})$ is satisfied. Now let $\lambda = p \vee \neg q$ and $\chi = \neg p$. Then $[\circledast_\mathcal{C}(\lambda)] = \{01\}$, $[\circledast_\mathcal{C}(\chi)] = \{11, 01, 10\}$ and $[\circledast_\mathcal{C}(\lambda \wedge \chi)] = \{01, 10\}$ and we have $\circledast_\mathcal{C}(\lambda) \vdash \chi$, $\circledast_\mathcal{C}(\chi) \nvdash \lambda$, and $\circledast_\mathcal{C}(\lambda \wedge \chi) \nvdash \chi$. Hence $(\circledast\mathbf{P})$ is not satisfied.

The question now is, which postulate corresponds to $(\mathcal{C}\mathbf{P})$ for general semi-modular contexts? Here is the answer:

Proposition 5.5. *(i).* If $\mathcal{C}$ is a semi-modular context which satisfies $(\mathcal{C}\mathbf{P})$, then $\circledast_\mathcal{C}$ satisfies the following rule:

$(\circledast\mathbf{P}')$ If $\circledast(\lambda) \vdash \chi$ and $\circledast(\chi) \vdash \circledast(\lambda \wedge \chi)$ then $\circledast(\chi) \vdash \lambda$

(ii). If $\circledast$ satisfies $(\circledast\mathbf{P}')$, plus $(\circledast\mathbf{C})$ and $(\circledast\mathbf{1})$, then $\mathcal{C}(\circledast)$ satisfies $(\mathcal{C}\mathbf{P})$.

It is straightforward to see $(\circledast\mathbf{P}')$ is weaker than $(\circledast\mathbf{P})$ given $(\circledast\mathbf{1})$, while it implies $(\circledast\mathbf{P})$ given $(\circledast\mathbf{6})$.

6 Finite Base-Generated Removal

In this section we mention a concrete and important subfamily of our general family of removal functions, the ideas behind which can be seen already throughout the literature on nonmonotonic reasoning and belief change (see in particular [Bochman, 2001] for a general treatment in a belief removal context). Given any, possibly inconsistent, set Σ of sentences, let $cons(\Sigma)$ denote the set of all consistent subsets of Σ. We assume agent $\mathcal{A}$ is in possession of a finite set Σ of sentences which are possible *assumptions* or *defaults*, together with a strict preference ordering $\Subset$ on $cons(\Sigma)$ (with sets "higher" in the ordering assumed more preferred). We assume the following two properties of $\Subset$:

$(\mathbf{\Sigma 1})$ $\Subset$ is a strict partial order

$(\mathbf{\Sigma 2})$ If $A \subset B$ then $A \Subset B$

$(\mathbf{\Sigma 2})$ is a monotonicity requirement stating a given set of defaults is strictly preferred to all its proper subsets.

Definition 6.1. If $\Sigma \subseteq L$ is a finite set of sentences and $\Subset$ is a binary relation over $cons(\Sigma)$ satisfying $(\mathbf{\Sigma 1})$ and $(\mathbf{\Sigma 2})$. Then we call $\mathbb{\Sigma} = \langle \Sigma, \Subset \rangle$ a *prioritised default base.* If in addition $\Subset$ is modular then we call $\mathbb{\Sigma}$ a *modular prioritised default base.*

In practice we might expect the ordering $\Subset$ over $cons(\Sigma)$ to itself be generated from some (not necessarily total) preorder $\precsim$ over over the individual sentences in Σ (again we equate "higher" with "more preferred"). Let $E_1, \ldots, E_k$ be the equivalence classes of $cons(\Sigma)$ under such a $\precsim$, themselves ordered in the natural way by $\precsim$, i.e., $E_1 \precsim E_2$ iff $\alpha \precsim \beta$ for some $\alpha \in E_1$ and $\beta \in E_2$. Then to give but two prominent examples from the lierature (where $\prec$ is the strict part of $\precsim$):

Inclusion-Based [Brewka, 1989] $A \Subset_{ib} B$ iff $\exists i$ s.t. $E_i \cap A \subset E_i \cap B$ and $\forall j$ s.t. $E_i \prec E_j$, $E_j \cap B = E_j \cap A$

Generalised-Lexicographic [Yahi *et al.*, 2008] $A \underline{\Subset}_{gl} B$ iff $\forall i$, if $|E_i \cap B| < |E_i \cap A|$ then $\exists j$ s.t. $E_i \prec E_j$ and $|E_j \cap A| < |E_j \cap B|$. Then $\Subset_{gl}$ is the strict part of $\underline{\Subset}_{gl}$.

We remark that the inclusion-based preference usually assumes the underlying order $\precsim$ over Σ is total. For the generalised-lexicographic example, note if the preorder $\precsim$ over Σ is total then $\Subset_{gl}$ becomes modular and the generalised-lexicographic example reduces to the standard lexicographic case familiar from [Benferhat *et al.*, 1993; Lehmann, 1995].

Proposition 6.2. Let Σ be a finite set of sentences equipped with some preorder $\precsim$ over its elements, and let $\Subset_{ib}$ and $\Subset_{gl}$ be relations over $cons(\Sigma)$ defined from $\precsim$ as above. Then both $\Subset_{ib}$ and $\Subset_{gl}$ satisfy $(\mathbf{\Sigma 1})$ and $(\mathbf{\Sigma 2})$.

How does the agent use a prioritised default base $\mathbb{\Sigma} = \langle \Sigma, \Subset \rangle$ to remove beliefs? For $\Sigma \subseteq L$ and $\lambda \in L_*$ let $cons(\Sigma, \lambda) \stackrel{\text{def}}{=} \{S \in cons(\Sigma) \mid S \nvdash \lambda\}$. Then from $\mathbb{\Sigma}$ we may define a removal function $\circledast_{\mathbb{\Sigma}}$ by setting, for each $\lambda \in L_*$,

$$\circledast_{\mathbb{\Sigma}}(\lambda) = \bigvee \left\{ \bigwedge S \mid S \in \max_{\Subset} cons(\Sigma, \lambda) \right\}.$$

In other words, after removing λ, $\mathcal{A}$ will believe precisely those sentences which are consequences of *all maximally preferred* subsets of Σ which do not imply λ.

We will now show how the family of removal functions generated from prioritised default bases fits into our general family. From a given $\mathbb{\Sigma} = \langle \Sigma, \Subset \rangle$ we may define a context $\mathcal{C}(\mathbb{\Sigma}) = (<, \prec)$ as follows. Let $sent_\Sigma(x) \stackrel{\text{def}}{=} \{\alpha \in \Sigma \mid x \in [\alpha]\}$. Then

- $x < y$ iff $sent_\Sigma(y) \Subset sent_\Sigma(x)$
- $x \prec y$ iff $sent_\Sigma(x) \nsubseteq sent_\Sigma(y)$

Thus we define x to be more plausible than y iff the set of sentences in Σ satisfied by x is more preferred than the set of sentences in Σ satisfied by y. Meanwhile we have the natural interpretation for $\prec$ that $\mathcal{A}$ has a reason to hold x to be more plausible than y precisely when one of the sentences in Σ is satisfied by x but not y.

Theorem 6.3. (i). $\mathcal{C}(\mathbb{\Sigma})$ *defined above forms a semi-modular context (which is modular if* $\Subset$ *is modular).*
(ii). $\preceq$ *is transitive and the condition* $(\mathcal{C}\mathbf{P})$ *from the previous section is satisfied.*
(iii). $\circledast_{\mathbb{\Sigma}} = \circledast_{\mathcal{C}(\mathbb{\Sigma})}$.

Thus we have shown that every removal function generated by a prioritised default base may *always* be generated by a semi-modular context which furthermore satisfies the two conditions on contexts mentioned in the previous section. By the results of the previous sections, this means we automatically obtain a list of sound postulates for the default base-generated removals.

Corollary 6.4. Let $\mathbb{\Sigma}$ be any prioritised default base. Then $\circledast_{\mathbb{\Sigma}}$ satisfies all the rules listed at the end of Section 3, as well as $(\circledast\mathbf{C}+)$ and $(\circledast\mathbf{P}')$ from the last section.

Note we have shown how every prioritised default base gives rise to a semi-modular context satisfying $\preceq$-transitivity and $(\mathcal{C}\mathbf{P})$. An open question is whether *every* such context arises in this way.

7 AGM Preferential Removal

Recall that three of the basic AGM postulates for contraction do not hold in general for the removal functions generated by semi-modular contexts, namely Inclusion, Recovery and Vacuity. In this section we show how each of these rules can be captured. In [Booth *et al.*, 2004] it was shown already how they may be captured within the class of modular context-generated removal. It turns out that more or less the same constructions can be used for the wider class considered here, although some complications arise regarding Vacuity.

7.1 Inclusion

The Inclusion rule is written in our setting as follows:

$(\circledast\mathbf{I})$ $\circledast(\bot) \vdash \circledast(\lambda)$

To capture $(\circledast\mathbf{I})$ for any removal generated from any semi-modular context $\mathcal{C} = (<, \prec)$, we need only to require the following condition on $\mathcal{C}$:

$(\mathcal{C}\mathbf{I})$ $\min_<(W) \subseteq \min_\prec(W)$

According to our interpretation of $\prec$, $(\mathcal{C}\mathbf{I})$ is stating that, for any world x, if $\mathcal{A}$ has some explicit reason favour some world y over x (i.e., $y \prec x$) then in the final reckoning $\mathcal{A}$ must hold *some* world z (not necessarily the same as y) more plausible than x (i.e., $z < x$).

Proposition 7.1. *(i).* If $\mathcal{C}$ satisfies $(\mathcal{C}\mathbf{I})$ then $\circledast_{\mathcal{C}}$ satisfies $(\circledast\mathbf{I})$. *(ii).* If $\circledast$ satisfies $(\circledast\mathbf{I})$ then $\mathcal{C}(\circledast)$ satisfies $(\mathcal{C}\mathbf{I})$.

Given any removal function $\circledast$ we can always obtain a removal function which satisfies $(\circledast\mathbf{I})$ by taking the *incarceration* $\circledast^{\mathrm{I}}$ of $\circledast$ [Booth *et al.*, 2005].

$$\circledast^{\mathrm{I}}(\lambda) \overset{\mathrm{def}}{=} \circledast(\bot) \vee \circledast(\lambda).$$

Or alternatively we can modify a given context $\mathcal{C} = (<, \prec)$ into $\mathcal{C}^{\mathrm{I}} = \left(<, \prec^{\mathrm{I}}\right)$, where $x \preceq^{\mathrm{I}} y$ iff either $x \preceq y$ or $x \in \min_<(W)$. It is easy to check $\mathcal{C}^{\mathrm{I}} = \mathcal{C}\left(\circledast^{\mathrm{I}}\right)$.

7.2 Recovery

The Recovery rule is written as follows:

$(\circledast\mathbf{R})$ $\circledast(\lambda) \wedge \lambda \vdash \circledast(\bot)$

The corresponding property on contexts $\mathcal{C} = (<, \prec)$ is:

$(\mathcal{C}\mathbf{R})$ If $y \notin \min_<(W)$ and $x \neq y$ then $x \prec y$

Thus the only worlds $\nabla_{\preceq}(x)$ contains, other than x itself, are worlds in $\min_<(W)$.

Proposition 7.2. *(i).* If $\mathcal{C}$ satisfies $(\mathcal{C}\mathbf{R})$ then $\circledast_{\mathcal{C}}$ satisfies $(\circledast\mathbf{R})$. *(ii).* If $\circledast$ satisfies $(\circledast\mathbf{R})$ then $\mathcal{C}(\circledast)$ satisfies $(\mathcal{C}\mathbf{R})$.

Note the combination of $(\mathcal{C}\mathbf{I})$ and $(\mathcal{C}\mathbf{R})$ specifies $\prec$, equivalently $\preceq$, uniquely in terms of $<$, viz.

$$x \preceq_{agm} y \text{ iff } x = y \text{ or } x \in \min_<(W).$$

and we obtain the removal recipe of AGM contraction, in which removal of λ boils down to just adding the $<$-minimal $\neg\lambda$-worlds to the $<$-minimal worlds:

$$[\circledast_{agm}(\lambda)] = \min_<(W) \cup \min_<([\neg\lambda]).$$

It is easy to check that the resulting context $\mathcal{C}$ satisfies condition $(\mathcal{C}\mathbf{3a})$ and thus forms a semi-modular context. It is also easy to check $(\mathcal{C}\mathbf{P})$ is satisfied and that the above-defined $\preceq_{agm}$ is transitive. Thus the above $\circledast_{agm}$ also satisfies $(\circledast\mathbf{C}+)$ and $(\circledast\mathbf{P}')$ from Section 5.

7.3 Vacuity

The Vacuity rule is written as follows:

$(\circledast\mathbf{V})$ If $\circledast(\bot) \nvdash \lambda$ then $\circledast(\lambda) \equiv \circledast(\bot)$

Unlike in the modular case, where Vacuity is known to follow from Inclusion for modular removal functions [Booth *et al.*, 2004], $(\circledast\mathbf{V})$ does not even hold in general for the above preferential AGM contraction $\circledast_{agm}$. This was essentially noticed, in a revision context, in [Benferhat *et al.*, 2005].

Example 7.3. Let $L = \{p, q\}$ and $<= \{(11, 01)\}$. So $[\circledast_{agm}(\bot)] = \{00, 11, 10\}$. Let $\lambda = p$. Then we have $\circledast_{agm}(\bot) \nvdash \lambda$ (because $00 \in [\circledast_{agm}(\bot)]$), but $\min_<([\neg\lambda]) = \{00, 01\}$, so $[\circledast_{agm}(\lambda)] = \min_<(W) \cup \min_<([\neg\lambda]) = W \neq [\circledast_{agm}(\bot)]$.

In order to ensure $\circledast_{agm}$ satisfies $(\circledast\mathbf{V})$ it is necessary, as is done in [Katsuno and Mendelzon, 1992], to enforce the following property on $<$.

$(<\mathbf{V})$ $\forall x, y\, ((x \in \min_<(W) \wedge y \notin \min_<(W)) \rightarrow x < y)$.

In other words all $<$-minimal worlds can be compared with, and are below, every world which is not $<$-minimal. For general semi-modular contexts $\mathcal{C} = (<, \prec)$ we also require the following condition, which is weaker than $(\mathcal{C}\mathbf{I})$:

$(\mathcal{C}\mathbf{V})$ If $x, y \in \min_<(W)$ then $x \not\prec y$

This property says that for any two of his $<$-minimal worlds, $\mathcal{A}$ will not have explicit reason to hold one to be more plausible than the other.

Proposition 7.4. *(i).* If $\mathcal{C}$ satisfies $(\mathcal{C}\mathbf{V})$ and $(<\mathbf{V})$ then $\circledast_{\mathcal{C}}$ satisfies $(\circledast\mathbf{V})$. *(ii).* If $\circledast$ satisfies $(\circledast\mathbf{V})$ then $\mathcal{C}(\circledast)$ satisfies $(\mathcal{C}\mathbf{V})$.

8 Conclusion

In this paper we introduced a family of removal functions, generalising the one given in [Booth *et al.*, 2004] to allow for incomparabilities in the plausibility relation $<$ between possible worlds. Removal is carried out using the plausibility relation in combination with a second relation $\prec$ which can be thought of as indicating "reasons" for holding one world to be more plausible than another. We axiomatically characterised this general family as well as certain subclasses, and we showed how this family includes some important and natural families of belief removal, specifically those which may be generated from prioritised default bases and the preferential counterpart of AGM contraction. Our results show the central construct used in this paper, i.e., semi-modular contexts, to be a very useful tool in the study of belief removal functions.

For future work we would like to locate further subclasses of interest, for example the counterparts in this setting of systematic withdrawal [Meyer *et al.*, 2002] and severe withdrawal [Rott and Pagnucco, 1999]. We would also like to employ semi-modular contexts in the setting of *social belief removal* [Booth and Meyer, 2008], in which there are several agents, each assumed to have their own removal function, and in which all agents must remove some belief to become consistent with each other. [Booth and Meyer, 2008] showed that, under the assumption that each agent uses a removal function generated from a *modular* context, certain *equilibrium points* in the social removal process are guaranteed to exist. An interesting question would be whether these results generalise to the *semi-modular* case. Since semi-modular contexts are built from strict partial orders, this question should also be of some relevance to the problem of *aggregating strict partial orders* [Pini *et al.*, 2005].

References

[Alchourrón *et al.*, 1985] C. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, 1985.

[Arrow, 1959] K. Arrow. Rational choice functions and orderings. *Economica*, 26:121–127, 1959.

[Benferhat *et al.*, 1993] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency management and prioritized syntax-based entailment. In *IJCAI*, pages 640–647, 1993.

[Benferhat *et al.*, 2005] S. Benferhat, S. Lagrue, and O. Papini. Revision of partially ordered information: Axiomatization, semantics and iteration. In L. Pack Kaelbling and A. Saffiotti, editors, *IJCAI*, pages 376–381. Professional Book Center, 2005.

[Bochman, 2001] A. Bochman. *A Logical Theory of Nonmonotonic Inference and Belief Change*. Springer, 2001.

[Booth and Meyer, 2008] R. Booth and T. Meyer. Equilibria in social belief removal. In *KR*, pages 145–155, 2008.

[Booth *et al.*, 2004] R. Booth, S. Chopra, T. Meyer, and A. Ghose. A unifying semantics for belief change. In *Proceedings of ECAI'04*, pages 793–797, 2004.

[Booth *et al.*, 2005] R. Booth, S. Chopra, A. Ghose, and T. Meyer. Belief liberation (and retraction). *Studia Logica*, 79(1):47–72, 2005.

[Brewka, 1989] G. Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *IJCAI*, pages 1043–1048, 1989.

[Cantwell, 1999] J. Cantwell. Relevant contraction. In *Proceedings of the Dutch-German Workshop on Non-Monotonic Reasoning (DGNMR'99)*, 1999.

[Cantwell, 2003] J. Cantwell. Eligible contraction. *Studia Logica*, 73:167–182, 2003.

[Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux*. MIT Press, 1988.

[Hansson, 1991] S. O. Hansson. Belief contraction without recovery. *Studia Logica*, 50(2):251–260, 1991.

[Hansson, 1993] S. O. Hansson. Changes on disjunctively closed bases. *Journal of Logic, Language and Information*, 2:255–284, 1993.

[Katsuno and Mendelzon, 1992] H. Katsuno and A. O. Mendelzon. Propositional knowledge base revision and minimal change. *Artif. Intell.*, 52(3):263–294, 1992.

[Kraus *et al.*, 1991] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1991.

[Lehmann, 1995] D. Lehmann. Another perspective on default reasoning. *Ann. Math. Artif. Intell.*, 15(1):61–82, 1995.

[Levi, 1991] I. Levi. *The Fixation of Belief and Its Undoing*. Cambridge University Press, Cambridge, 1991.

[Maynard-Zhang and Lehmann, 2003] P. Maynard-Zhang and D. Lehmann. Representing and aggregating conflicting beliefs. *Journal of Artificial Intelligence Research*, 19:155–203, 2003.

[Meyer *et al.*, 2002] T. Meyer, J. Heidema, W. Labuschagne, and L. Leenen. Systematic withdrawal. *Journal of Philosophical Logic*, 31(5):415–443, 2002.

[Pini *et al.*, 2005] M. S. Pini, F. Rossi, K. Brent Venable, and T. Walsh. Aggregating partially ordered preferences: impossibility and possibility results. In *TARK*, pages 193–206, 2005.

[Rott and Pagnucco, 1999] H. Rott and M. Pagnucco. Severe withdrawal (and recovery). *Journal of Philosophical Logic*, 28:501–547, 1999.

[Rott, 1992] H. Rott. Preferential belief change using generalized epistemic entrenchment. *Journal of Logic, Language and Information*, 1:45–78, 1992.

[Shoham, 1987] Y. Shoham. A semantical approach to nonmonotic logics. In *LICS*, pages 275–279, 1987.

[Yahi *et al.*, 2008] S. Yahi, S. Benferhat, S. Lagrue, M. Sérayet, and O. Papini. A lexicographic inference for partially preordered belief bases. In *KR*, pages 507–517, 2008.

How Do You Prefer?
An Empirical Analysis of Comparative Non-monotonic Preferences Semantics

Rui da Silva Neves
Université Toulouse-II,
CLLE-LTC, CNRS UMR 5263
5 Allées Machado
31058 Toulouse Cedex 9, France
neves@univ-tlse2.fr

Souhila Kaci
Université Lille-Nord de France, Artois
CRIL, CNRS UMR 8188
IUT de Lens
F-62307, France
kaci@cril.fr

Abstract

Representing preferences and reasoning about them are important issues for many real-life applications. Several monotonic and non-monotonic qualitative formalisms have been developed for this purpose. Most of them are based on comparative preferences, for e.g. "I prefer red wine to white wine". However this simple and natural way to express preferences comes also with many difficulties regarding their interpretation. Several (more or less strong) semantics have been proposed leading to different (pre)orders on outcomes. So far researchers have argued for each semantics and compared them from purely theoretical standpoint. In this paper, we report results of the first empirical comparison of existing non-monotonic semantics (strong, optimistic, pessimistic and ceteris paribus) based on psychological data. Thirty participants were asked to rank 8 menus according to their preferences and to compare 31 pairs of menus. The recorded preferences allowed to compute compact preferences and ranking menus for each participant according to the four semantics under study, and to compare these ranks to participant's ones. Results show that non-monotonic optimistic and pessimistic preferences are the semantics that better fit human data, strong and ceteris paribus semantics being less psychologically plausible given our task.

1 Introduction

Preferences are very useful in many real-life problems. They are inherently a multi-disciplinary topic, of interest to economists, computer scientists, operations researchers, mathematicians, logicians, philosophers and psychologists. Preferences are a relatively new topic in Artificial Intelligence and are becoming of greater interest in many areas such as non-monotonic reasoning, multi-agent systems, constraint satisfaction, decision making, social choice theory, decision-theoretic planning, etc.

It has been early recognized that value functions/orderings cannot be explicitly defined because of a great number of outcomes or simply because the user is not willing to state her/his preferences on each pair of outcomes. Indeed preferences should be handled in a compact (or succint) way, starting from non completely explicit preferences expressed by a user. Compact representation languages allow a simple readability of user's preferences. Therefore, a good compact language should be as close as possible to user's specification of preferences on possible outcomes.

The compact languages for preference representation have been extensively developed in Artificial Intelligence in the last decade [Boutilier *et al.*, 2004; Wilson, 2004; Brewka *et al.*, 2004]. In particular, (conditional) comparative statements are often used for describing preferences in a local, contextualized manner for e.g., "I prefer fish to meat", "if meat is served then I prefer red wine to white wine", etc. Indeed, it is easier and more natural to express such qualitative comparative statements than to say that I prefer fish with the weight .8 and prefer meat with the weight .2. Some generic principles are often used for completing the qualitative comparative preference statements[1] [Hansson, 1996; Boutilier, 1994; Benferhat *et al.*, 2002; Wilson, 2004]. Although comparative preference statements allow for a simple and natural way to express preferences, they come however with many difficulties regarding their interpretation.

Comparative preferences are often interpreted following the well known ceteris paribus semantics [Hansson, 1996]. This is due to the CP-net approach [Boutilier *et al.*, 1999] which has emerged in the last decade as the preeminent and prominent method for processing preferences in Artificial Intelligence, thanks to its intuitive appeal. Following this pinciple, the statement "I prefer fish to meat" is interpreted as, given two meals that differ *only* in the main dish, the meal with fish is preferred to the meal with meat. However, CP-nets behave monotonically and do not allow for the handling of preferences with defaults. For example, we can prefer fish to meat, but when available fish is red tuna and meat is poultry, we can prefer the reverse. As such, CP-nets are not appropriate when a vegetarian expresses her/his preference about meat-based meals and fish-based meals. Moreover, in CP-nets, ceteris paribus semantics states that the two meals *fish-cake* and *meat-ice cream* are incomparable w.r.t. the preference statement "I prefer fish to meat" while a veg-

[1]From now on, we simply speak about comparative preference statements.

etarian would prefer any fish-based meal to any meat-based meal. Fortunately, ceteris paribus is not the only possible reading of comparative preference statements and other intuitively non-monotonic meaningful semantics may also be encountered, and researchers have also argued for other semantics [Boutilier, 1994; Benferhat *et al.*, 2002] based on insights from non-monotonic reasoning such as system Z [Pearl, 1990]. Note also that ceteris paribus semantics can also be non-monotonic outside CP-net framework. For example, the menu $fish - red$ is preferred to the menu $fish - white$ w.r.t. the preference statement "red is preferred to $\neg red$" following ceteris paribus semantics. However the additional preference statement "$fish \wedge white$ is preferred to $fish \wedge \neg white$" induces the reverse preference, namely $fish - white$ is preferred to $fish - red$.

So far researchers have argued for a semantics or another from purely theoretical standpoint or for modeling a specific application. In this paper, we provide the first empirical comparison of existing non-monotonic semantics (including ceteris paribus) based on psychological data. This psychological inquiry is founded by previous work on the non-monotonic nature of human reasoning. For example, it has been shown that human inference is consistent with System P [Kraus *et al.*, 1990] (see [Neves *et al.*, 2002; Benferhat *et al.*, 2004; 2005]) and that System P constitutes a psychologically sound base of rationality postulates for the evaluation of non-monotonic reasoning systems. In our study, participants were asked to rank 8 menus according to their preferences and to compare 31 pairs of menus. The recorded preferences were compared to those provided by the considered semantics. Results show that optimistic and pessimistic preferences are the semantics that better fit human data, strong, ceteris paribus semantics being less psychologically plausible given our task.

The remainder of this paper is organized as follows. After providing notations and necessary definitions, we recall in Section 3 the different semantics of comparatives preferences proposed in literature. Opportunistic semantics is introduced at the theoretical level but is not studied empirically because a unique complete preorder does not exist w.r.t. this principle. In Section 4 we recall algorithms to rank-order outcomes for each semantics. In Section 5 we provide empirical comparison of the different semantics based on psychological data. Lastly we conclude.

2 Notations

Let $V = \{X_1, \cdots, X_h\}$ be a set of h variables. Each variable X_i takes its values in a domain $Dom(X_i)$ which is a set of uninterpreted constants $\mathcal{D}$ or rational numbers $\mathcal{Q}$. A possible outcome, denoted t, is the result of assigning a value in $Dom(X_i)$ to each variable X_i in V. Ω is the set of all possible outcomes. We suppose that this set is fixed and finite. $Asst(V')$, with $V' \subseteq V$, is the set of all possible assignments to variables in V'. Therefore $\Omega = Asst(V)$.
Let $\mathcal{L}$ be a language based on V. $Mod(\varphi)$ denotes the set of outcomes that make the formula φ (built on $\mathcal{L}$) true. We write $t \models \varphi$ when $t \in Mod(\varphi)$ and say that t satisfies φ.

An ordering relation $\succeq$ on $\mathcal{X} = \{x, y, z, \cdots\}$ is a reflexive binary relation such that $x \succeq y$ stands for x is at least as preferred as y. $x \approx y$ means that both $x \succeq y$ and $y \succeq x$ hold, i.e., x and y are equally preferred. Lastly $x \sim y$ means that neither $x \succeq y$ nor $y \succeq x$ holds, i.e., x and y are incomparable. A strict ordering relation on $\mathcal{X}$ is an irreflexive binary relation such that $x \succ y$ means that x is strictly preferred to y. We also say that x dominates y. A strict ordering relation $\succ$ can be defined from an ordering relation $\succeq$ as $x \succ y$ if $x \succeq y$ holds but $y \succeq x$ does not.
When neither $x \succ y$ nor $y \succ x$ holds, we also write $x \sim y$.
$\succeq$ (resp. $\succ$) is a preorder (resp. order) on $\mathcal{X}$ if and only if $\succeq$ (resp. $\succ$) is transitive, i.e., if $x \succeq y$ and $y \succeq z$ then $x \succeq z$ (if $x \succ y$ and $y \succ z$ then $x \succ z$).
$\succeq$ (resp. $\succ$) is a complete preorder (resp. order) if and only if $\forall x, y \in \mathcal{X}$, we have either $x \succeq y$ or $y \succeq x$ (resp. either $x \succ y$ or $y \succ x$).
The set of the best (or undominated) elements of $A \subseteq \mathcal{X}$ w.r.t. $\succ$, denoted $\max(A, \succ)$, is defined by $\max(A, \succ) = \{x | x \in A, \nexists y \in A, y \succ x\}$. The set of the worst elements of $A \subseteq \mathcal{X}$ w.r.t. $\succ$, denoted $\min(A, \succ)$, is defined by $\min(A, \succ) = \{x | x \in A, \nexists y \in A, x \succ y\}$.
The best (resp. worst) elements of A w.r.t. $\succeq$ is $\max(A, \succ)$ (resp. $\min(A, \succ)$) where $\succ$ is the strict ordering relation associated to $\succeq$.

A complete preorder $\succeq$ can also be represented by a well ordered partition of Ω. This is an equivalent representation, in the sense that each preorder corresponds to one ordered partition and vice versa.

Definition 1 (Partition) *A sequence of sets of outcomes of the form $(E_1, \ldots, E_n)$ is a partition of Ω if and only if*

(i) $\forall i$, $E_i \neq \emptyset$,

(ii) $E_1 \cup \cdots \cup E_n = \Omega$, and

(iii) $\forall i, j$, $E_i \cap E_j = \emptyset$ for $i \neq j$.

A partition of Ω is ordered if and only if it is associated with a preorder $\succeq$ on Ω such that ($\forall t, t' \in \Omega$ with $t \in E_i, t' \in E_j$ we have $i \leq j$ if and only if $t \succeq t'$).

3 Comparative preference statements

We denote comparative statements of the form "I prefer p to q" as $p > q$ and denote conditional (called also contextual) comparative statements of the form "if r is true then I prefer p to q" as $r : p > q$, where p, q and r are any propositional formulas. Note that $r : p > q$ is equivalent to $r \wedge p > r \wedge q$. Indeed we will simply focus on statements of the form $p > q$.

Comparative statements, apparently very simple, come with difficulties regarding their interpretation. How should we interpret such statements? For example, given the preference statement "I prefer fish to meat", how do we rank-order meals based on fish and those based on meat? Five semantics have been proposed in literature:

- *ceteris paribus preferences:* [Hansson, 1996]
  any fish-based meal is preferred to any meat-based meal if the two meals are exactly the same elsewhere (for example wine and dessert).

- *strong preferences:* [Boutilier, 1994; Chomicki, 2003; Wilson, 2004]
  any fish-based meal is preferred to any meat-based meal.
- *optimistic preferences:* [Benferhat *et al.*, 1992; Boutilier, 1994; Pearl, 1990]
  at least one fish-based meal is preferred to all meat-based meals.
- *pessimistic preferences:* [Benferhat *et al.*, 2002]
  at least one meat-based meal is less preferred to all fish-based meals.
- *opportunistic preferences:* [van der Torre and Weydert, 2001]
  at least one fish-based meal is preferred to at least one meat-based meal.

We define preference of the formula p over the formula q as preference of $p \wedge \neg q$ over $\neg p \wedge q$. This is standard and known as von Wright's expansion principle [von Wright, 1963]. Additional clauses may be added for the cases in which sets of outcomes are nonempty, to prevent the satisfiability of preferences like $p > \top$ and $p > \bot$. We do not consider this borderline condition to keep the formal machinery as simple as possible.

We denote the preference of p over q following strong semantics (resp. ceteris paribus, optimistic, pessimistic, opportunistic) by $p >_{st} q$ (resp. $p >_{cp} q$, $p >_{opt} q$, $p >_{pes} q$, $p >_{opp} q$).

Definition 2 *Let p and q be two propositional formulas and $\succeq$ be a preorder on Ω.*

- *$\succeq$ satisfies $p >_{st} q$, denoted $\succeq \models p >_{st} q$, iff $\forall t \models p \wedge \neg q$, $\forall t' \models \neg p \wedge q$ we have $t \succ t'$.*
- *$\succeq$ satisfies $p >_{cp} q$, denoted $\succeq \models p >_{cp} q$, iff $\forall t \models p \wedge \neg q$, $\forall t' \models \neg p \wedge q$ we have $t \succ t'$, where t and t' have the same assignment on variables that do not appear in p and q.*
- *$\succeq$ satisfies $p >_{opt} q$, denoted $\succeq \models p >_{opt} q$, iff $\exists t \models p \wedge \neg q$, $\forall t' \models \neg p \wedge q$ we have $t \succ t'$.*
- *$\succeq$ satisfies $p >_{pes} q$, denoted $\succeq \models p >_{pes} q$, iff $\exists t' \models \neg p \wedge q$, $\forall t \models p \wedge \neg q$ we have $t \succ t'$.*
- *$\succeq$ satisfies $p >_{opp} q$, denoted $\succeq \models p >_{opp} q$, iff $\exists t \models p \wedge \neg q$, $\exists t' \models \neg p \wedge q$ we have $t \succ t'$.*

The following lemma gives the mathematical description of Definition 2:

Lemma 1 *Let p and q be two propositional formulas and $\succeq$ be a preorder on Ω.*

- *$\succeq$ satisfies $p >_{st} q$ iff $\forall t \in \min(p \wedge \neg q, \succeq)$, $\forall t' \in \max(\neg p \wedge q, \succeq)$ we have $t \succ t'$.*
- *$\succeq$ satisfies $p >_{cp} q$ iff $\forall t \in \min(p \wedge \neg q, \succeq)$, $\forall t' \in \max(\neg p \wedge q, \succeq)$ we have $t \succ t'$, where t and t' have the same assignment on variables that do not appear in p and q.*
- *$\succeq$ satisfies $p >_{opt} q$ iff $\forall t \in \max(p \wedge \neg q, \succeq)$, $\forall t' \in \max(\neg p \wedge q, \succeq)$ we have $t \succ t'$.*
- *$\succeq$ satisfies $p >_{pes} q$ iff $\forall t \in \min(p \wedge \neg q, \succeq)$, $\forall t' \in \min(\neg p \wedge q, \succeq)$ we have $t \succ t'$.*
- *$\succeq$ satisfies $p >_{opp} q$ iff $\forall t \in \max(p \wedge \neg q, \succeq)$, $\forall t' \in \min(\neg p \wedge q, \succeq)$ we have $t \succ t'$.*

A preference set is a set of preferences of the same type.

Definition 3 (Preference set) *A preference set of type $\rhd$, denoted $\mathcal{P}_\rhd$, is a set of preferences of the form $\{p_i \rhd q_i | i = 1, \cdots, n\}$, where $\rhd \in \{ >_{st}, >_{cp}, >_{opt}, >_{pes}, >_{opp} \}$. A complete preorder $\succeq$ is a model of $\mathcal{P}_\rhd$ if and only if $\succeq$ satisfies each preference $p_i \rhd q_i$ in $\mathcal{P}_\rhd$.*

A set $\mathcal{P}_\rhd$ is consistent if it has a model.

4 From comparative preference statements to preorders on outcomes

Generally we have to deal with several comparative preference statements expressed by a user. Once the semantics is fixed, the problem to tackle is how to deal with such statements? Several types of queries can be asked about preferences: what are the preferred outcomes? Is one outcome better than the other? In many applications (for e.g. database queries), users are more concerned with the preferred outcomes. However preferred outcomes are not always feasible. For example the best menus w.r.t. a user's preferences may be no longer available so we have to look for menus that are immediately less preferred w.r.t. user's preferences. In such a case a complete preorders on menus is needed to answer user's preferences. Indeed we restrict ourselves to semantic models that derive complete preorders on outcomes. In the following, we recall algorithms which derive a unique complete preorder given a set of preferences of the same type w.r.t. specificity principle [Yager, 1983]. A unique complete preorder for opportunistic preferences does not exist w.r.t. this principle. Indeed we do not consider such preferences in the remainder of the paper.

Let $\mathcal{P}_\rhd = \{s_i : p_i \rhd q_i | i = 1, \cdots, n\}$ be a preference set with $\rhd \in \{ >_{st}, >_{cp}, >_{opt}, >_{pes} \}$. Given $\mathcal{P}_\rhd$, we define a set of pairs on Ω as follows:

$$\mathcal{L}(\mathcal{P}_\rhd) = \{C_i = (L(s_i), R(s_i)) | i = 1, \cdots, n\},$$

where $L(s_i) = \{t | t \in \Omega, t \models p_i \wedge \neg q_i\}$ and $R(s_i) = \{t | t \in \Omega, t \models \neg p_i \wedge q_i\}$.

Example 1 *Let* dish, wine *and* dessert *be three variables such that $Dom(dish) = \{fish, meat\}$, $Dom(wine) = \{white, red\}$ and $Dom(dessert) = \{cake, ice\_cream\}$. We have $\Omega = \{t_0 = fish - white - ice\_cream$, $t_1 = fish - white - cake, t_2 = fish - red - ice\_cream$, $t_3 = fish - red - cake, t_4 = meat - white - ice\_cream$, $t_5 = meat - white - cake, t_6 = meat - red - ice\_cream$, $t_7 = meat - red - cake\}$.*
Let $\mathcal{P}_\rhd = \{s_1 : fish \rhd meat, s_2 : red \wedge cake \rhd white \wedge ice\_cream, s_3 : fish \wedge white \rhd fish \wedge red\}$. We have $\mathcal{L}(\mathcal{P}_\rhd) = \{C_1 = (\{t_0, t_1, t_2, t_3\}, \{t_4, t_5, t_6, t_7\})$, $C_2 = (\{t_3, t_7\}, \{t_0, t_4\}), C_3 = (\{t_0, t_1\}, \{t_2, t_3\})\}$.

4.1 Optimistic preferences

Several complete preorders may satisfy a set of optimistic preferences. It is however possible to characterize a unique preorder among them under certain assumption. The semantics of optimistic preferences is close to the one of conditionals. Indeed system Z [Pearl, 1990] has been used [Benferhat *et al.*, 1992; Boutilier, 1994]. It rank-orders outcomes under the assumption that outcomes are preferred unless the contrary is stated. Indeed outcomes are put in the highest possible rank in the preorder while being consistent with preferences at hand. This principle ensures that the complete preorder is unique and the most compact one among preorders satisfying the set of preferences[2]. Algorithm 1 gives the way this preorder is computed. At each step of the algorithm, we put in E_i outcomes that are not dominated by any other outcomes. These outcomes are those which do not appear in the right-hand side of any pair $(L(s_i), R(s_i))$ of $\mathcal{L}(\mathcal{P}_{>_{opt}})$.

Algorithm 1: A complete preorder associated with $\mathcal{P}_{>_{opt}}$.

```
Data: A preference set $\mathcal{P}_{>_{opt}}$.
Result: A complete preorder $\succeq$ on $\Omega$.
begin
    $l = 0$
    while $\Omega \neq \emptyset$ do
        $l = l + 1$
        $E_l = \{t | t \in \Omega, \nexists (L(s_i), R(s_i)) \in \mathcal{L}(\mathcal{P}_{>_{opt}}), t \in R(s_i)\}$
        if $E_l = \emptyset$ then
            stop (inconsistent preferences), $l = l - 1$
        - $\Omega = \Omega \backslash E_l$
        /** remove satisfied preferences **/
        - remove $(L(s_i), R(s_i))$ where $L(s_i) \cap E_l \neq \emptyset$
    return $\succeq = (E_1, \cdots, E_l)$.
end
```

Example 2 *(Example 1 con'd)*
We have $E_1 = \{t_1\}$. We remove C_1 and C_3 since $s_1 = fish >_{opt} meat$ and $s_3 : fish \wedge white >_{opt} fish \wedge red$ are satisfied. We get $\mathcal{L}(\mathcal{P}_{>_{opt}}) = \{C_2 = (\{t_3, t_7\}, \{t_0, t_4\})\}$. Now $E_2 = \{t_2, t_3, t_5, t_6, t_7\}$. We remove C_2 since $s_2 : red \wedge cake >_{opt} white \wedge ice\_cream$ is satisfied. So $\mathcal{L}(\mathcal{P}_{>_{opt}}) = \emptyset$. Lastly, $E_3 = \{t_0, t_4\}$. Indeed $\succeq = (\{t_1\}, \{t_2, t_3, t_5, t_6, t_7\}, \{t_0, t_4\})$. We can check that each outcome has been put in the highest possible rank in $\succeq$. Therefore, if we push an outcome to a higher rank then the preorder does not satisfy the preference set. For example, $\succeq' = (\{t_1, t_5\}, \{t_2, t_3, t_6, t_7\}, \{t_0, t_4\})$ does not satisfy $s_1 = fish >_{opt} meat$.

4.2 Pessimistic preferences

The converse reasoning is drawn when dealing with pessimistic preferences [Benferhat *et al.*, 2002]. The basic principle is that outcomes are not preferred unless the contrary is stated. Indeed outcomes are put in the lowest possible rank in the preorder while being consistent with preferences at hand. This principle also ensures that the complete preorder is unique and the most compact one among preorders satisfying the set of preferences[3]. Algorithm 2 gives the way this preorder is computed.

Algorithm 2: A complete preorder associated with $\mathcal{P}_{>_{pes}}$.

```
Data: A preference set $\mathcal{P}_{>_{pes}}$.
Result: A complete preorder $\succeq$ on $\Omega$.
begin
    $l = 0$
    while $\Omega \neq \emptyset$ do
        $l = l + 1$
        $E_l = \{t | t \in \Omega, \nexists (L(s_i), R(s_i)) \in \mathcal{L}(\mathcal{P}_{>_{pes}}), t \in L(s_i)\}$
        if $E_l = \emptyset$ then
            stop (inconsistent preferences), $l = l - 1$
        - $\Omega = \Omega \backslash E_l$
        /** remove satisfied preferences **/
        - remove $(L(s_i), R(s_i))$ where $R(s_i) \cap E_l \neq \emptyset$
    return $\succeq = (E'_1, \cdots, E'_l)$ s.t. $0 \leq h \leq l$, $E'_h = E_{l-h+1}$
end
```

Example 3 *(Example 1 con'd)*
We have $E_1 = \{t_4, t_5, t_6\}$. We remove C_1 and C_2 since $s_1 : fish >_{pes} meat$ and $s_2 : red \wedge cake >_{pes} white \wedge ice\_cream$ are satisfied. We repeat the same reasoning and get $E_2 = \{t_2, t_3, t_7\}$ and $E_3 = \{t_0, t_1\}$. So $\succeq = (\{t_0, t_1\}, \{t_2, t_3, t_7\}, \{t_4, t_5, t_6\})$. We can check that each outcome has been put in the lowest possible rank in the preorder.

4.3 Strong preferences

Strong preferences induce a unique partial order on outcomes. We can use both construction principles used in optimistic and pessimistic preferences to linearize the partial order and compute a unique complete preorder. Algorithms 1 and 2 can be adapted to deal with strong preferences. For brevity, we only give the algorithm adapting Algorithm 1.

Similarly, the adaptation of Algorithm 2 consists in the following modification:

replace *"remove $(L(s_i), R(s_i))$ where $R(s_i) \cap E_l \neq \emptyset$"* by *["replace $(L(s_i), R(s_i))$ by $(L(s_i), R(s_i) \backslash E_l)$" and "remove $(L(s_i), R(s_i))$ with empty $R(s_i)$"].*

Example 4 *(Example 1 con'd)*
There is no complete preorder which satisfies $\mathcal{P}_{>_{st}}$ so $\mathcal{P}_{>_{st}}$ is inconsistent. This is due to s_1 and s_2. Following s_1, t_0 is preferred to t_7 while t_7 is preferred to t_0 following s_2.

Example 5 *(Consistent strong preferences)*
Let $\mathcal{P}_{>_{st}} = \{fish \wedge white >_{st} fish \wedge red, red \wedge cake >_{st} red \wedge ice\_cream, meat \wedge red >_{st} meat \wedge white\}$. Then following Algorithm 3, we have $\succeq = (\{t_0, t_1, t_7\}, \{t_3\}, \{t_2, t_6\}, \{t_4, t_5\})$. Now following the adaptation of Algorithm 2 to deal with strong preferences, we have $\succeq = (\{t_0, t_1\}, \{t_3, t_7\}, \{t_6\}, \{t_2, t_4, t_5\})$.

[2]Technically speaking, this preorder can be obtained by max-based aggregation operator of all preorders satisfying the set of preferences

[3]Technically speaking, this preorder can be obtained by min-based aggregation operator of all preorders satisfying the set of preferences.

Algorithm 3: A complete preorder associated with $\mathcal{P}_{>_{st}}$.

```
Data: A preference set $\mathcal{P}_{>_{st}}$.
Result: A complete preorder $\succeq$ on $\Omega$.
begin
    $l \leftarrow 0$
    while $\Omega \neq \emptyset$ do
        $l = l + 1$
        $E_l = \{t | t \in \Omega, \nexists (L(s_i), R(s_i)) \in \mathcal{L}(\mathcal{P}_{>_{st}}), t \in R(s_i)\}$
        if $E_l = \emptyset$ then
            stop (inconsistent preferences), $l = l - 1$
        - $\Omega = \Omega \backslash E_l$
        - replace $(L(s_i), R(s_i))$ by $(L(s_i) \backslash E_l, R(s_i))$
        /** remove satisfied preferences **/
        - remove $(L(s_i), R(s_i))$ where $L(s_i) = \emptyset$
    return $\succeq = (E_1, \cdots, E_l)$.
end
```

Notice that optimistic (resp. pessimistic) preferences are a right hand (resp. left hand) side weakening of strong preferences in the sense that "any outcome satisfying $p \wedge \neg q$" (resp. all outcomes satisfying $\neg p \wedge q$) for the preference $p > q$ is weakened to "at least one outcome satisfying $p \wedge \neg q$" (resp. at least one outcome satisfying $\neg p \wedge q$).

4.4 Ceteris paribus preferences

These preferences are similar to strong preferences. They also induce a unique partial order on outcomes. We can also use both construction principles used in optimistic and pessimistic semantics to compute a unique complete preorder.

Example 6 *(Example 1 con'd)*
Following the gravitation towards the ideal we have $\succeq = (\{t_1\}, \{t_3, t_5\}, \{t_0, t_7\}, \{t_2, t_4\}, \{t_6\})$ *while following the gravitation towards the worst we have* $\succeq = (\{t_1\}, \{t_3\}, \{t_0\}, \{t_2, t_7\}, \{t_4, t_5, t_6\})$.

Note that the pairs associated to ceteris paribus preferences need to be pre-processed. In fact, these preferences express that any outcome in the left hand side of the pair $(L(s_i), R(s_i))$ is preferred to any outcome in the right hand side of the pair following ceteris paribus semantics. Indeed when an outcome in the left (resp. right) hand side of a pair does not have its associated outcome in the right (resp. left) hand side of the pair following ceteris paribus semantics then it should be removed from the pair. This situation occurs when the set of outcomes is incomplete.

Example 7 *Let* dish, wine *and* dessert *be three propositional variables taking their values in* $\{fish, meat\}$, $\{white, red\}$ *and* $\{cake, ice\_cream\}$ *respectively. Let* $\Omega = \{t_0, t_1, t_2\}$ *with* $t_0 = fish - white - ice\_cream$, $t_1 = fish - red - ice\_cream$ *and* $t_2 = fish - red - cake$. *Let* $\mathcal{P}_{>_{cp}} = \{s_1 : fish >_{cp} meat, s_2 : fish \wedge white >_{cp} fish \wedge red\}$. *Then* $\mathcal{L}(\mathcal{P}_{>_{cp}}) = \{C_1 = (\{t_0, t_1, t_2\}, \{\}), C_2 = (\{t_0\}, \{t_1, t_2\})\}$. C_1 *is irrelevant and should be removed. Regarding* C_2*,* t_2 *has not its associated outcome in* $L(s_2)$ *following ceteris paribus semantics and should be removed. Indeed we have* $\mathcal{L}(\mathcal{P}_{>_{cp}}) = \{C_2 = (\{t_0\}, \{t_1\})\}$.

5 Experimental Study

Our main objective in this section is to evaluate the psychological plausibility of strong, optimistic, pessimistic and ceteris paribus semantics. In order to reach this objective, we have conducted a psychological experiment devoted to collect sets of comparative preferences formulated by participants to this experiment, and the associated models (a (pre)order on the set of outcomes). The adopted methodology and main results are presented in the next subsections.

5.1 Method

Participants

Thirty first-year psychology students at the University of Toulouse-Le Mirail, all native French speakers, contributed to this study. None of them had previously received any formal logical training or any course on preferences. Note that our objective is not to study participant's real preferred menus. Such an objective would necessitate a much more large number of participants, like in opinion studies. Rather, our objective is to compare statistically the fit of the semantics under study with human preference's judgments. For such an objective, our sample size is sufficient according to scientific standards.

Material and procedure

Comparative preference judgments were collected via a booklet where subjects were asked to suppose that they are at the restaurant and they must compose their menu. In the first page of the booklet, they were asked to compare and to rank-order the following objects (unranked objects where skipped from analyses):

t_0: fish-white-ice-cream

t_1: fish-white-cake

t_2: fish-red-ice-cream

t_3: fish-red-cake

t_4: meat-white-ice-cream

t_5: meat-white-cake

t_6: meat-red-ice-cream

t_7: meat-red-cake.

Next, they were asked to compare the 31 pairs of menus given in Table 1. An object o_1 can be preferred to an object o_2 or o_2 preferred to o_1, or be both equally preferred, or be incomparable. Answers of the kinds "equally preferred" or "incomparable" have been discarded from analysis.

Rationale

Given participant's comparative preference judgments, for each participant, we computed the set of compact preferences consistent with participant's preferences. The set of possible comparative preferences is given in Table 2. For a given participant, a comparative preference is retained as compact if it is consistent with all her/his preferred menus (see Table 1).

Next, given these compact preferences and the algorithms provided in the paper, for each participant, four preorders have been inferred according to the principles underling the

$(white, red)$
$(meat, fish)$
$(ice-cream, cake)$
$(meat-white-ice-cream, meat-white-cake)$
$(meat-white-ice-cream, meat-red-ice-cream)$
$(meat-white-ice-cream, meat-red-cake)$
$(meat-white-ice-cream, fish-white-ice-cream)$
$(meat-white-ice-cream, fish-white-cake)$
$(meat-white-ice-cream, fish-red-ice-cream)$
$(meat-white-ice-cream, fish-red-cake)$
$(meat-white-cake, meat-red-ice-cream)$
$(meat-white-cake, meat-red-cake)$
$(meat-white-cake, fish-white-ice-cream)$
$(meat-white-cake, fish-white-cake)$
$(meat-white-cake, fish-red-ice-cream)$
$(meat-white-cake, fish-red-cake)$
$(meat-red-ice-cream, meat-red-cake)$
$(meat-red-ice-cream, fish-white-ice-cream)$
$(meat-red-ice-cream, fish-white-cake)$
$(meat-red-ice-cream, fish-red-ice-cream)$
$(meat-red-ice-cream, fish-red-cake)$
$(meat-red-cake, fish-white-ice-cream)$
$(meat-red-cake, fish-white-cake)$
$(meat-red-cake, fish-red-ice-cream)$
$(meat-red-cake, fish-red-cake)$
$(fish-white-ice-cream, fish-white-cake)$
$(fish-white-ice-cream, fish-red-ice-cream)$
$(fish-white-ice-cream, fish-red-cake)$
$(fish-white-cake, fish-red-ice-cream)$
$(fish-white-cake, fish-red-cake)$
$(fish-red-ice-cream, fish-red-cake)$

Table 1: Pairs of menus participants have to compare.

$white$	$>$	red
red	$>$	$white$
$meat$	$>$	$fish$
$fish$	$>$	$meat$
$ice-cream$	$>$	$cake$
$cake$	$>$	$ice-cream$
$white \wedge meat$	$>$	$white \wedge fish$
$white \wedge fish$	$>$	$white \wedge meat$
$white \wedge ice-cream$	$>$	$white \wedge cake$
$white \wedge cake$	$>$	$white \wedge ice-cream$
$red \wedge meat$	$>$	$red \wedge fish$
$red \wedge fish$	$>$	$red \wedge meat$
$red \wedge ice-cream$	$>$	$red \wedge cake$
$red \wedge cake$	$>$	$red \wedge ice-cream$
$meat \wedge ice-cream$	$>$	$meat \wedge cake$
$meat \wedge cake$	$>$	$meat \wedge ice-cream$
$fish \wedge ice-cream$	$>$	$fish \wedge cake$
$fish \wedge cake$	$>$	$fish \wedge ice-cream$

Table 2: Set of a priori possible compact preferences.

inferential machinery of the four studied semantics. For evaluating the psychological relevance of the semantics under study, the key comparison is between participant's (pre)order on the 8 menus $\{t_0, \cdots, t_7\}$ and (pre)orders computed according to the four semantics given participant's compact preferences. Two cues have been used for ordering semantics according to their psychological relevance: *The percentages of cases* where the semantics provide an inconsistent set of models; and *the distance* and *mean distance* between ranks allowed by participants and models to the 8 menus.

- *Percentages of inconsistency:* For each semantics, we computed the number of cases where it produces an inconsistent set of models given inferred participant's compact preferences. The percentage is then computed by the classical division of this number by the number (30) of participants $(\times 100)$. A semantics better fits psychological data if it allows producing a consistent set of models from participant's compact preferences.

- *Mean Distances:* Two distances based on participants and semantics orders have been computed. In both cases, distances are computed from the ranks attributed to each of the 8 menus by participants and semantics. Several menus can have the same rank. Suppose participant 1 prefers the menu "meat, red wine, ice-cream", if this menu is also the preferred one for a given semantics, then the distance is zero. If only one menu is more preferred, then the rank is 1, and so on. A semantics better fits psychological data if the rank it attributes is closer to the menu preferred by participants. At the sample level, a semantics better fits psychological data if the mean distance between participants's preferred models and the rank (or level) attribute by the semantics is smaller. This methodology can be generalized, and the same calculus can be made for each menu involved in participant's ranking (note that this order doesn't necessarily involve the 8 proposed menus). So, for each participant, it is possible to compute the mean of the distance between each menu and ranks predicted by semantics. Next, the mean of these means is computed. As before, a small mean means a better fit.

In order to conclude at the inferential level (and not only at the descriptive level), cognitive psychology, exactly as other experimental sciences, makes use of statistical tools for hypothesis testing. The student's t-test is a well known parametrical statistical test that allows testing the null hypothesis that two means are not different. The probability p provided by the test express the risk (called alpha) that we reject by error the null hypothesis. In social and human sciences, it is usual to consider that this risk is acceptable at the level $.05$, that is, if p is greater than $.05$, we cannot reject the null hypothesis without a significant risk. Under $.05$, we reject the null hypothesis, and so accept the hypothesis of the difference between the two means. In our analyses, when a difference between means is significant ($p =< .05$), it is interpreted as: the semantics exhibiting the less mean distance significantly fits better human data than the other semantics.

	cp	*str.*	*pess.*	*opt.*
% inconsistency	36.6	10	0	0
Distance between participants and models′ levels for participant preferred outcome. (standard deviation)	1.5 (.81) $n =$ 26	.83 (.93) $n =$ 29	.62 (.71) $n =$ 30	.55 (.65) $n =$ 30
Means of the mean distance between participants and models′ outcome levels (standard deviation) $n = 19$	2.44 (.63)	2.46 (.65)	2.54 (.71)	2.53 (.66)

Table 3: Cues for evaluation of the fit of semantics with participant's preference judgment. "cp", "str", "pess." and "opt." stand respectively for ceteris paribus, strong, pessimistic and optimistic.

5.2 Results

Participant's answers allowed to compute a set of compact preferences containing between 3 and 7 compact preferences out of 18 a priori ones. Table 3 shows that the ceteris paribus semantics doesn't fit participant's orders in 36% of the cases and that the strong semantics failed in 10% of the cases, while optimistic and pessimistic semantics provide always a consistent set of preferences. This order is confirmed by comparisons of distances between participants and semantics' levels for participant preferred outcome in Table 3. Table 3 also shows that the optimistic semantics has a better fit than the pessimistic one (however mean's comparison by Student's t-test is not significant at the .05 level: $t = -1.43$, $df = 29$, $p = .16$) while the latter has a better fit than the strong semantics ($t = 2.7$, $df = 28$, $p = .01$, significant) which better fits participant's data than ceteris paribus semantics ($t = -5.7$, $df = 24$, $p < .001$, significant). These results are broadly confirmed by the comparison of the means of the mean distance between participants and semantics (pre)orders. Table 3 confirms previous results. Indeed, statistical comparisons by Student's t-test shows a significant difference between strong and pessimistic semantics ($t = -2.37$, $df = 18$, $p = .036$) but not between ceteris paribus and strong, and pessimistic and optimistic semantics. This result confirms that two distinct sets of semantics can be distinguished from their psychological relevance: Pessimistic and optimistic semantics on one hand, and strong and ceteris paribus on the other one. Except for percentages, more the values are low, better is the fit.

As such, given all the information summarized in table 3, it appears that optimistic and pessimistic semantics are more plausible psychologically than strong and ceteris paribus semantics. Note that the latter exhibits a bad fit of participants' judgments.

6 Conclusion

We focused on comparative preference statements and distinguished different non-monotonic semantics that have been studied in literature. So far, researchers have argued for a semantics or another from purely theoretical standpoint (also philosophical for ceteris paribus semantics) or for modeling a specific application. In this paper, we explored another dimension, namely psychological plausibility, to compare the semantics.

This work gives an indication about human behavior when interpreting comparative preferences. Our results suggest that pessimistic and optimistic semantics better fit human preferences organization and inference than ceteris paribus and strong semantics. Neverthless, it doesn't mean that every human in every situation would "prefer" according to the principles underling these semantics. Rather, it suggests that in familiar domains, a population (students in psychology) known as representative of global occidental people, "prefer" in a manner more closed to pessimistic and optimistic semantics. It doesn't mean that strict and ceteris paribus semantics should be rejected. Psychological plausibility is not of course the sole criterion for evaluating formal models in AI, but it is a criterion, every time a model could be in cognitive contact with human mind, trough human-machines interactions and communication, and every time it could have incidences in human adaptation, including cognitive comfort and efficiency.

This first attempt opens the door to more ambitious and deeper comparison of preference representations. In a future work we intend to perform a comparison of the main different compact representations of preferences such as CP-nets [Boutilier *et al.*, 2004], QCL [Brewka *et al.*, 2004], etc.

References

[Benferhat *et al.*, 1992] S. Benferhat, D. Dubois, and H. Prade. Representing default rules in possibilistic logic. In *Proceedings of 3rd International Conference of Principles of Knowledge Representation and Reasoning (KR'92)*, pages 673–684, 1992.

[Benferhat *et al.*, 2002] S. Benferhat, D. Dubois, S. Kaci, and H. Prade. Bipolar possibilistic representations. In *18th International Conference on Uncertainty in Artificial Intelligence (UAI'02)*, pages 45–52, 2002.

[Benferhat *et al.*, 2004] S. Benferhat, J.F. Bonnefon, and R. Da Silva Neves. An experimental analysis of possibilistic default reasoning. In *9th International Conference on Principles of Knowledge Representation and Reasonning (KR'04)*, pages 130–140, 2004.

[Benferhat *et al.*, 2005] S. Benferhat, J.F. Bonnefon, and R. Da Silva Neves. An overview of possibilistic handling of default reasoning: an experimental study. *Synthese*, 146(1):53–70, 2005.

[Boutilier *et al.*, 1999] C. Boutilier, R.I. Brafman, H.H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *UAI*, pages 71–80, 1999.

[Boutilier *et al.*, 2004] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

[Boutilier, 1994] C. Boutilier. Toward a logic for qualitative decision theory. In *4th International Conference on Principles of Knowledge Representation, (KR'94)*, pages 75–86, 1994.

[Brewka *et al.*, 2004] G. Brewka, S. Benferhat, and D. Le Berre. Qualitative choice logic. *Artificial Intelligence*, 157(1-2):203–237, 2004.

[Chomicki, 2003] J. Chomicki. Preference formulas in relational queries. *ACM Transactions on Database Systems*, 28:1–40, 2003.

[Hansson, 1996] S.O. Hansson. What is ceteris paribus preference? *Journal of Philosophical Logic*, 25:307–332, 1996.

[Kraus *et al.*, 1990] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2):167–207, 1990.

[Neves *et al.*, 2002] R. Da Silva Neves, J.F. Bonnefon, and E. Raufaste. An empirical test of patterns for nonmonotonic inference. *Annals of Mathematics and Artificial Intelligence*, 34(1-3):107–130, 2002.

[Pearl, 1990] J. Pearl. System Z: A natural ordering of defaults with tractable applications to default reasoning. In *Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge (TARK'90)*, pages 121–135, 1990.

[van der Torre and Weydert, 2001] L. van der Torre and E. Weydert. Parameters for utilitarian desires in a qualitative decision theory. *Applied Intelligence*, 14(3):285–301, 2001.

[von Wright, 1963] G. H. von Wright. *The Logic of Preference*. University of Edinburgh Press, 1963.

[Wilson, 2004] N. Wilson. Extending CP-nets with stronger conditional preference statements. In *19th National Conference on Artificial Intelligence (AAAI'04)*, pages 735–741, 2004.

[Yager, 1983] R.R. Yager. Entropy and specificity in a mathematical theory of evidence. *International Journal of General Systems*, 9:249–260, 1983.

Manifold Answer-Set Programs for Meta-Reasoning[*]

Wolfgang Faber
University of Calabria, Italy
wf@wfaber.com

Stefan Woltran
Vienna University of Technology, Austria
woltran@dbai.tuwien.ac.at

Abstract

In answer-set programming (ASP), the main focus usually is on computing answer sets which correspond to solutions to the problem represented by a logic program. Simple reasoning over answer sets is sometimes supported by ASP systems (usually in the form of computing brave or cautious consequences), but slightly more involved reasoning problems require external postprocessing. Generally speaking, it is often desirable to use (a subset of) brave or cautious consequences of a program P_1 as input to another program P_2 in order to provide the desired solutions to the problem to be solved. So far, the evaluation of the program P_1 has to be decoupled from the evaluation of P_2 using an intermediate step which collects the desired consequences of P_1 and provides them as input to P_2. In this work, we present a novel method for representing such a procedure within a *single* program, and thus within the realm of ASP itself. Our technique relies on rewriting P_1 into a so-called *manifold program*, which allows for accessing all desired consequences of P_1 within a single answer set. Then, this manifold program can be evaluated jointly with P_2 avoiding any intermediate computation step. For determining the consequences within the manifold program we use *weak constraints*, which is strongly motivated by complexity considerations. As an application, we present an encoding for computing the ideal extension of an abstract argumentation framework.

1 Introduction

In the last decade, *Answer Set Programming* (ASP) [Marek and Truszczyński, 1999; Niemelä, 1999], also known as A-Prolog [Baral, 2002; Gelfond, 2002], has emerged as a declarative programming paradigm which combines techniques stemming from databases, logic programming and non-monotonic reasoning. ASP is well suited for modelling and solving problems which involve common sense reasoning, and has been fruitfully applied to a wide variety of applications including diagnosis, data integration, configuration, and many others. Moreover, the efficiency of the latest tools for processing ASP programs (so-called ASP solvers) reached a state that makes them applicable for problems of practical importance [Gebser *et al.*, 2007].

The basic idea of ASP is to obtain solutions to a problem as the answer sets (usually stable models) of a logic program, which consists of rules and constraints that define necessary and sufficient properties of the solutions. The program is then input into an ASP solver, which computes the answer set(s) of the program, from which the solutions of the problem can be read.

However, frequently one is interested less in the solutions per se, but rather in reasoning tasks that have to take some or even all solutions into account. As an example, consider the problem of database repair, in which a given database instance does not satisfy some of the constraints imposed in the database. One can attempt to modify the data in order to obtain a consistent database by changing as little as possible. This will in general yield multiple possibilities and can be encoded conveniently using ASP (see, e.g., [Bravo and Bertossi, 2003]). However, usually one is not interested in the repairs themselves, but in the data which is present in *all* repairs. For the ASP encoding, this means that one is interested in the elements which occur in all answer sets; these are also known as *cautious consequences*. Indeed, ASP systems provide special interfaces for computing cautious consequences by means of query answering. But sometimes one has to do more, such as answering a complex query over the cautious consequences or simply counting them. So far, ASP solvers provide no support for such tasks, which therefore have to be done outside ASP systems, which hampers usability and limits the potential of ASP.

In this work, we tackle this limitation by providing a technique, which transforms an ASP program P into a *manifold program* M_P which we use to identify all consequences of a certain type[1] within a *single* answer set. The main advantage of the manifold approach is that the resulting program can be extended by additional rules representing a query over the brave (or cautious, definite) consequences of the original program P, thereby using ASP itself for this additional reasoning. In order to identify the consequences, we use *weak constraints* [Buccafurri *et al.*, 2000], which are supported by the ASP-solver DLV [Leone *et al.*, 2006]. Weak constraints have been introduced to prefer a certain subset of answer sets

[*]This work was supported by the Vienna Science and Technology Fund (WWTF) under grant ICT08-028 and by M.I.U.R. within the Italia-Austria internazionalization project "Sistemi basati sulla logica per la rappresentazione di conoscenza: estensioni e tecniche di ottimizzazione.".

[1]We consider here the aforementioned concepts of brave and cautious consequence, and definite consequence [Saccà, 1996].

via penalization. Their use for computing consequences is justified by a complexity-theoretic argument: One can show that computing consequences is complete for the complexity classes $\mathrm{FP}_{||}^{\mathrm{NP}}$ or $\mathrm{FP}_{||}^{\Sigma_2^P}$ (depending on the presence of disjunction), for which also computing answer sets for programs with weak constraints is complete[2], which means that an equivalent compact ASP program without these extra constructs most likely does not exist. In principle, other preferential constructs similar to weak constraints could be used as well for our purposes, as long as they meet these complexity requirements.

We discuss two particular applications of the manifold approach. First, we specify an encoding which decides the SAT-related *unique minimal model problem*, which is closely related to closed-world reasoning [Reiter, 1978]. The second problem stems from the area of argumentation (cf. [Bench-Capon and Dunne, 2007] for an overview) and concerns the computation of the ideal extension [Dung *et al.*, 2007] of an argumentation framework. For both problems we make use of manifold programs of well-known encodings (computing all models of a CNF-formula for the former application, computing all admissible extensions of an argumentation framework for the latter) in order to compute consequences. Extensions by a few more rules then directly provide the desired solutions, requiring little effort in total.

Organization and Main Results. After introducing the necessary background in the next section, we

- introduce in Section 3 the concept of a manifold program for rewriting propositional programs in such a way that all brave (resp. cautious, definite) consequences of the original program are collected into a single answer set;
- lift the results to the non-ground case (Section 4); and
- present applications for our technique in Section 5. In particular, we provide an ASP encoding for computing the ideal extension of an argumentation framework.

The paper concludes with a brief discussion of related and further work.

2 Preliminaries

In this section, we review the basic syntax and semantics of ASP with weak constraints, following [Leone *et al.*, 2006], to which we refer for a more detailed definition.

An *atom* is an expression $p(t_1,\ldots,t_n)$, where p is a *predicate* of arity $\alpha(p) = n \geq 0$ and each t_i is either a variable or a constant. A *literal* is either an atom a or its negation $\mathrm{not}\ a$. A *(disjunctive) rule* r is of the form

$$a_1 \ \vee\ \cdots\ \vee\ a_n \ \text{:-}\ b_1, \ldots, b_k,\ \mathrm{not}\ b_{k+1}, \ldots,\ \mathrm{not}\ b_m$$

with $n \geq 0$, $m \geq k \geq 0$, $n + m > 0$, and where $a_1, \ldots, a_n, b_1, \ldots, b_m$ are atoms.

The *head* of r is the set $H(r) = \{a_1, \ldots, a_n\}$, and the *body* of r is $B(r) = \{b_1, \ldots, b_k,\ \mathrm{not}\ b_{k+1}, \ldots,\ \mathrm{not}\ b_m\}$. Furthermore, $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. We will sometimes denote a rule r as $H(r)$:- $B(r)$.

A *weak constraint* [Buccafurri *et al.*, 2000] is an expression wc of the form

$$:\sim\ b_1, \ldots, b_k,\ \mathrm{not}\ b_{k+1}, \ldots,\ \mathrm{not}\ b_m.\ [w : l]$$

where $m \geq k \geq 0$ and $b_1, \ldots, b_m$ are literals, while $weight(wc) = w$ (the *weight*) and l (the *level*) are positive integer constants or variables. For convenience, w and/or l may be omitted and are set to 1 in this case. The sets $B(wc)$, $B^+(wc)$, and $B^-(wc)$ are defined as for rules. We will sometimes denote a weak constraint wc as $:\sim B(wc)$.

A *program* P is a finite set of rules and weak constraints. $Rules(P)$ denotes the set of rules and $WC(P)$ the set of weak constraints in P. w^P_{max} and l^P_{max} denote the maximum weight and maximum level over $WC(P)$, respectively. A program (rule, atom) is *propositional* or *ground* if it does not contain variables. A program is called *strong* if $WC(P) = \emptyset$, and *weak* otherwise.

For any program P, let U_P be the set of all constants appearing in P (if no constant appears in P, an arbitrary constant is added to U_P); let B_P be the set of all ground literals constructible from the predicate symbols appearing in P and the constants of U_P; and let $Ground(P)$ be the set of rules and weak constraints obtained by applying, to each rule and weak constraint $r \in P$, all possible substitutions from the variables in P to elements of U_P. U_P is usually called the *Herbrand Universe* of P and B_P the *Herbrand Base* of P.

A ground rule r is *satisfied* by a set I of ground atoms iff $H(r) \cap I \neq \emptyset$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$. I satisfies a ground program P, if each $r \in P$ is satisfied by I. For non-ground P, I satisfies P iff I satisfies $Rules(Ground(P))$. A ground weak constraint wc is *violated* by I, iff $B^+(wc) \subseteq I$ and $B^-(wc) \cap I = \emptyset$; it is satisfied otherwise.

Following [Gelfond and Lifschitz, 1991], a set $I \subseteq B_P$ of atoms is an *answer set* for a strong program P with $WC(P) = \emptyset$ iff it is a subset-minimal set that satisfies the *reduct*

$$P^I = \{H(r) \text{ :- } B^+(r) \mid I \cap B^-(r) = \emptyset, r \in Ground(P)\}.$$

A set of atoms $I \subseteq B_P$ is an *answer set* for a weak program P with $WC(P) \neq \emptyset$ iff I is an answer set of $Rules(P)$ and $H^{Ground(P)}(I)$ is minimal among all the answer sets of $Rules(P)$, where the penalization function $H^P(I)$ for weak constraint violation of a ground program P is defined as follows:

$$\begin{aligned} H^P(I) &= \textstyle\sum_{i=1}^{l^P_{max}} \left(f_P(i) \cdot \sum_{w \in N^P_i(I)} weight(w) \right) \\ f_P(1) &= 1, \text{ and} \\ f_P(n) &= f_P(n-1) \cdot |WC(P)| \cdot w^P_{max} + 1 \ \text{ for } n > 1. \end{aligned}$$

where $N^P_i(I)$ denotes the set of weak constraints of P in level i violated by I. For any program P, we denote the set of its answer sets by $AS(P)$. In this paper, we use only weak

[2]The first of these results is fairly easy to see, for the second, Buccafurri *et al.* [2000] have shown that the related decision problem is complete for the class Θ_2^P or Θ_3^P, from which the $\mathrm{FP}_{||}^{\mathrm{NP}}$ and $\mathrm{FP}_{||}^{\Sigma_2^P}$ results can be obtained. Also note that frequently cited NP, Σ_2^P, and co-NP, Π_2^P completeness results hold for brave and cautious query answering, respectively, but not for computing brave and cautious consequences.

constraints with weight and level 1, for which $H^{Ground(P)}(I)$ amounts to the number of weak constraints violated in I.

A ground atom a is a *brave* (sometimes also called credulous or possible) consequence of a program P, denoted $P \models_b a$, if $a \in A$ holds for at least one $A \in AS(P)$. A ground atom a is a *cautious* (sometimes also called skeptical or certain) consequence of a program P, denoted $P \models_c a$, if $a \in A$ holds for all $A \in AS(P)$. A ground atom a is a *definite* consequence [Saccà, 1996] of a program P, denoted $P \models_d a$, if $AS(P) \neq \emptyset$ and $a \in A$ holds for all $A \in AS(P)$. The sets of all brave, cautious, definite consequences of a program P are denoted as $BC(P)$, $CC(P)$, $DC(P)$, respectively.

3 Propositional Manifold Programs

In this section, we present a translation which essentially creates a copy of a given strong propositional program for each of (resp. for a subset of) its atoms. Thus, we require several copies of the alphabet used by the given program.

Definition 3.1 *Given a set I of literals, a collection $\mathcal{I}$ of sets of literals, and an atom a, define $I^a = \{p^a \mid$ atom $p \in I\} \cup \{\text{not } p^a \mid \text{not } p \in I\}$ and $\mathcal{I}^a = \{I^a \mid I \in \mathcal{I}\}$.*

The actual transformation to a manifold is given in the next definition. We copy a given program P for each atom a in a given set S, whereby the transformation guarantees the existence of an answer set by enabling the copies conditionally.

Definition 3.2 *For a strong propositional program P and a set $S \subseteq B_P$, define its* manifold *as*

$$P^{tr}_S = \bigcup_{r \in P} r^{tr}_S \cup \{c \text{ :- not } i \;\; ; \;\; i \text{ :- not } c\}$$

where

$$r^{tr}_S = \{H(r)^a \text{ :- } \{c\} \cup B(r)^a \mid a \in S\}$$

and without loss of generality $B_P \cap B_{P^{tr}_S} = \emptyset$, so all symbols in P^{tr}_S are assumed to be fresh.

Example 3.3 *Consider $\Phi = \{p \vee q \text{ :- } \;\; ; \;\; r \text{ :- } p \;\; ; \;\; r \text{ :- } q\}$ for which $AS(\Phi) = \{\{p, r\}, \{q, r\}\}$, $BC(\Phi) = \{p, q, r\}$ and $CC(\Phi) = DC(\Phi) = \{r\}$. When forming the manifold for $B_\Phi = \{p, q, r\}$, we obtain*

$$\Phi^{tr}_{B_\Phi} = \left\{ \begin{array}{l} p^p \vee q^p \text{ :- } c \;\; ; \;\; r^p \text{ :- } c, p^p \;\; ; \;\; r^p \text{ :- } c, q^p \\ p^q \vee q^q \text{ :- } c \;\; ; \;\; r^q \text{ :- } c, p^q \;\; ; \;\; r^q \text{ :- } c, q^q \\ p^r \vee q^r \text{ :- } c \;\; ; \;\; r^r \text{ :- } c, p^r \;\; ; \;\; r^r \text{ :- } c, q^r \\ c \text{ :- not } i \;\; ; \;\; i \text{ :- not } c \end{array} \right\}$$

Note that given a strong program P and set $S \subseteq B_P$, the construction of P^{tr}_S can be done in polynomial time (w.r.t. the size of P). The answer sets of the transformed program consist of all combinations (of size $|S|$) of answer sets of the original program (augmented by c) plus the special answer set $\{i\}$ which we shall use to indicate inconsistency of P.

Proposition 3.4 *For a strong propositional program P and a set $S \subseteq B_P$, $AS(P^{tr}_S) = A \cup \{\{i\}\}$, where*

$$A = \{\bigcup_{i=1}^{|S|} A_i \cup \{c\} \mid \langle A_1, \ldots, A_{|S|} \rangle \in \prod_{a \in S} AS(P)^a\}.$$

Note that $\prod$ denotes the Cartesian product in Proposition 3.4.

Example 3.5 *For Φ of Example 3.3, we obtain that $AS(\Phi^{tr}_{B_\Phi})$ consists of $\{i\}$ plus*

$$\begin{array}{l} \{c, p^p, r^p, p^q, r^q, p^r, r^r\}, \{c, \underline{q^p, r^p}, p^q, r^q, p^r, r^r\}, \\ \{c, p^p, r^p, \underline{q^q, r^q}, p^r, r^r\}, \{c, p^p, r^p, p^q, r^q, \underline{q^r, r^r}\}, \\ \{c, \underline{q^p, r^p}, \underline{q^q, r^q}, p^r, r^r\}, \{c, \underline{q^p, r^p}, p^q, r^q, \underline{q^r, r^r}\}, \\ \{c, p^p, r^p, \underline{q^q, r^q}, \underline{q^r, r^r}\}, \{c, \underline{q^p, r^p}, \underline{q^q, r^q}, \underline{q^r, r^r}\}. \end{array}$$

The underlined parts are used to highlight the copies of the original answer sets $\{q, r\}$ compared to the copies of the other original answer sets $\{p, r\}$.

Using this transformation, each answer set encodes an association of an atom with some answer set of the original program. If an atom a is a brave consequence of the original program, then a witnessing answer set exists, which contains the atom a^a. The idea is now to prefer those atom-answer set associations where the answer set is a witness. We do this by means of weak constraints and penalize each association where the atom is not in the associated answer set, that is, where a^a is not in the answer set of the transformed program. Doing this for each atom means that an optimal answer set will not contain a^a only if there is no answer set of the original program that contains a, so each a^a contained in an optimal answer set is a brave consequence of the original program.

Definition 3.6 *Given a strong propositional program P and a set $S \subseteq B_P$, let*

$$P^{bc}_S = P^{tr}_S \cup \{:\sim \text{not } a^a \mid a \in S\} \cup \{:\sim i\}$$

Observe that all weak constraints are violated in the special answer set $\{i\}$, while in the answer set $\{c\}$ (which occurs if the original program has an empty answer set) all but $:\sim i$ are violated. The following result would also hold without $:\sim i$ being included.

Proposition 3.7 *Given a strong propositional program P and a set $S \subseteq B_P$, we have, for any $A \in AS(P^{bc}_S)$, $\{a \mid a^a \in A\} = BC(P) \cap S$.*

Example 3.8 *For the program Φ as given Example 3.3, $\Phi^{bc}_{B_\Phi}$ is given by*

$$\Phi^{tr}_{B_\Phi} \cup \{:\sim \text{not } p^p \;\; ; \;\; :\sim \text{not } q^q \;\; ; \;\; :\sim \text{not } r^r \;\; ; \;\; :\sim i\}.$$

We obtain that $AS(\Phi^{bc}_{B_\Phi}) = \{A_1, A_2\}$, where

$$\begin{array}{rcl} A_1 & = & \{c, p^p, r^p, q^q, r^q, p^r, r^r\}; \\ A_2 & = & \{c, p^p, r^p, q^q, r^q, q^r, r^r\}, \end{array}$$

as these two answer sets are the only ones that violate no weak constraint. We can observe that $\{a \mid a^a \in A_1\} = \{a \mid a^a \in A_2\} = \{p, q, r\} = BC(\Phi)$.

Concerning cautious consequences, we first observe that if a program is inconsistent (in the sense that it does not have any answer set), each atom is a cautious consequence. But if P is inconsistent, then P^{tr}_S will have only $\{i\}$ as an answer set, so we will need to find a suitable modification in order to deal with this in the correct way. In fact, we can use a similar approach as for brave consequences, but penalize those associations where an atom is contained in its associated answer set. Any optimal answer set will thus contain a^a for an atom only if a is contained in each answer set. If an answer set containing i exists, it is augmented by all atoms a^a, which also causes all weak constraints to be violated.

Definition 3.9 *Given a strong propositional program P and a set $S \subseteq B_P$, let*

$$P_S^{cc} = P_S^{tr} \cup \{:\sim a^a \mid a \in S\} \cup \{a^a :\text{-}\ i \mid a \in S\} \cup \{:\sim i\}$$

As for P_S^{bc}, the following result also holds without including $:\sim i$.

Proposition 3.10 *Given a strong propositional program P and a set $S \subseteq B_P$, we have, for any $A \in AS(P_S^{cc})$, $\{a \mid a^a \in A\} = CC(P) \cap S$.*

Example 3.11 *Recall program Φ from Example 3.3. We have*

$$\begin{aligned}\Phi_{B_\Phi}^{cc} \quad = \quad & \Phi_{B_\Phi}^{tr} \cup \{:\sim p^p \ ; \ :\sim q^q \ ; \ :\sim r^r \ ; \\ & p^p :\text{-}\ i \ ; \ q^q :\text{-}\ i \ ; \ r^r :\text{-}\ i \ ; \ :\sim i\}.\end{aligned}$$

We obtain that $AS(\Phi_{B_\Phi}^{cc}) = \{A_3, A_4\}$, where

$$\begin{aligned} A_3 &= \{c, q^p, r^p, p^q, r^q, p^r, r^r\}; \\ A_4 &= \{c, q^p, r^p, p^q, r^q, q^r, r^r\}, \end{aligned}$$

as these two answer sets are the only ones that violate only one weak constraint, namely $:\sim r^r$. We observe that $\{a \mid a^a \in A_3\} = \{a \mid a^a \in A_4\} = \{r\} = CC(\Phi)$.

We next consider the notion of definite consequences. Different to cautious consequences, we do not add the annotated atoms to the answer set containing i. However, this answer set should never be among the optimal ones unless it is the only one. Therefore we inflate it by new atoms i^a, all of which incur a penalty, as does i itself. This guarantees that this answer set will incur a higher penalty ($|B_P| + 1$) than any other ($\leq |B_P|$).

Definition 3.12 *Given a strong propositional program P and a set $S \subseteq B_P$, let*

$$P_S^{dc} \quad = \quad P_S^{tr} \cup \{:\sim a^a;\ i^a :\text{-}\ i;\ :\sim i^a \mid a \in S\} \cup \{:\sim i\}$$

Proposition 3.13 *Given a strong propositional program P and a set $S \subseteq B_P$, we have, for any $A \in AS(P_S^{dc})$, $\{a \mid a^a \in A\} = DC(P) \cap S$.*

Example 3.14 *Recall program Φ from Example 3.3. We have*

$$\begin{aligned}\Phi_{B_\Phi}^{dc} \quad = \quad & \Phi_{B_\Phi}^{tr} \cup \{:\sim p^p \ ; \ :\sim q^q \ ; \ :\sim r^r \ ; \\ & i^p :\text{-}\ i \ ; \ i^q :\text{-}\ i \ ; \ i^r :\text{-}\ i \\ & :\sim i^p \ ; \ :\sim i^q \ ; \ :\sim i^r \ ; \ :\sim i\}.\end{aligned}$$

We obtain that $AS(\Phi_{B_\Phi}^{dc}) = \{A_3, A_4\}$, where as in Example 3.11

$$\begin{aligned} A_3 &= \{c, q^p, r^p, p^q, r^q, p^r, r^r\}; \\ A_4 &= \{c, q^p, r^p, p^q, r^q, q^r, r^r\}, \end{aligned}$$

as these two answer sets are the only ones that violate only one weak constraint, namely $:\sim r^r$. We observe that $\{a \mid a^a \in A_3\} = \{a \mid a^a \in A_4\} = \{r\} = DC(\Phi)$.

Obviously, one can compute all brave, cautious, or definite consequences of a program by choosing $S = B_P$. We also note that the programs from Definitions 3.6, 3.9 and 3.12 yield multiple answer sets. However each of these yields the same atoms a^a, so it is sufficient to compute one of these. The programs could be extended in order to admit only one answer set by suitably penalizing all atoms a^b ($a \neq b$). To avoid interference with the weak constraints already used, these additional weak constraints would have to pertain to a different level.

4 Non-Ground Manifold Programs

We will now generalize the techniques introduced in Section 3 to non-ground strong programs. In principle, one could annotate each predicate (rather than atom as in Section 3) with ground atoms of a subset of the Herbrand Base. However, one can also move the annotations to the non-ground level: For example, instead of annotating a rule $p(X,Y) :\text{-}\ q(X,Y)$ by a set of ground atoms $\{r(a), r(b)\}$ to yield $p^{r(a)}(X,Y) :\text{-}\ q^{r(a)}(X,Y)$ and $p^{r(b)}(X,Y) :\text{-}\ q^{r(b)}(X,Y)$ we will annotate using only the predicate r and extend the arguments of p, yielding the compact rule $\mathrm{d}_p^r(X,Y,Z) :\text{-}\ \mathrm{d}_q^r(X,Y,Z)$ (we use predicate symbols d_p^r and d_q^r rather than p^r and q^r just for pointing out the difference between annotation by predicates versus annotation by ground atoms). In this particular example we have assumed that the program is to be annotated by all ground instances of $r(Z)$; we will use this assumption also in the following for simplifying the presentation. In practice, one can clearly add atoms to the rule body for restricting the instances of the predicate by which we annotate, in the example this would yield $p^r(X,Y,Z) :\text{-}\ q^r(X,Y,Z), dom(Z)$ where the predicate dom should be defined appropriately. In the following definition, recall that $\alpha(p)$ denotes the arity of a predicate p.

Definition 4.1 *Given an atom $a = p(t_1, \dots, t_n)$ and a predicate q, let a_q^{tr} be $\mathrm{d}_p^q(t_1, \dots, t_n, X_1, \dots, X_{\alpha(q)})$ where $X_1, \dots, X_{\alpha(q)}$ are fresh variables and d_p^q is a new predicate symbol with $\alpha(\mathrm{d}_p^q) = \alpha(p) + \alpha(q)$.*

Furthermore, given a set $\mathcal{L}$ of literals, and a predicate q, let $\mathcal{L}_q^{tr}$ be $\{a_q^{tr} \mid$ atom $a \in \mathcal{L}\} \cup \{\text{not } a_q^{tr} \mid \text{not } a \in \mathcal{L}\}$.

Note that we assume that even though the variables $X_1, \dots, X_{\alpha(q)}$ are fresh, they will be the same for each a_q^{tr}. One could define similar notions also for partially ground atoms or for sets of atoms characterized by a collection of defining rules, from which we refrain here for the ease of presentation. We define the manifold program in analogy to Definition 3.2, the only difference being the different way of annotating.

Definition 4.2 *Given a strong program P and a set S of predicates, define its* manifold *as*

$$P_S^{tr} = \bigcup_{r \in P} r_S^{tr} \cup \{c :\text{-}\ \text{not } i \ ; \ i :\text{-}\ \text{not } c\}$$

where

$$r_S^{tr} = \{H(r)_q^{tr} :\text{-}\ \{c\} \cup B(r)_q^{tr} \mid q \in S\}$$

and w.l.o.g. $B_P \cap B_{P_S^{tr}} = \emptyset$.

Example 4.3 *Consider*

$$\Psi \;=\; \{p(X) \vee q(X) \text{ :- } r(X) \;\; ; \;\; r(a) \text{ :- } \;\; ; \;\; r(b) \text{ :- } \}$$

for which

$$\begin{aligned} AS(\Psi) \;=\; \{&\{p(a), p(b), r(a), r(b)\}, \{p(a), q(b), r(a), r(b)\}, \\ &\{q(a), p(b), r(a), r(b)\}, \{q(a), q(b), r(a), r(b)\}\} \end{aligned}$$

$BC(\Psi) = \{p(a), p(b), q(a), q(b), r(a), r(b)\}$ and $CC(\Psi) = DC(\Psi) = \{r(a), r(b)\}$. When forming the manifold for $S = \{p\}$, we obtain

$$\Psi^{tr}_S = \left\{ \begin{array}{l} \mathrm{d}^p_p(X, X_1) \vee \mathrm{d}^p_q(X, X_1) \text{ :- } \mathrm{d}^p_r(X, X_1) \\ \mathrm{d}^p_r(a, X_1) \text{ :- } \;\; ; \;\; \mathrm{d}^p_r(b, X_1) \text{ :- } \\ c \text{ :- not } i \;\; ; \;\; i \text{ :- not } c \end{array} \right\}$$

$AS(\Psi^{tr}_S)$ consists of $\{i\}$ plus 16 answer sets, corresponding to all combinations of the 4 answer sets in $AS(\Psi)$.

Now we can generalize the encodings for brave, cautious, and definite consequences. These definitions are direct extensions of Definitions 3.6, 3.9, and 3.12, the differences are only due to the non-ground annotations. In particular, the diagonalization atoms a^a should now be written as $\mathrm{d}^p_p(X_1, \ldots, X_{\alpha(p)}, X_1, \ldots, X_{\alpha(p)})$ which represent the set of ground instances of $p(X_1, \ldots, X_{\alpha(p)})$, each annotated by itself. So a weak constraint $:\sim \; \mathrm{d}^p_p(X_1, \ldots, X_{\alpha(p)}, X_1, \ldots, X_{\alpha(p)})$ gives rise to $\{:\sim \; \mathrm{d}^p_p(c_1, \ldots, c_{\alpha(p)}, c_1, \ldots, c_{\alpha(p)}) \mid c_1, \ldots, c_{\alpha(p)} \in U\}$ where U is the Herbrand base of the program in question, that is one weak constraint for each ground instance annotated by itself.

Definition 4.4 *Given a strong program P and a set S of predicate symbols, let*

$$\begin{aligned} P^{bc}_S &= P^{tr}_S \cup \{:\sim \text{not } \Delta_q \mid q \in S\} \cup \{:\sim i\} \\ P^{cc}_S &= P^{tr}_S \cup \{:\sim \Delta_q;\; \Delta_q \text{ :- } i \mid q \in S\} \cup \{:\sim i\} \\ P^{dc}_S &= P^{tr}_S \cup \{:\sim \Delta_q;\; I_q \text{ :- } i;\; :\sim I_q \mid q \in S\} \cup \{:\sim i\} \end{aligned}$$

where $\Delta_q = \mathrm{d}^q_q(X_1, \ldots, X_{\alpha(q)}, X_1, \ldots, X_{\alpha(q)})$ and $I_q = i_q(X_1, \ldots, X_{\alpha(q)})$.

Proposition 4.5 *Given a strong program P and a set S of predicates, for an arbitrary $A \in AS(P^{bc}_S)$, (resp., $A \in AS(P^{cc}_S)$, $A \in AS(P^{dc}_S)$), the set*

$$\{p(c_1, \ldots, c_{\alpha(p)}) \mid \mathrm{d}^p_p(c_1, \ldots, c_{\alpha(p)}, c_1, \ldots, c_{\alpha(p)}) \in A\}$$

is the set of brave (resp., cautious, definite) consequences of P with a predicate in S.

Example 4.6 *Consider again Ψ and $S = \{p\}$ from Example 4.3. We obtain*

$$\Psi^{bc}_S \;=\; \Psi^{tr}_S \cup \{:\sim \text{not } \mathrm{d}^p_p(X_1, X_1) \;\; ; \;\; :\sim i\}$$

and we can check that

$$\begin{aligned} AS(\Psi^{bc}_S) \;=\; \{&R \cup \{\mathrm{d}^p_p(a,a), \mathrm{d}^p_p(b,b), \mathrm{d}^p_q(a,b), \mathrm{d}^p_q(b,a)\}, \\ &R \cup \{\mathrm{d}^p_p(a,a), \mathrm{d}^p_p(b,b), \mathrm{d}^p_p(a,b), \mathrm{d}^p_q(b,a)\}, \\ &R \cup \{\mathrm{d}^p_p(a,a), \mathrm{d}^p_p(b,b), \mathrm{d}^p_q(a,b), \mathrm{d}^p_p(b,a)\}, \\ &R \cup \{\mathrm{d}^p_p(a,a), \mathrm{d}^p_p(b,b), \mathrm{d}^p_p(b,a), \mathrm{d}^p_p(b,a)\}\} \end{aligned}$$

where $R = \{\mathrm{d}^p_r(a,a), \mathrm{d}^p_r(a,b), \mathrm{d}^p_r(b,a), \mathrm{d}^p_r(b,b)\}$. For each A of these answer sets we obtain

$$\{p(t) \mid \mathrm{d}^p_p(t,t) \in A\} = \{p(a), p(b)\}$$

which corresponds exactly to the brave consequences of Ψ with a predicate of $S = \{p\}$.

For cautious consequences,

$$\begin{aligned} \Psi^{cc}_S \;=\; \Psi^{tr}_S \cup \{&:\sim \mathrm{d}^p_p(X_1, X_1) \;\; ; \\ &\mathrm{d}^p_p(X_1, X_1) \text{ :- } i \;\; ; \;\; :\sim i\} \end{aligned}$$

and we can check that

$$\begin{aligned} AS(\Psi^{cc}_S) \;=\; \{&R \cup \{\mathrm{d}^p_q(a,a), \mathrm{d}^p_q(b,b), \mathrm{d}^p_q(a,b), \mathrm{d}^p_q(b,a)\}, \\ &R \cup \{\mathrm{d}^p_q(a,a), \mathrm{d}^p_q(b,b), \mathrm{d}^p_p(a,b), \mathrm{d}^p_q(b,a)\}, \\ &R \cup \{\mathrm{d}^p_q(a,a), \mathrm{d}^p_q(b,b), \mathrm{d}^p_q(a,b), \mathrm{d}^p_p(b,a)\}, \\ &R \cup \{\mathrm{d}^p_q(a,a), \mathrm{d}^p_q(b,b), \mathrm{d}^p_p(b,a), \mathrm{d}^p_p(b,a)\}\} \end{aligned}$$

where $R = \{\mathrm{d}^p_r(a,a), \mathrm{d}^p_r(a,b), \mathrm{d}^p_r(b,a), \mathrm{d}^p_r(b,b)\}$. For each A of these answer sets we obtain

$$\{p(t) \mid \mathrm{d}^p_p(t,t) \in A\} = \emptyset$$

and indeed there are no cautious consequences of Ψ with a predicate of $S = \{p\}$.

Finally, for definite consequences,

$$\begin{aligned} \Psi^{dc}_S \;=\; \Psi^{tr}_S \cup \{&:\sim \mathrm{d}^p_p(X_1, X_1) \;\; ; \;\; i_p(X_1) \text{ :- } i \;\; ; \\ &:\sim i_p(X_1) \;\; ; \;\; :\sim i\} \end{aligned}$$

It is easy to see that $AS(\Psi^{dc}_S) = AS(\Psi^{cc}_S)$ and so

$$\{p(t) \mid \mathrm{d}^p_p(t,t) \in A\} = \emptyset$$

for each answer set A of Ψ^{dc}_S, and indeed there is also no definite consequence of Ψ with a predicate of $S = \{p\}$.

These definitions exploit the fact that the semantics of non-ground programs is defined via their grounding with respect to their Herbrand Universe. So the fresh variables introduced in the manifold will give rise to one copy of a rule for each ground atom in question.

In practice, ASP systems usually require rules to be safe, that is, that each variable occurs (also) in the positive body. The manifold for a set of predicates may therefore contain unsafe rules (because of the fresh variables). But this can be repaired by adding a *domain atom* $dom_q(X_1, \ldots, X_m)$ to a rule which is to be annotated with q. This predicate can in turn be defined by a rule $dom_q(X_1, \ldots, X_m) \text{ :- } u(X_1), \ldots, u(X_m)$ where u is defined using $\{u(c) \mid c \in U_P\}$. One can also provide smarter definitions for dom_q by using a relaxation of the definition for q.

We also observe that ground atoms that are contained in all answer sets of a program need not be annotated in the manifold. Note that these are essentially the cautious consequences of a program and therefore determining all of those before rewriting does not make sense. But for some atoms this property can be determined only by the structure of the program. For instance, facts will be in all answer sets. In the sequel we will not annotate extensional atoms (those defined only by facts) in order to obtain more concise programs. One could also go further and omit the annotation of atoms which are defined using stratified programs.

As an example, we present an ASP encoding for boolean satisfiability and then create its manifold program for resolving the following problem: Given a propositional formula in CNF φ, compute all atoms which are true in all models of φ. We provide a fixed program which takes a representation of φ as facts as input. To apply our method we first require a program whose answer sets are in a one-to-one correspondence to the models of φ. To start with, we fix the representation of CNFs. Let φ (over atoms A) be of the form $\bigwedge_{i=1}^{n} c_i$. Then,

$$\begin{aligned} D_\varphi \;=\;& \{\mathrm{at}(a) \mid a \in A\} \cup \{\mathrm{cl}(i) \mid 1 \leq i \leq n\} \cup \\ & \{\mathrm{pos}(a,i) \mid \text{atom } a \text{ occurs positively in } c_i\} \cup \\ & \{\mathrm{neg}(a,i) \mid \text{atom } a \text{ occurs negatively in } c_i\}. \end{aligned}$$

We construct program SAT as set of the following rules

$$\begin{aligned} \mathit{true}(X) &\text{ :- } \mathrm{not}\ \mathit{false}(X), \mathrm{at}(X) \\ \mathit{false}(X) &\text{ :- } \mathrm{not}\ \mathit{true}(X), \mathrm{at}(X) \\ \mathit{ok}(C) &\text{ :- } \mathit{true}(X), \mathrm{pos}(C,X) \\ \mathit{ok}(C) &\text{ :- } \mathit{false}(X), \mathrm{neg}(C,X) \\ &\text{ :- } \mathrm{not}\ \mathit{ok}(C), \mathrm{cl}(C). \end{aligned}$$

It can be checked that the answer sets of $\mathrm{SAT} \cup D_\varphi$ are in a one-to-one correspondence to the models (over A) of φ. In particular, for any model $I \subseteq A$ of φ there exists an answer set M of $\mathrm{SAT} \cup D_\varphi$ such that $I = \{a \mid \mathit{true}(a) \in M\}$. We now consider $\mathrm{SAT}^{cc}_{\{true\}}$ which consists of the following rules.

$$\begin{aligned} c &\text{ :- } \mathrm{not}\ i \\ i &\text{ :- } \mathrm{not}\ c \\ \mathrm{d}^{true}_{true}(X,Y) &\text{ :- } c, \mathrm{not}\ \mathrm{d}^{true}_{false}(X,Y), \mathrm{at}(X) \\ \mathrm{d}^{true}_{false}(X,Y) &\text{ :- } c, \mathrm{not}\ \mathrm{d}^{true}_{true}(X,Y), \mathrm{at}(X) \\ \mathrm{d}^{true}_{ok}(C,Y) &\text{ :- } c, \mathrm{d}^{true}_{true}(X,Y), \mathrm{pos}(C,X) \\ \mathrm{d}^{true}_{ok}(C,Y) &\text{ :- } c, \mathrm{d}^{true}_{false}(X,Y), \mathrm{neg}(C,X) \\ &\text{ :- } c, \mathrm{not}\ \mathrm{d}^{true}_{ok}(C,Y), \mathrm{cl}(C) \\ &\text{ :}\sim \mathrm{d}^{true}_{true}(X,X) \\ \mathrm{d}^{true}_{true}(X,X) &\text{ :- } i \\ &\text{ :}\sim i \end{aligned}$$

Given Proposition 4.5, it is easy to see that, given some answer set A of $\mathrm{SAT}^{cc}_{\{true\}} \cup D_\varphi$, $\{a \mid \mathrm{d}^{true}_{true}(a,a) \in A\}$ is precisely the set of atoms which are true in all models of φ.

5 Applications

In this section, we put our technique to work and show how to use meta-reasoning over answer sets for two application scenarios. The first one is a well-known problem from propositional logic, and we will reuse the example from above. The second example takes a bit more background, but presents a novel method to compute ideal extensions for argumentation frameworks.

5.1 The Unique Minimal Model Problem

As a first example, we show how to encode the problem of deciding whether a given propositional formula φ has a unique minimal model. This problem is known to be in Θ_2^P and to be co-NP-hard (the exact complexity is an open problem). The following relation is quite obvious. Let I be the intersection of all models of φ. Then φ has a unique minimal model iff I is also a model of φ. We thus use our example from the previous section, and define UNIQUE as $\mathrm{SAT}^{cc}_{\{true\}}$ augmented by the following rules:

$$\begin{aligned} \mathit{ok}(C) &\text{ :- } \mathrm{d}^{true}_{true}(X,X), \mathrm{pos}(C,X) \\ \mathit{ok}(C) &\text{ :- } \mathrm{not}\ \mathrm{d}^{true}_{true}(X,X), \mathrm{neg}(C,X) \\ &\text{ :- } \mathrm{not}\ \mathit{ok}(C), \mathrm{cl}(C) \end{aligned}$$

We immediately obtain the following result

Theorem 5.1 *For any CNF formula φ, it holds that φ has a unique minimal model, if and only if program* $\mathrm{UNIQUE} \cup D_\varphi$ *has at least one answer set.*

A slight adaption of this encoding allows us to formalize CWA-reasoning [Reiter, 1978] over a propositional knowledge base φ, since the atoms a in φ, for which the corresponding atoms $\mathrm{d}^{true}_{true}(a,a)$ are not contained in an answer set of $\mathrm{SAT}^{cc}_{\{true\}} \cup D_\varphi$, are exactly those which are added negated to φ for CWA-reasoning.

5.2 Computing the Ideal Extension

Our second example is from the area of argumentation, where the problem of computing the ideal extension [Dung *et al.*, 2007] of an abstract argumentation framework was recently shown to be complete for $\mathrm{FP}^{\mathrm{NP}}_{||}$ by Dunne [2008]. Thus, this task cannot be compactly encoded via normal programs (under usual complexity theoretic assumptions). On the other hand, the complexity shows that employing disjunction is not necessary, if one instead uses weak constraints.

We first give the basic definitions for argumentation frameworks following Dung [1995].

Definition 5.2 *An* argumentation framework (AF) *is a pair $F = (A, R)$ where $A \subseteq \mathcal{U}$ is a set of arguments and $R \subseteq A \times A$. $(a, b) \in R$ means that a attacks b. An argument $a \in A$ is* defended *by $S \subseteq A$ (in F) if, for each $b \in A$ such that $(b, a) \in R$, there exists a $c \in S$, such that $(c, b) \in R$. An argument a is admissible (in F) w.r.t. a set $S \subseteq A$ if each $b \in A$ which attacks a is defended by S.*

Semantics for argumentation frameworks are given in terms of so-called extensions. The next definitions introduces two such notions which also underly the concept of an ideal extension.

Definition 5.3 *Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is said to be* conflict-free (in F)*, if there are no $a, b \in S$, such that $(a, b) \in R$. A set S is an* admissible extension *of F, if S is conflict-free in F and each $a \in S$ is admissible in F w.r.t. S. The collection of admissible extensions is denoted by $adm(F)$. An admissible extension S of F is a* preferred extension *of F, if for each $T \in adm(F)$, $S \not\subset T$. The collection of preferred extensions of F is denoted by $pref(F)$.*

The original definition of ideal extensions is as follows [Dung *et al.*, 2007].

Definition 5.4 *Let F be an AF. A set S is called* ideal *for F, if $S \in adm(F)$ and $S \subseteq \bigcap_{T \in pref(F)} T$. A maximal (w.r.t. set-inclusion) ideal set of F is called an* ideal extension *of F.*

It was shown that for each AF F, a unique ideal extension exists. Dunne [2008] gave the following algorithm to compute the ideal extension of an AF $F = (A, R)$. Let $X_F^- = A \setminus \bigcup_{S \in adm(F)} S$ and $X_F^+ = \{a \in A \mid \forall b, c : (b,a), (a,c) \in R \Rightarrow b, c \in X_F^-\} \setminus X_F^-$, and define an AF

$$F^* = (X_F^+ \cup X_F^-, R^*), \text{ where}$$
$$R^* = R \cap \{(a,b), (b,a) \mid a \in X_F^+, b \in X_F^-\}.$$

F^* is a bipartite AF in the sense that R^* is a bipartite graph.

Proposition 5.5 *The ideal extension of an AF F is given by $\bigcup_{S \in adm(F^*)} (S \cap X_F^+)$.*

The set of all admissible atoms for a bipartite AF F can be computed in polynomial time using Algorithm 1 of [Dunne, 2007]. This is basically a fixpoint iteration identifying arguments in X_F^+ that cannot be in an admissible extension: First, arguments in $X_0 = X_F^+$ are excluded, which are attacked by unattacked arguments (which are necessarily in X_F^-), yielding X_1. Now, arguments in X_F^- may be unattacked by X_1, and all arguments in X_1 attacked by such newly unattacked arguments should be excluded. This process is iterated until either no arguments are left or no more argument can be excluded. There may be at most $|X_F^+|$ iterations in this process.

We exploit this technique to formulate an ASP-encoding IDEAL. We first report a program the answer sets of which characterize admissible extensions. Then, we use the brave manifold of this program in order to determine all arguments contained in some admissible extension. Finally, we extend this manifold program in order to identify F^* and to simulate Algorithm 1 of [Dunne, 2007].

The argumentation frameworks will be given to IDEAL as sets of input facts. Given an AF $F = (A, R)$, let

$$D_F = \{a(x) \mid x \in A\} \cup \{\mathrm{r}(x,y) \mid (x,y) \in R\}.$$

Program ADM, given by the rules below, computes admissible extensions (cf. [Osorio *et al.*, 2005; Egly *et al.*, 2008]):

$$\begin{aligned}
\mathrm{in}(X) &\text{ :- } \mathrm{not\ out}(X), a(X)\\
\mathrm{out}(X) &\text{ :- } \mathrm{not\ in}(X), a(X)\\
&\text{ :- } \mathrm{in}(X), \mathrm{in}(Y), \mathrm{r}(X,Y)\\
\mathrm{def}(X) &\text{ :- } \mathrm{in}(Y), \mathrm{r}(Y,X)\\
&\text{ :- } \mathrm{in}(X), \mathrm{r}(Y,X), \mathrm{not\ def}(Y)
\end{aligned}$$

Indeed one can show that, given an AF F the answer sets of $\mathrm{ADM} \cup D_F$ are in a one-to-one correspondence to the admissible extensions of F via the $\mathrm{in}(\cdot)$ predicate. In order to determine the brave consequences of ADM for predicate in, we form $\mathrm{ADM}^{bc}_{\{\mathrm{in}\}}$, and extend it by collecting all brave consequences of $\mathrm{ADM} \cup D_F$ (for a given AF $F = (A, R)$) in predicate $\mathrm{in}(\cdot)$, from which we can determine X_F^- (represented by $\mathrm{in}^-(\cdot)$), X_F^+ (represented by $\mathrm{in}^+(\cdot)$, using auxiliary predicate $\mathrm{not\_in}^+(\cdot)$), and R^* (represented by $\mathrm{q}(\cdot,\cdot)$).

$$\begin{aligned}
\mathrm{in}(X) &\text{ :- } \mathrm{d}^{\mathrm{in}}_{\mathrm{in}}(X,X)\\
\mathrm{in}^-(X) &\text{ :- } a(X), \mathrm{not\ in}(X)\\
\mathrm{not\_in}^+(X) &\text{ :- } \mathrm{in}(Y), r(X,Y)\\
\mathrm{not\_in}^+(X) &\text{ :- } \mathrm{in}(Y), r(Y,X)\\
\mathrm{in}^+(X) &\text{ :- } \mathrm{in}(X), \mathrm{not\ not\_in}^+(X)\\
\mathrm{q}(X,Y) &\text{ :- } \mathrm{r}(X,Y), \mathrm{in}^+(X), \mathrm{in}^-(Y)\\
\mathrm{q}(X,Y) &\text{ :- } \mathrm{r}(X,Y), \mathrm{in}^-(X), \mathrm{in}^+(Y)
\end{aligned}$$

In order to simulate Algorithm 1 of [Dunne, 2007], we use the elements in X_F^+ for marking the iteration steps. To this end, we use an arbitrary order $<$ on ASP constants (all ASP systems provide such a predefined order) and define successor, infimum and supremum among the constants representing X_F^+ w.r.t. the order $<$.

$$\begin{aligned}
\mathrm{nsucc}(X,Z) &\text{ :- } \mathrm{in}^+(X), \mathrm{in}^+(Y), \mathrm{in}^+(Z), X{<}Y, Y{<}Z\\
\mathrm{succ}(X,Y) &\text{ :- } \mathrm{in}^+(X), \mathrm{in}^+(Y), X{<}Y, \mathrm{not\ nsucc}(X,Y)\\
\mathrm{ninf}(Y) &\text{ :- } \mathrm{in}^+(X), \mathrm{in}^+(Y), X{<}Y\\
\mathrm{inf}(X) &\text{ :- } \mathrm{in}^+(X), \mathrm{not\ ninf}(X)\\
\mathrm{nsup}(X) &\text{ :- } \mathrm{in}^+(X), \mathrm{in}^+(Y), X{<}Y\\
\mathrm{sup}(X) &\text{ :- } \mathrm{in}^+(X), \mathrm{not\ nsup}(X)
\end{aligned}$$

We now use this to iteratively determine arguments that are not in the ideal extension, using $\mathrm{nideal}(\cdot,\cdot)$, where the first argument is the iteration step. In the first iteration (identified by the infimum) all arguments in X_F^+ which are attacked by an unattacked argument are collected. In subsequent iterations, all arguments from the previous steps are included and augmented by arguments that are attacked by an argument not attacked by arguments in X_F^+ that were not yet excluded in the previous iteration. Finally, arguments in the ideal extension are those that are not excluded from X_F^+ in the final iteration (identified by the supremum).

$$\begin{aligned}
\mathrm{att}_0(X) &\text{ :- } \mathrm{q}(Y,X)\\
\mathrm{nideal}(I,Y) &\text{ :- } \mathrm{inf}(I), \mathrm{q}(Z,Y), \mathrm{in}^+(Y), \mathrm{not\ att}_0(Z)\\
\mathrm{nideal}(I,Y) &\text{ :- } \mathrm{succ}(J,I), \mathrm{nideal}(J,Y)\\
\mathrm{nideal}(I,Y) &\text{ :- } \mathrm{succ}(J,I), q(Z,Y), \mathrm{in}^+(Y), \mathrm{not\ att}_i(J,Z)\\
\mathrm{att}_i(J,Z) &\text{ :- } q(Y,Z), \mathrm{in}^+(Y), \mathrm{not\ nideal}(J,Y), \mathrm{in}^+(J)\\
\mathrm{ideal}(X) &\text{ :- } \mathrm{in}^+(X), \mathrm{sup}(I), \mathrm{not\ nideal}(I,X)
\end{aligned}$$

If we put $\mathrm{ADM}^{bc}_{\{\mathrm{in}\}}$ and all of these additional rules together to form the program IDEAL, we obtain the following result:

Theorem 5.6 *Let F be an AF and $A \in AS(\mathrm{IDEAL} \cup D_F)$. Then, the ideal extension of F is given by $\{a \mid \mathrm{ideal}(a) \in A\}$.*

6 Conclusion

In this paper, we provided a novel method to rewrite ASP-programs in such a way that reasoning over all answer sets of the original program can be formulated within the same program. Our method exploits the well-known concept of weak constraints. We illustrated the impact of our method by encoding the problems of (i) deciding whether a propositional formula in CNF has a unique minimal model, and (ii) computing the ideal extension of an argumentation framework. Known complexity results witness that our encodings are adequate in the sense that efficient ASP encodings without weak constraints or similar constructs are assumed to be infeasible.

The manifold program for cautious consequences is also closely related to the concept of data disjunctions [Eiter and Veith, 2002] (this paper also contains a detailed discussion about the complexity class Θ_2^P and related classes for functional problems). Related work has also been done in the area of default logic, where Delgrande and Schaub [2002] proposed a method for reasoning within a single extension. Their method uses set-variables which characterize the set of generating defaults of the original extensions. Thus, their approach differs considerably from ours as it encodes certain aspects of the semantics (which ours does not), which puts it closer to meta-programming (cf. [Eiter *et al.*, 2003]).

Future work includes studying how alternative preferential constructs can be used in place of weak constraints for obtaining manifold programs. Another issue is to generalize the presented methodology in such a way that different manifold programs are combined within a single program. Note that this would amount to an ASP programming language which allows to call several oracles (which compute brave/skeptical/definite consequences of different modules) either in parallel or one after each other. We believe that (besides a more careful renaming technique) such a generalization is, in principle, rather straightforward. The main issue is to devise a suitable extension of ASP syntax for denoting the interfaces among different manifold programs in an intuitive and succinct manner. A frontend would then compile such a specification into a standard program with weak constraints. Using such an advanced framework, we would like to employ manifold programs for encoding various further problems in complexity classes Θ_2^P, Θ_3^P, $\mathrm{FP}_{||}^{\mathrm{NP}}$, and $\mathrm{FP}_{||}^{\Sigma_2^P}$.

References

[Baral, 2002] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving.* Cambridge University Press, 2002.

[Bench-Capon and Dunne, 2007] T. J. M. Bench-Capon and P. E. Dunne. Argumentation in artificial intelligence. *Artif. Intell.*, 171(10-15):619–641, 2007.

[Bravo and Bertossi, 2003] L. Bravo and L. E. Bertossi. Logic programs for consistently querying data integration systems. In *Proc. IJCAI'03*, pages 10–15. Morgan Kaufmann, 2003.

[Buccafurri *et al.*, 2000] F. Buccafurri, N. Leone, and P. Rullo. Enhancing disjunctive datalog by constraints. *IEEE Trans. Knowl. Data Eng.*, 12(5):845–860, 2000.

[Delgrande and Schaub, 2002] J. P. Delgrande and T. Schaub. Reasoning credulously and skeptically within a single extension. *Journal of Applied Non-Classical Logics*, 12(2):259–285, 2002.

[Dung *et al.*, 2007] P. M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artif. Intell.*, 171(10-15):642–674, 2007.

[Dung, 1995] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

[Dunne, 2007] P. E. Dunne. Computational properties of argument systems satisfying graph-theoretic constraints. *Artif. Intell.*, 171(10-15):701–729, 2007.

[Dunne, 2008] P. E. Dunne. The computational complexity of ideal semantics I: Abstract argumentation frameworks. In *Proc. COMMA'08*, pages 147–158. IOS Press, 2008.

[Egly *et al.*, 2008] U. Egly, S. Gaggl, and S. Woltran. Answer-set programming encodings for argumentation frameworks. In *Proc. ASPOCP'08*, 2008.

[Eiter and Veith, 2002] T. Eiter and H. Veith. On the complexity of data disjunctions. *Theor. Comput. Sci.*, 288(1):101–128, 2002.

[Eiter *et al.*, 2003] T. Eiter, W. Faber, N. Leone, and G. Pfeifer. Computing preferred answer sets by meta-interpretation in answer set programming. *TPLP*, 3(4-5):463–498, 2003.

[Gebser *et al.*, 2007] M. Gebser, L. Liu, G. Namasivayam, A. Neumann, T. Schaub, and M. Truszczyński. The first answer set programming system competition. In *Proc. LPNMR'07*, vol. 4483 of *LNCS*, pages 3–17. Springer, 2007.

[Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.

[Gelfond, 2002] M. Gelfond. Representing knowledge in A-Prolog. In A. Kakas and F. Sadri, editors, *Computational Logic: From Logic Programming into the Future*, vol. 2408 of LNCS/LNAI, pages 413–451. Springer, 2002.

[Leone *et al.*, 2006] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.

[Marek and Truszczyński, 1999] V. W. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm – A 25-Year Perspective*, pages 375–398. Springer, 1999.

[Niemelä, 1999] I. Niemelä. Logic programming with stable model semantics as a constraint programming paradigm. *Ann. Math. Artif. Intell.*, 25(3–4):241–273, 1999.

[Osorio *et al.*, 2005] M. Osorio, C. Zepeda, J. C. Nieves, and U. Cortés. Inferring acceptable arguments with answer set programming. In *Proc. ENC'05*, pages 198–205. IEEE, 2005.

[Reiter, 1978] R. Reiter. On closed world data bases. In *Logic and Databases*, pages 55–76. Plenum Press, 1978.

[Saccà, 1996] D. Saccà. Multiple total stable models are definitely needed to solve unique solution problems. *Inf. Process. Lett.*, 58(5):249–254, 1996.

A Transport Reaction Language: Preliminary Report

Thibaut Henin
IRISA
Campus de Beaulieu
F-35042 Rennes Cedex

Torsten Schaub
Universität Potsdam
August-Bebel-Str. 89
D-14482 Potsdam

Abstract

We present a qualitative approach to represent and reason upon transport reactions in metabolic networks. Our approach is built on action description languages for representing the dynamics of metabolic networks. To begin with, we introduce the transport reaction language $\mathcal{T}$ that is a customized sub-language of action language $\mathcal{C}$. We illustrate the modelling capacities by authentic biological examples. Moreover, we describe the system architecture of our system and detail its current major application.

1 Introduction

Molecular biology has seen a technological revolution with the establishment of high-throughput methods over the last years. This has resulted in a rapid growth of biological knowledge, gathered in web databases such as KEGG[12], Biomodels [14], Reactome [11], or MetaCyc[5]. Although the diverse biological networks are expressed in a computer-readable format, namely the *System Biology Markup Language* (SBML), the encompassing knowledge bases do not provide any general form of reasoning or query-answering.

We address this problem by proposing a reaction description language whose domain descriptions can be generated from networks expressed in SBML. A particular feature of our language is that it takes the location of molecules into account and thus allows for describing the transport of species through compartments. To provide the aforementioned reasoning capacities in a well understood framework, we embed our language into action language $\mathcal{C}$. Apart from clear semantic underpinnings, this approach allows us to benefit from the high performance of modern Answer Set Programming (ASP; [1]) systems via well known mappings of $\mathcal{C}$ into ASP. We illustrate the modelling capacities of our language by authentic biological examples. Moreover, we describe the system architecture of our system and detail its current major application.

2 Background

Action languages use *fluents* to describe the states of a dynamical system and *actions* influence the values of fluents. In $\mathcal{C}$, *static laws* describe properties between fluents that need to be satisfied in every state of the system. *Dynamic laws* describe the effects of actions, that is, how the system evolves when actions are executed.

More formally, we consider *action language* $\mathcal{C}$ [9] over a Boolean *action signature* $\langle B, F, A\rangle$, where B is the set $\{f, t\}$ of truth values, F is a set of *fluent names*, and A is a set of *action names*. In $\mathcal{C}$, an *action description* $D_{\mathcal{C}}$ over a signature $\langle B, F, A\rangle$ consists of *static* and *dynamic laws*:

$$(\mathbf{caused}\ \varphi\ \mathbf{if}\ \psi) \tag{1}$$

$$(\mathbf{caused}\ \varphi\ \mathbf{if}\ \psi\ \mathbf{after}\ \omega) \tag{2}$$

where φ and ψ are propositional combinations of fluent names and ω is a propositional combination of fluent and action names. Every action description $D_{\mathcal{C}}$ induces a unique transition system $\mathcal{T}_{\mathcal{C}}(D_{\mathcal{C}}) = \langle S, V, R\rangle$, where S is a set of *states*, V is a function determining fluents values in state s, and R is a relation containing all possible transitions between states. A *trajectory* $s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$ in a transition system $\langle S, V, R\rangle$ is a sequence of sets of actions $A_i \subseteq A$ and states $s_i \in S$ where $(s_{i-1}, A_i, s_i) \in R$ for $0 \leq i \leq n$. Intuitively, a trajectory represents one possible history (or simply path) within a transition system.

In [9], several syntactic extensions are defined. For instance, the rule $(\omega\ \mathbf{may\ cause}\ \varphi\ \mathbf{if}\ \psi)$ is a shorthand for $(\mathbf{caused}\ \varphi\ \mathbf{if}\ \varphi\ \mathbf{after}\ \psi \wedge \omega)$. Similarly, $(\mathbf{inertial}\ \varphi)$ is a shorthand for $(\mathbf{caused}\ \varphi\ \mathbf{if}\ \varphi\ \mathbf{after}\ \varphi)$. We refer to [9] for more detailed definitions.

$\mathcal{C}$ has an associated *query language*, $\mathcal{Q}$ [9], defined in terms of *axioms* and *queries*. The semantics of a query language is defined in terms of trajectories. The language $\mathcal{Q}$ defines two types of propositions.

- A proposition of form (A **occurs at** t_i) is satisfied by a trajectory $s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$, if $A \in A_{i+1}$, where A is an elementary action name and $i < n$.
- A proposition of form (F **holds at** t_i) is satisfied by a trajectory $s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$, if $s_i \models F$, where F a fluent name and $i \leq n$.

An axiom is a proposition possibly preceded by $\neg$. A query is a propositional combination of propositions. A query Q is a consequence of a set Φ of axioms, written $\Phi \models_{\mathcal{T}} Q$, if every trajectory of a transition system $\mathcal{T}$ satisfying all axioms in Φ also satisfies Q.

3 Language $\mathcal{T}$

A *reaction* is a process transforming some species, called *reactants* of the reaction, into some other species called *products* of the reaction. In our setting, a *species* is a molecule associated with a compartment, indicating the presence of the molecule in the compartment. Often reactions rely on species, called *enzymes* of the reaction whose presence is mandatory although they are not subject to any transformations. When a reaction deals with species in different compartments, the reaction is called a *transport*.

Language $\mathcal{T}$. We begin with three disjoints nonempty sets of symbols, viz. *molecule names* M, *compartment names* C, and *reaction names* R.

Our language $\mathcal{T}$ for specifying biological *transport networks* consists of the following expressions:

$$m \textbf{ in } c \tag{3}$$

$$r \textbf{ consumes } s_1, \ldots, s_n \tag{4}$$

$$r \textbf{ produces } s_1, \ldots, s_n \tag{5}$$

$$r \textbf{ needs } s_1, \ldots, s_n \tag{6}$$

$$r_1 \textbf{ overtakes } r_2 \tag{7}$$

A *species* is written as in (3) where $m \in M$ is a molecule and $c \in C$ is a compartment name. In the remainder, $r \in R$ is a reaction name and s_i are species for $1 \leq i \leq n$.

A *consume proposition* is an expression of form (4). This means that reaction r needs the presence of reactants $s_1, \ldots, s_n$ and consumes them during its process.

A *produce proposition* is an expression of form (5), meaning that reaction r produces the species $s_1, \ldots, s_n$.

A *need proposition* is an expression of form (6). This means that reaction r requires the presence of enzymes $s_1, \ldots, s_n$ without consuming them.

An *overtake proposition* is an expression of form (7), meaning that reaction r_1 takes priority over reaction r_2.

Example 1 (Detoxification) *Consider a simplistic metabolic pathway dealing with a cell facing hydrogen peroxide (H_2O_2) [16]. The oxidizing capacity of hydrogen peroxide is so strong that the chemical is considered a highly reactive oxygen species, corroding many materials, including human skin as well as DNA, RNA, and proteins.*

We distinguish two compartments, cytosol *and* outside*, and four molecules, viz. hydrogen peroxide H_2O_2 along with the product of detoxification, viz. H_2O,* catalase[1] *(written CAT), and some critical resource (abstracted as the generic molecule $R - OOH$). The presence of H_2O_2 implies some response of the cell, depending on its state:*

1. *If catalase has been secreted and becomes available outside the cell, H_2O_2 is transformed in H_2O extracellularly.*
2. *If not, H_2O_2 enters the cell and if catalase is present, H_2O_2 is transformed in H_2O in the cytosol.*
3. *If H_2O_2 enters the cell and if catalase is not present, then H_2O_2 reacts with $R - OOH$, which is then degraded into $R - OH$, and the cell dies.*

This network contains two transport reactions. One to enter the H_2O_2 in the cytosol (r_1) and one to export the enzyme (r_2). In $\mathcal{T}$, those two reactions are written as follows:

$$\begin{aligned} r_1 &\textbf{ consumes } H_2O_2 \textbf{ in } \mathit{outside} \\ r_1 &\textbf{ produces } H_2O_2 \textbf{ in } \mathit{cytosol} \\ r_2 &\textbf{ consumes } CAT \textbf{ in } \mathit{cytosol} \\ r_2 &\textbf{ produces } CAT \textbf{ in } \mathit{outside} \end{aligned}$$

Our network contains also some reactions to transform H_2O_2. Catalase *is involved in two reactions transforming H_2O_2; one outside the cell (r_3) and another in the cytosol (r_4). Finally, there is a reaction transforming hydrogen peroxide into its inactive variant within the cytosol, but consuming the critical resource (r_5).*

$$\begin{aligned} r_3 &\textbf{ consumes } H_2O_2 \textbf{ in } \mathit{outside} \\ r_3 &\textbf{ produces } H_2O \textbf{ in } \mathit{outside} \\ r_3 &\textbf{ needs } CAT \textbf{ in } \mathit{outside} \\ r_4 &\textbf{ consumes } H_2O_2 \textbf{ in } \mathit{cytosol} \\ r_4 &\textbf{ produces } H_2O \textbf{ in } \mathit{cytosol} \\ r_4 &\textbf{ needs } CAT \textbf{ in } \mathit{cytosol} \\ r_5 &\textbf{ consumes } H_2O_2 \textbf{ in } \mathit{cytosol},\ R - OH \textbf{ in } \mathit{cytosol} \\ r_5 &\textbf{ produces } R - OOH \textbf{ in } \mathit{cytosol} \end{aligned}$$

Furthermore, this pathway contains one priority relation between r_4 and r_5. Indeed, whenever catalase is present in the cell, it transforms H_2O_2 into water before H_2O_2 can react with $R - OOH$. Usually, this last reaction never occurs because of the presence of catalase. However, in some cases, catalase is not fast enough and the cell dies (generally the concentration of H_2O_2 becomes too high).

In $\mathcal{T}$, this relation is expressed as follows:

$$r_4 \textbf{ overtakes } r_5$$

Example 2 (Glycolysis) *Glycolysis is a metabolic pathway converting glucose into pyruvate, while generating high energy compounds (ATP and NADH). In the early stage, glucose is imported inside the cytosol. Then, a series of reactions transform glucose into pyruvate molecules. Pyruvate is then transformed into ethanol which can be transported outside the cell. In the middle of the pathway, an intermediary compound (dihydroxyacetone-phosphate) can be used and lead to the formation of glycerol. This is detailed in Figure 1.*

As the complete glycolysis pathway contains too many reactions, we focus on two reactions that allow for illustrating two kind of reactions. The first reaction of this pathway is a transport reaction which takes some glucose outside the cell and imports it inside the cell. In $\mathcal{T}$, this reaction is written as follows:

$$\begin{aligned} r_6 &\textbf{ consumes } \mathit{glucose} \textbf{ in } \mathit{outside} \\ r_6 &\textbf{ produces } \mathit{glucose} \textbf{ in } \mathit{cell} \end{aligned}$$

[1]Enzyme found in most plant and animal cells that functions as an oxidative catalyst; decomposes hydrogen peroxide into oxygen and water.

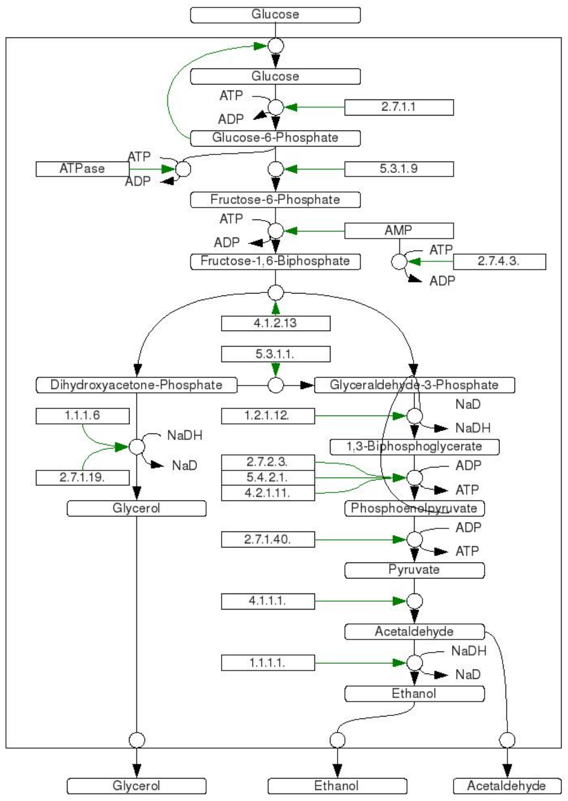


Figure 1: Map of the glycolysis

Another reaction of interest is the transformation of 1,3-Biphosphoglycerate into phosphoenolpyruvate because it needs three enzymes. In $\mathcal{T}$, this reaction is written as follows:

r_7	**consumes**	$1,3-\mathit{Biphosphoglycerate}$ **in** *cell*,
		ADP **in** *cell*
r_7	**produces**	*phosphoenolpyruvate* **in** *cell*,
		ATP **in** *cell*
r_7	**needs**	2.7.2.3 **in** *cell*,
		5.4.2.1 **in** *cell*,
		4.2.1.11 **in** *cell*

Translation into $\mathcal{C}$. The meaning of a transport network described in $\mathcal{T}$ is fixed through a translation into action language $\mathcal{C}$. Apart from providing a formal semantics, this allows us to draw upon the greatly elaborated framework of $\mathcal{C}$, offering query and observation languages, extensions such as additive fluents, needed for expressing resources, and finally an efficient implementation through off-the-shelf $\mathcal{C}$ solvers.

To this end, we map a transport network $N_{\mathcal{T}}$ in $\mathcal{T}$ over signature (M, C, R) into a (definite) action description $\mathbf{T}(N_{\mathcal{T}})$ in $\mathcal{C}$ over a Boolean signature (B, F, A), where $B = \{t, f\}$, $A = R$ and $F = \{\mathit{present}(m, c) \mid (m, c) \in M \times C\} \cup \{\bot\} \cup \{\mathit{possible}(r) \mid r \in R\}$ where $\bot$ is interpreted by f. In what follows, we detail our translation $\mathbf{T}$ by giving the translation of each expression in $\mathcal{T}$ into propositions of $\mathcal{C}$.

For each reactant (m **in** c) occurring in a consume proposition as in (4), we define one dynamic law.

$$\textbf{caused } \neg\mathit{present}(m, c) \textbf{ if } \neg\mathit{present}(m, c) \textbf{ after } r \quad (8)$$

It expresses that the reactant may but must not be consumed.

For each product (m **in** c) in a product proposition of form (5), we define a dynamic law expressing that the species is produced by the reaction.

$$\textbf{caused } \mathit{present}(m, c) \textbf{ after } r \quad (9)$$

For each enzyme (e_i **in** c_i) in a need proposition in (6) of each reaction, plus each reactant (r_j **in** c_j) in a consume proposition as in (4) of the same reaction we define a static law and a dynamic law to express that those species are mandatory for the reaction.

$$\mathit{possible}(r) \textbf{ if } \bigwedge_i \mathit{present}(e_i, c_i) \ \bigwedge_j \mathit{present}(r_j, c_j) \quad (10)$$

$$\textbf{caused } \bot \textbf{ after } r \ \wedge \ \neg\mathit{possible}(r) \quad (11)$$

For each overtakes expression of the form (7), we define a static law expressing that the less prior reaction cannot occur alone when both reaction can occur.

$$\textbf{caused } \bot \textbf{ after } r_2 \ \wedge \ \neg\, r_1 \ \wedge \ \mathit{possible}(r_1) \quad (12)$$

Action query language $\mathcal{Q}_{\mathcal{T}}$ We adapt and extend action query language $\mathcal{Q}$ in order to use it with action language $\mathcal{T}$ by defining a new axiom dealing with species instead of fluents and another one providing confidence levels. As with $\mathcal{Q}$, the semantics of query language $\mathcal{Q}_{\mathcal{T}}$ is given in terms of trajectories.

$\mathcal{Q}_{\mathcal{T}}$ is build from three types of propositions:

- A proposition of form
$$(m \textbf{ in } c \textbf{ is present at } t_i) \quad (13)$$
is satisfied by a trajectory $s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$ if $\mathit{present}(m, c) \in s_i$ where (m **in** c) is a species and $i \leq n$.
- A proposition of form
$$(r \textbf{ occurs at } t_i) \quad (14)$$
is satisfied by a trajectory $s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$ if $r \in A_{i+1}$ where r is a reaction name and $i < n$.
- A proposition of form
$$(r \textbf{ has confidence level } l) \quad (15)$$
where r is a reaction name and l is an integer. Every trajectory satisfies this proposition. If no confidence level is given for a reaction, we give it the highest among all reactions. We denote by $L(r)$ the confidence level attributed to r. The confidence level of a trajectory $T = s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$, written $L(T)$, is defined as the minimal confidence level of the reaction in T. In symbols:
$$L(T) = \min\{L(r) \mid r \in A_i, 1 \leq i \leq n\}$$

An axiom is a proposition of form (13) or (14) possibly preceded by $\neg$. A query is a propositional combination of propositions of form (13) or (14).

Confidence propositions induce an order on trajectories via the confidence, $L(T)$, associated with each trajectory, T, in a given transition system. Given two trajectories T and T', we say T' is *more confident* than T, written $T \leq T'$, if

$$L(T) \leq L(T')$$

A query Q is a *confident consequence* of a set Φ of axioms, written $\Phi \models^c_{\mathcal{T}} Q$, if every $\leq$-maximal trajectory of a transition system $\mathcal{T}$ satisfying all axioms in Φ also satisfies Q.

4 Toolbox

We have implemented our approach as a modular toolbox, comprising the off-the-shelf ASP grounder *gringo*[2] and ASP solver *clasp*[2] as reasoning engines.

The initial input of the system is a set of metabolic pathway given in SBML (or directly in our language $\mathcal{T}$). SBML files are usually downloaded from internet databases, like Biomodels [14] or Reactome [11]. To begin with, a pathway expressed in SBML is translated into our reaction language $\mathcal{T}$ (using the libsbml library [4]). A pathway in $\mathcal{T}$ can then be queried via query language $\mathcal{Q}_{\mathcal{T}}$. To this end, both the query and the pathway are translated into logic programs and given to the ASP grounder and solver, respectively. The resulting answer sets represent maximal trajectories satisfying the axioms and the query.

For displaying trajectories, we developed a parser to translate the answer sets into the language *dot* [7], providing an easy and human readable way to specify graphs and being readable by various software packages. Among them, we have chose *rtgraph3d* [3] to display trajectories in a 3-dimensional view. When clicking on a state, the set of species present in the state is displayed (as shown in Figure 2).

For the sake of readability, the states of the system are displayed as spheres and colored depending upon whether they are initial (in red), final (in blue), both (in turquoise) or none (in black). The transitions are displayed as links between these spheres.

Finally, as our approach produces many trajectories which can be equivalent from a biologist's point of view, we moreover provide the following additional features:

- minimize the set of species present in the initial state (in order to remove species being irrelevant to the query);
- fix the maximum number of reactions in each transition (to avoid displaying many equivalent paths);
- fix whether or not the reactant may disappear (instead of the non-deterministic approach in (8));
- fix a set of relevant species to be shown in the answer set; the trajectories are then projected onto this set of relevant species, which provide a more abstract view.[3]

[2]`http://potassco.sourceforge.net`

[3]For the reader interested in ASP, we mention that this is accomplished with *clasp*'s projective enumeration, invoked with `--project`.

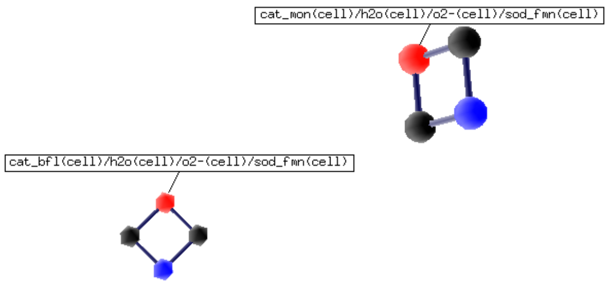

cat_mon(cell)/h2o(cell)/o2-(cell)/sod_fmn(cell)
cat_bfl(cell)/h2o(cell)/o2-(cell)/sod_fmn(cell)

Figure 2: Detoxification pathway of *Sinorhizobium Meliloti 1021.*: Two initial situations (in red) can lead to the detoxification of O_2^-.

Example 3 (Detoxification of O_2^-) *We now complete the detoxification pathway given in Example 1 by taking the whole detoxification pathway of a real bacteria,* Sinorhizobium Meliloti 1021 *[16]. Instead of facing hydrogen peroxide, the cell faces superoxide (viz. O_2^-). Superoxide is so toxic that intracellular levels above 1nM are lethal. This compound is one of the main causes of oxidative stress.*

To survive, the cell must have a superoxide dismutases (SOD) to transform superoxide into hydrogen peroxide and then some catalase (CAT) to transform hydrogen peroxide into water and oxygen. This is the case with almost all aerobic organisms, and thus applies to Sinorhizobium Meliloti 1021 *which has multiple isoforms of these enzymes.*

In Figure 2, we give a screenshot of our toolbox for this detoxification problem. We have taken the full detoxification network given in OxyGen [16] and the confidence levels given by OxyGen. The axioms specify that initially the superoxide is present in the cell and no oxidant is present in the final state (and the query is empty). Here, we show all maximal trajectories (in term of confidence levels) and then minimize the number of enzymes present initially.[4] *Furthermore, we restrict the trajectories to contain only one reaction per transition.*

Sinorhizobium Meliloti 1021 *contains one SOD and two isoforms of CAT that have been experimentally proved. In fact, it contains further isoforms of these enzyme (but not experimentally proved) and other enzyme of less interest. Since the proved enzyme are sufficient, we refrained from using the other ones and thus obtain only two possibilities (as shown in the figure).*

In this example, the trajectories are of length 2. Also, we have minimized the number of initial species. Figure 2 shows two initial states (in red), one state per catalase. For each initial state, there are two trajectories, one where the superoxide disappears after the first transition and the other where it disappears after the second step.

Example 4 (Glycolysis) *Next, we further elaborate on the glycolysis pathway, already described in Example 2. Recall that glycolysis takes glucose as input and after a first series of reactions, glycolysis has the choice to produce glycerol or not. Both cases result in the production of ethanol.*

[4]This is done with `#maximize` and `#minimize` statements provided by ASP systems.

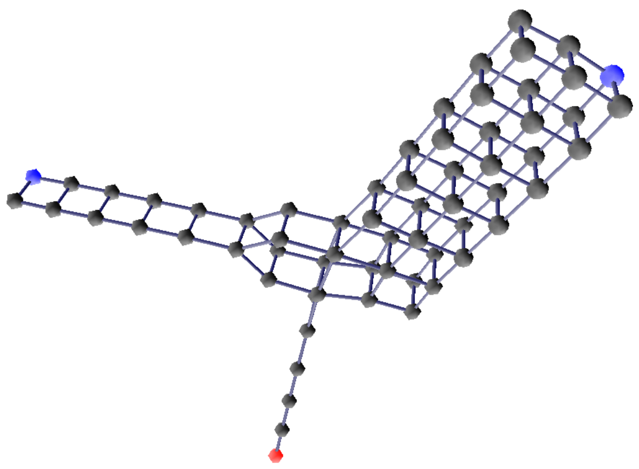


Figure 3: Glycolysis: The initial condition (where the glucose is outside, in red) can lead to two distinct cases (where glycerol is produced (left) or not (right)).

Figure 3 shows the output of our toolbox with this pathway. The axioms specify that initially glucose is present outside the cell along with each useful enzyme. The axioms also specify that ethanol is present outside the cell in the final state. Again, the query is empty.

For the sake of clarity, we have restricted trajectories to contain only one reaction per transition and every reactant of a reaction disappears after the corresponding reaction occurred. After a first series of reactions (in the bottom of Figure 3), the cell chooses to produce either only ethanol (in the left part) or to produce both ethanol and glycerol.

5 Application

We have used our method for identifying biological experiments in view of gathering new biological knowledge. As the networks are often incomplete or contain automatically generated reactions (which have not been proved), we aim at finding easy experiments that can prove parts of the network.

For this, we have used the whole detoxification networks of 655 bacteria given in OxyGen [16]. In OxyGen, each reaction (of each bacteria) is given a confidence level that expresses the accuracy of the annotation of the enzyme. There are three confidence levels in OxyGen:

- Enzymes that have been experimentally demonstrated, that is, found by comparison with a database of experimentally validated proteins (in this case, the confidence level is 3);
- Enzymes without biological evidence but for which the signature of the corresponding gene was found in the genome (in this case, the confidence level is 2);
- Enzymes from disrupted regions, like frameshifts[5] or pseudogenes[6] (in this case, the confidence level is 1).

[5]Two separate motifs of the same signature are found in two different frames of the same strand.

[6]One or two stops in frame.

Oxidant	3	2	1	0
O_2^-	4.7 %	86.5 %	0.3 % (2)	8.3 %
H_2O_2	12.8 %	84.8 %	0.1 % (1)	2.1 %
$R-OOH$	7.1 %	87.4 %	0 %	5.3 %
$ONOOH$	4.8 %	73.1 %	0.1 % (1)	21.8 %
NO	2.7 %	58.1 %	0 %	39.1 %

Table 1: For each oxidant, the bacteria are grouped by the confidence level needed to detoxify the oxidant.

We have used our method to find reactions of confidence level 1 which are mandatory to detoxify an arbitrary oxidant. Indeed, there is not so much knowledge gained in proving reactions of confidence level 2 because they are likely to exist. More interesting knowledge is obtained by asking whether reactions of level 1 exist. However, for those reaction, there often exist reactions of a higher confidence level that accomplish the same function. Proving them experimentally implies to knock-out a gene for all those reaction, or inhibit their effects and make the experiment more expensive. We thus search for reactions of level 1 which are mandatory, that is, there is no other reaction (or set of reactions) that accomplishes the same function.

To find them, we computed for each (network of each) bacteria and each oxidant, the maximal confidence level of the maximal trajectories that detoxifies the oxidant. This maximal level tells us the confidence level one can put on the detoxification of the oxidant for the given bacteria. For each oxidant, we have grouped the bacteria depending on this confidence level, and give this result in Table 1. For instance, for detoxifying H_2O_2, we noticed 12.8 % bacterias with confidence level 3, 84.8 % bacterias with confidence level 2, a single bacterium with confidence level 1, and 2.1 % bacterias that were unable to detoxify the oxidant at hand (relative to the knowledge comprised in OxyGen).

For a lot of bacteria, an enzyme of confidence level 2 is needed to detoxify the oxidant. For those bacteria, doing an experiment which shows that the detoxification occurs will only show that the annotation was accurate.

But we actually discovered four cases (two with O_2^-, one with H_2O_2, and one with $ONOOH$, indicated in parentheses in Table 1) where a reaction having a confidence level 1 is mandatory to detoxify an oxidant. For these bacteria, doing a simple experiment showing that the detoxification occurs will point out an unlikely yet existing reaction.

6 Related Work

Modeling methods for biological system fall into two categories, quantitative ones, focusing on measurable information, like concentration of molecules, and qualitative ones, focusing on the mere presence or absence of molecules.

The two major quantitative methods are Ordinary Differential Equations (ODEs) [6] and Flux Balance Analysis (FBA) [13]. In the former, the concentration of a metabolite is given by an ODE summarizing all reactions in which the metabolite is consumed or produced. FBA relies on the steady state assumption and models reactions by two matrices, one for stoichiometry and another for fluxes. In large scale networks, however, numerous parameters (in ODEs) and fluxes (in FBA) are unknown and thus estimated, so that despite the partly fine-grained input data the final results are prone to inaccuracy.

Among the qualitative approaches, we find petri nets. For modeling metabolic networks [15], compounds are modeled by places and quantities by tokens; reactions are transitions from reactants' places to products' places. Hybrid petri nets have been developed to use petri nets with differential equations. Along with petri nets, one can compute the "elementary modes", i.e. vectors of transition which forms the base of the petri net. Those vectors provide information on the topology of the network.

Biocham is a framework dedicated to biochemical reasoning [8]. Reactions are modeled by transformation rules which form a Kripke structure. Queries can be made in CTL and a symbolic model checker is used to solved them.

Action languages have already been used to model biological networks. Action language $\mathcal{A}$ [9] has been extended with triggers and inhibitions to model signaling networks [2]. This language is referred as $\mathcal{A}_T^0$ and has been further extended in [10] to form Language $\mathcal{C}_{TAID}$ featuring allowance statements and defaults.

In [17], the authors used an abductive logic programming system to revise metabolic pathways. Their method allows removing or adding new reactions, enzymes, or inhibition rules from a network and given observational data but they do not define any abstract language to express network and queries.

Between all those method, the Systems Biology Markup Language (SBML) is an attempt to model biological networks in a machine-readable format. It's applicable to models of metabolism, cell-signaling, and many others. Reactions are modeled as relations between chemical compound in compartments. Those relations can contain information on stoichiometry and differential equations. BioModels [14] is a database referencing all SBML models from the literature.

7 Discussion

We have presented a new action language dedicated to metabolic networks. This language handles chemical compounds in compartments and chemical reactions along with essential features as priorities and confidence level. We have applied our method to the detoxification pathway of 655 bacteria and found 4 reactions of interest.

With confidence level, our method facilitates the completion of networks. By giving a high confidence level to published reactions and a low one to non-validated reactions, our toolbox finds out essential non-validated reaction to accomplish some function. It shows where biological experiments can bring new knowledge.

As a next step, we plan to use more genericity in queries and compartments. The static time step in expression of our query language could be improved by the use of modal logic and allow more expressiveness. We also want to allow for the definition of types of compartments. Those types will allow for more general observations of the system.

Acknowledgements The second author was supported by the Federal Ministry of Education and Research within the GoFORSYS project (http://www.goforsys.org/; grant 0313924).

References

[1] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University, 2003.

[2] C. Baral, K. Chancellor, N. Tran, N. Tran, A. Joy, and M. Berens. A knowledge based approach for representing and reasoning about signaling networks. *Bioinformatics*, 20(1):15–22, 2004.

[3] P. Biondi. `http://www.secdev.org/projects/rtgraph3d/`.

[4] B. Bornstein, S. Keating, A. Jouraku, and M. Hucka. Libsbml: An API library for SBML. *Bioinformatics*, pages btn051+, 2008.

[5] R. Caspi, H. Foerster, C. Fulcher, P. Kaipa, M. Krummenacker, M. Latendresse, S. Paley, S. Rhee, A. Shearer, C. Tissier, T. Walk, P. Zhang, and P. Karp. The metacyc database of metabolic pathways and enzymes and the biocyc collection of pathway/genome databases. *Nucleic Acids Res*, 2007.

[6] N. Duarte, M. Herrgård, and B. Palsson. Reconstruction and validation of saccharomyces cerevisiae ind750, a fully compartmentalized genome-scale metabolic model. *Genome Res*, 14(7):1298–1309, 2004.

[7] S. North, E. Koutsofios. Drawing graphs with dot. Technical report, At&T.

[8] F. Fages, S. Sollman, and N. Chabrier-Rivier. Modelling and querying interaction networks in the biochemical abstract machine biocham. *Journal of Biological Physics and Chemistry*, 4:64–73, 2004.

[9] M. Gelfond and V. Lifschitz. Action languages. *Electronic Transactions on AI*, 3(6):193–210, 1998.

[10] S. Dworschak, S. Grell, V. Nikiforova, T. Schaub, and J. Selbig. Modeling biological networks by action languages via answer set programming. *Constraints*, 13(1-2):21–65, 2008.

[11] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G. Gopinath, G. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein. Reactome: a knowledgebase of biological pathways. *Nucleic Acids Res*, 33, 2005.

[12] M. Kanehisa and S. Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.

[13] K. Kauffman, P. Prakash, and J. Edwards. Advances in flux balance analysis. *Curr Opin Biotechnol*, 14(5):491–496, 2003.

[14] N. Le Novère, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. Sauro, M. Schilstra, B. Shapiro, J. Snoep, and M. Hucka. Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Res*, 34, 2006.

[15] J. Pinney, D. Westhead, and G. McConkey. Petri net representations in systems biology. *Biochem Soc Trans*, 31(Pt 6):1513–1515, 2003.

[16] D. Thybert, S. Avner, C. Lucchetti-Miganeh, A. Chéron, and F. Barloy-Hubler. Oxygene: an innovative platform for investigating oxidative-response genes in whole prokaryotic genomes. *BMC Genomics*, 9:637+, 2008.

[17] Ray, O., Whelan, K., King, R.: A nonmonotonic logical approach for modelling and revising metabolic networks. In: Proceedings CISIS'09, IEEE Press (2009) To appear.

A Coherence-Driven approach to Action Selection

Sindhu Joseph and **Carles Sierra** and **Marco Schorlemmer**
Artificial Intelligence Research Institute, IIIA-CSIC
Bellaterra (Barcelona), Catalonia, Spain
{joseph,sierra,marco}@iiia.csic.es

Abstract

In this paper, we propose a coherence-driven approach to action selection in agents. The mechanism is inspired by the cognitive theory of coherence as proposed by Thagard . Based on a proposal to extend BDI agents with coherence, we interpret, how action selection can be viewed as a coherence-maximising problem. Contrasted against the classical BDI approach to action selection where actions are selected against a pre-determined set of beliefs and desires, this method offers a dynamic view of the cognitions of an agent, where a set of beliefs desires and intentions are selected together to keep the coherence of the agent. We illustrate the approach by simulating how a coherence-driven robot selects its next action to pursue.

1 Introduction

A BDI-based reasoning process consists of a deliberative cycle in which an agent decides what state of affairs it wants to achieve from among all those desirable states of affairs [4; 15; 14]. The output of the deliberation process is a set of intentions (desires that the agent wants to pursue paired with a 'top-level' plan of action) [1]. Once the intentions are created and their associated preconditions (in the form of a set of beliefs) are met, then it is immediate that these intentions are realised.

As it should be apparent, there are a few major difficulties with this kind of reasoning. Among the many alternatives, it is not clear how a particular desire or a set of desires are chosen to be pursued further. In a graded cognitive agent, this could be done simply by selecting the desire that has the highest degree [3]. However, such a selection will not guarantee that the chosen desire is the best to be satisfied. To qualify for it, a desire should be consistent with most of the fundamental beliefs of the agent, and it should not conflict with other desires which are already in pursuit. Finally, this desire should be realisable. The last point is taken into account in the BDI deliberation cycle, however, not during the selection of the desire, rather at the point where intentions are generated [4; 15]. At this point, conflicts with other intentions may be discovered. In such cases, the plan is aborted and another plan or another desire itself has to be chosen. However, this is a very ad-hoc procedure, and unlikely to result in an optimised or coherent agent.

Alternatively, our intuition says that among the many alternatives, a desire should be selected that is not only most desired, but also most coherent with the agent's set of beliefs, other desires, and plans. The same is true to incorporate a new perception (belief). A new perception is incorporated only when it is coherent within the set of cognitions. The same happens when adopting a plan. That is, the model is essentially dynamic, where beliefs, desires and intentions are subject to the criterion of coherence maximisation. Here we propose to incorporate such a reasoning to artificial agents. We do so over the basic BDI architecture, but the process of deliberation and action selection is inspired by coherence maximisation inspired from Thagard's theory of coherence.

Seen in a broader context, the theory of coherence can draw parallels with other established theories. The philosophers of science have long argued about what "claims" in a theory can be supported. Popper's view on the progress of knowledge [9] sees falsifiability as the main driving force, and knowledge as an evolving body that follows a process in which a number of theories 'compete' to account for a problem situation. When a set of theories is set, falsification is then the process that makes some theories fail, while allowing others to survive. In his view survival does not mean truth but 'fitness' to the situation. The notion of truthlikeness is for Popper a notion of verosimilitude ($V(a) = T(a) - F(a)$) that accounts for the comparison between the truth content of theory a and the falsity content of a, which permits to rank theories. As we will see later in this paper this concept is similar to the notion of 'strength' of a partition in a coherence graph. Falsification of a theory can be associated to the introduction of a highly incoherent fact that will make certain statements to be removed from the accepted set of claims. Although Popper would reject a complete theory as soon as empirical evidence would go against it, Kuhn [10] would consider that scientists tolerate a certain level of anomalies (in our context a certain level of incoherence) for a long time until a revolution happens in which a complete new theory is accepted and an old one rejected. This latter phenomenon may be reproduced in our context, as we will see, by the fact that partitions in graphs can change abruptly when two theories are similarly coherent and a new experimental result is added leading to a swap in the set of accepted claims. The reconciliation point made by

Lakatos [11] would be that scientific theories contain a hard core that contains the most crucial claims of the theory plus a protective belt of auxiliary hypothesis that in case of contradiction with the facts will be modified or removed while keeping the central core, of course until a major difficulty is found that leads to a drastic change of the core. The use of degrees in claims and the algorithmic introduced in the paper will show that we might implement a similar mechanism by eliminating first the auxiliary hypothesis (those with lower degrees of belief) before removing the hard core ones (with higher probability degrees).

In the remaining of the paper, we introduce Thagard's theory of coherence, and the coherence framework used to explain action selection in Section 2. In Section 3 we explain the architecture of a coherence-driven agent and explain coherence-driven action selection. With the help of an example, we illustrate the theory in Section 4 and conclusion and future works are in Section 5.

2 Thagard's Theory of Coherence

In this section, we discuss the intuitions behind Thagard's Theory of Coherence and introduce a coherence framework based on this theory.

Paul Thagard is one of the philosophers who have attempted to introduce a computational interpretation of coherence. Thagard postulates that the theory of coherence is a cognitive theory with foundations in philosophy that approaches problems in terms of the satisfaction of multiple constraints within networks of highly interconnected elements [16; 17]. At the interpretation level, Thagard's theory of coherence is the study of associations, that is, how a piece of information influences another and how best different pieces of information can fit together. Each piece of information imposes constraints on others, the constraints being positive or negative. Positive constraints strengthen pieces of information, thereby increasing coherence, while negative constraints weaken them, thereby increasing incoherence. Hence, a coherence problem is to put together those pieces of information that have a positive constraint between them, while separating those having a negative constraint. Coherence is maximised if we obtain such a partition of information where a maximum number of constraints is satisfied.

Thagard's Formalisation

Thagard formalises coherence as follows [16]: The basic notions are that of a set of pieces of information which are represented as nodes in a graph $V = \{v_i\}$ and weighted links or constraints $E = \{\{v, w\}\}$ between these nodes. Further, some of these constraints are positive (C^+) and others negative (C^-) and associated with each constraint a number ζ which indicate the weight of the constraint. Given these, maximising coherence is formulated as the problem of partitioning V into two sets, $\mathcal{A}$ (accepted) and $\mathcal{R}$ (rejected), in a way that maximises compliance with the following two coherence conditions:

1. if $(v, w) \in C^+$ then $v \in \mathcal{A}$ if and only if $w \in \mathcal{A}$.
2. if $(v, w) \in C^-$, then $v \in \mathcal{A}$ if and only if $w \in \mathcal{R}$.

If $\{v, w\}$ complies with one of the above conditions, then, Thagard defines it as a satisfied constraint. Then the coherence problem is to maximise the sum of the weights of the satisfied constraints.

Thagard further proposes six main kinds of coherence: *explanatory, deductive, conceptual, analogical, perceptual, and deliberative*, each with its own array of elements and constraints. Once these elements and constraints are specified, then the algorithms that solve the general coherence problem can be used to compute coherence in ways that apply to specific domain problems.

2.1 Comparison with Other Decision Theories

Keeping Thagard's approach to coherence as maximising constraint satisfaction, we try to understand the main concept behind this theory. We associate coherence with an ever-changing system where coherence is the only property that is preserved, while everything around it changes. In cognitive terms, this would mean that, there are no beliefs nor other cognitions that are taken for granted or fixed forever. Everything can be changed and may be changed to keep coherence. We humans tend to revise or re-evaluate adherence to social norms, our plans, goals and even beliefs when we are faced with incoherence. We do not suppose that taking decisions based on coherence imply an unstable system. Our claim is based on the fact that some beliefs are more fundamental than others, in line with Lakatos. Revision of such fundamental belief is less frequent compared to other beliefs. In coherence terms, these beliefs are fundamental because they support and get support from most other cognitions and hence are in positive coherence with them. Hence, such beliefs will almost always be part of the chosen set while maximising coherence. The same is the case with other cognitions while the process of coherence maximisation further helps resolve conflicts by selecting among the best alternatives.

When applied to decision making, this means that we may not only select the set of actions to be performed to achieve certain fixed goals, but also look for the best set of goals to be pursued. Further, since coherence affects everything from beliefs to goals and actions, it may happen that beliefs contradicting a decision made are discarded. There are psychological theories such as cognitive dissonance [5] that explains this phenomenon as an attempt to justify the action chosen. Thus, with coherence we are looking at a more dynamic model of cognitions where one picks and choses goals, actions and even beliefs to fit a grand plan of maximising coherence. In concrete terms, a highly desired state of the world (preferred in a classical sense) may get discarded in front of a less desired state of the world because it is incoherent with the rest of the beliefs, desires or intentions.

As discussed in [16], this view of decision making is very different from those of classical decision making theories where the notion of *preference* is atomic and there is no conceptual understanding of how preferences can be formed. In contrast, coherence based decision making tries to understand and evaluate these preferences from the available complex network of constraints. The assumption here is more basic because the only knowledge available to us are the various interacting constraints between pieces of information.

2.2 Coherence framework

Since we consider coherence-driven agents, in this section we summarise a generic coherence framework that will allow us to build coherence-driven agents. The framework is introduced in the work of Joseph et al [8; 7] based on Thagard's theory. It differs from other coherence-based frameworks in extending agent theories [2; 13] as in this framework coherence is treated as a fundamental property of the cognitions of an agent. Further, it is generic and fully computational. In the following we briefly introduce the necessary definitions of this framework to understand the formulation of coherence-driven action selection. The intuition behind these definitions and a few examples are given in [8; 7]. The core notion is that of a *coherence graph* whose nodes represent pieces of information and whose weighted edges represent the degree of coherence or incoherence between nodes.

Definition 2.1 *A* coherence graph *is an edge-weighted undirected graph $g = \langle V, E, \zeta \rangle$, where*

1. *V is a finite set of nodes representing pieces of information.*
2. *$E \subseteq \{\{v, w\} | v, w \in V\}$ is a finite set of edges representing the coherence or incoherence between pieces of information, and which we shall call* constraints.
3. *$\zeta : E \to [-1, 1] \setminus 0$ is an edge-weighted function that assigns a negative or positive value to the coherence between pieces of information, and which we shall call* coherence function.

Every coherence graph is associated with a number called the *coherence of the graph.* Based on Thagard's formalism, this can be calculated by partitioning the set of nodes V of the graph in two sets, $\mathcal{A}$ and $V \setminus \mathcal{A}$, where $\mathcal{A}$ contains the accepted elements of V, and $V \setminus \mathcal{A}$ contains the rejected ones. The aim is to partition V such that a maximum number of constraints is satisfied, taking their values into account. A constraint is satisfied only if it is positive and both the end nodes are in the same set, or negative and the end nodes are in complementary sets. The following definitions help clarify this idea.

Definition 2.2 *Given a coherence graph $g = \langle V, E, \zeta \rangle$, and a partition $(\mathcal{A}, V \setminus \mathcal{A})$ of V, the* set of satisfied constraints *$C_{\mathcal{A}} \subseteq E$ is given by*

$$C_{\mathcal{A}} = \left\{ \{v, w\} \in E \;\middle|\; \begin{array}{l} v \in \mathcal{A} \text{ iff } w \in \mathcal{A} \text{ when } \zeta(\{v, w\}) > 0 \\ v \in \mathcal{A} \text{ iff } w \notin \mathcal{A} \text{ when } \zeta(\{v, w\}) < 0 \end{array} \right\}$$

All other constraints (in $E \setminus C_{\mathcal{A}}$) are said to be unsatisfied.

Definition 2.3 *Given a coherence graph $g = \langle V, E, \zeta \rangle$,* the strength of a partition *$(\mathcal{A}, V \setminus \mathcal{A})$ of V is given by*

$$\sigma(g, \mathcal{A}) = \frac{\sum_{\{v,w\} \in C_{\mathcal{A}}} | \zeta(\{v, w\}) |}{| E |}$$

Notice that, by Definitions 2.2 and 2.3,

$$\sigma(g, \mathcal{A}) = \sigma(g, V \setminus \mathcal{A}) \qquad (1)$$

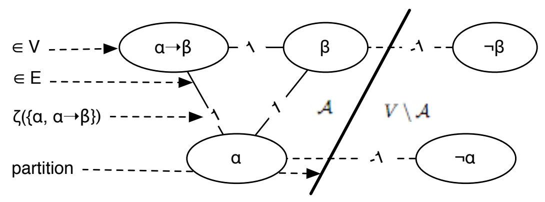


Figure 1: A typical coherence graph with a coherence maximising partition

Definition 2.4 *Given a coherence graph $g = \langle V, E, \zeta \rangle$ and given the strength $\sigma(g, \mathcal{A})$, for all subsets $\mathcal{A}$ of V,* the coherence of *g is given by*

$$\kappa(g) = \max_{\mathcal{A} \subseteq V} \sigma(g, \mathcal{A})$$

If for some partition $(\mathcal{A}, V \setminus \mathcal{A})$ of V, the strength of the partition is maximal (i.e., $\kappa(g) = \sigma(g, \mathcal{A})$ then the set $\mathcal{A}$ is called the accepted *set and $V \setminus \mathcal{A}$ the* rejected *set of the partition. A typical coherence graph is as shown in Figure 1.*

Due to Equation 1, the accepted set $\mathcal{A}$ is never unique for a coherence graph. Moreover, there could be other partitions that generate the same value for $\kappa(g)$. Here we mention a few criterias to select an accepted set among the alternatives. If $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n$ are sets from all those partitions that maximise coherence of the graph g, based on one of Thagard's principles (which we will formalise in the next definition) on deductive coherence[16] that *intuitively obvious propositions have an acceptability on their own*, we say an accepted set is the one in which the intuitively obvious propositions belong. Further, the coherence of the sub-graphs $(g|_{\mathcal{A}_i}, i \in [1, n])$ gives us an indication of how strongly connected they are. The higher the coherence,the more preferred the corresponding accepted set. And lastly, an accepted set with more number of elements should be preferred to another with less.

We now need a way in which the coherence graphs just defined can be constructed. That is, we need to define function ζ. As the nature of relationship between two pieces of information (corresponding to the different types of coherence as mentioned in the introduction) can vary greatly, we do not have one unique coherence function. That is, in an explanatory coherence, two pieces of information are coherent when they are related by an explanation. Thagard proposes certain principles to characterise coherence in each of the different types. Here we define one such coherence function which is inspired from Thagard's principles of *deductive coherence*.

Thagard's principle mainly states that *a proposition coheres with propositions that are deducible from it, propositions that are used together to deduce something cohere with each other, the more hypotheses it takes to deduce something, the less the degree of coherence, contradictory propositions are incoherent with each other*[1]. Since some of these principles make sense only in the context of a theory presentation,

[1]here we do not formalise the principle that *intuitively obvious propositions have a degree of acceptability on their own.* This we keep it as a disambiguation criteria to select among accepted sets.

we assume a theory presentation $\mathcal{T}$ in a multi-valued propositional logic while formalising these principles. We use a multi-valued logic to model uncertainty in agents, though Thagard's principles, we assume, are based on a boolean world. We formalise Thagard's principles in terms of a *support function* $\eta_{\mathcal{T}}$ which extract a coherence value between two nodes if either one implies the other, or together they are used to imply a third node (assuming some sort of deduction theorem such as $\mathcal{T}, \alpha \vdash \beta \quad$ implies $\quad \mathcal{T} \vdash \alpha \to \beta$). We also normalise the values between $[-1, 1]$.

Definition 2.5 *Let L be the set of all propositional sentences of a multi-valued propositional logic. Let $\mathcal{T} \subseteq L$ be a finite theory presentation and $\Gamma \subseteq \mathcal{T}$ and $\gamma \in L$. A support function $\eta_{\mathcal{T}} : L \times L \to [-1, 1]$ with respect to $\mathcal{T}$ is given by*

$$\eta_{\mathcal{T}}(\{\alpha, \beta\}) = \begin{cases} \max \begin{cases} \max \left\{ \frac{2 \cdot F_{\to}(\rho(\alpha), \rho(\beta)) - 1}{|\Gamma|} \middle| \right. \\ \quad \exists \Gamma \subseteq \mathcal{T} : \Gamma, \alpha \vdash \beta \,;\, \alpha \not\vdash \beta \}, \\ \\ \max \left\{ \frac{2 \cdot F_{\to}(\rho(\alpha), F_{\to}(\rho(\beta), \rho(\gamma))) - 1}{|\Gamma| + 1} \middle| \right. \\ \quad \exists \Gamma \subseteq \mathcal{T} : \Gamma, \alpha, \beta \vdash \gamma \,;\, \alpha, \beta \not\vdash \gamma \} \end{cases} \\ \\ \textit{undefined} \qquad \textit{otherwise} \end{cases}$$

where $F_{\to}$ is the truth connective defined for L and $\rho(\alpha)$ gives the truth value of α.

Thagard in his principles emphases the fact that, though a coherence value can be derived from the underlying implication relation, coherence functions are always symmetric. Due to this, even if there may only be a deductive relation in one direction, there will be a deductive coherence in both directions. Hence, we define the deductive coherence between two propositions as the value of the stronger η_T values.

Definition 2.6 *Let L be the set of all propositional sentences of a multi-valued propositional logic. Let $\mathcal{T} \subseteq L$ be a finite theory presentation and let $\eta_{\mathcal{T}} : L \times L \to [-1, 1]$ be a support function. A* deductive coherence function *$\zeta_{\mathcal{T}} : L \times L \to [-1, 1] \setminus \{0\}$ with respect to $\mathcal{T}$ is a partial function given by: For any pair (α, β) of formulas in L,*

$$\zeta_{\mathcal{T}}(\{\alpha, \beta\}) = \begin{cases} \max(\eta_{\mathcal{T}}(\alpha, \beta), \eta_{\mathcal{T}}(\beta, \alpha)) \\ \quad \textit{if } \eta_{\mathcal{T}}(\alpha, \beta) \textit{ and } \eta_{\mathcal{T}}(\beta, \alpha) \textit{ defined, } \neq 0 \\ \\ \eta_{\mathcal{T}}(\alpha, \beta) \\ \quad \textit{if } \eta_{\mathcal{T}}(\alpha, \beta) \textit{ defined and } \neq 0 \\ \quad \textit{and } \eta_{\mathcal{T}}(\beta, \alpha) \textit{ undefined or } = 0 \\ \\ \eta_{\mathcal{T}}(\beta, \alpha) \\ \quad \textit{if } \eta_{\mathcal{T}}(\beta, \alpha) \textit{ defined and } \neq 0 \\ \quad \textit{and } \eta_{\mathcal{T}}(\alpha, \beta) \textit{ undefined or} = 0 \\ \\ \textit{undefined} \\ \quad \textit{if } \eta_{\mathcal{T}}(\alpha, \beta) \textit{ and } \eta_{\mathcal{T}}(\beta, \alpha) \textit{ are undefined} \\ \quad \textit{or } = 0 \end{cases}$$

Note that both the support function and the deductive coherence function are partial functions. This is because we interpret zero coherence as the propositions not being related.

3 Coherence-driven Agent Architecture

A *coherence-driven agent* is an agent which always takes an action based on maximisation of coherence of its cognitions, norms and other social commitments. Further, these are cognitive agents based on BDI theory [14] and are modeled as a multi-context architecture (developed by Casali et al. [3]), which consists of a set of contexts and a set of bridge rules between contexts. Each context has its own language, logic and theory expressed as coherence graphs. Bridge rules turn formulae derivable in one or more contexts into premises for derivations for another context. We assume that each agent has its beliefs, desires, and intentions stored in its belief context C_B, desire context C_D, and intention context C_I.

3.1 Cognitive Contexts

Here we briefly describe how a belief context C_B is defined while desire C_D and intention C_I contexts are similar [8; 3]. C_B consists of a belief logic and a theory $\mathcal{T}_B$ of the logic expressed as a coherence graph.

A belief logic $\mathcal{K}_B$ consists of a belief language, a set of axioms and a deductive relation defined on the belief logic $\langle L_B, A_B, \vdash_B \rangle$. The belief language L_B is defined by extending the classical propositional language L defined upon a countable set of propositional variables PV and connectives $(\neg, \to)$. L is extended with a fuzzy unary modal operator B. The modal language L_B is built from the elementary modal formulae $B\varphi$ where φ is propositional, and truth constants r, for each rational $r \in \mathbb{Q} \cap [0, 1]$, using the connectives of Łukasiewicz many-valued logic. If φ is a proposition in L, the intended meaning of $B\varphi$ is that "φ is believable". A modal many-valued logic based on Łukasiewicz logic is used to formalise $\mathcal{K}_B$[2].

Definition 3.1 *[3] Given a propositional language L, a belief language L_B is given by:*

- *If $\varphi \in L$ then $B\varphi \in L_B$*
- *If $r \in \mathbb{Q} \cap [0, 1]$ then $\overline{r} \in L_B$*
- *If $\Phi, \Psi \in L_B$ then $\Phi \to_L \Psi \in L_B$ and $\Phi \& \Psi \in L_B$ (where $\&$ and $\to_L$ correspond to the conjunction and implication of Łukasiewicz logic)*

We call $\mathcal{T}_B$ a theory in the language L_B.

Other Łukasiewicz logic connectives for the modal formulae can be defined from $\&$, $\to_L$ and $\overline{0}$: $\neg_L \Phi$ (defined as $\Phi \to_L \overline{0}$). Formulae of the type $\overline{r} \to_L \Psi$ (the probability of φ is at least r) will be denoted as (Ψ, r).
The axioms A_B of $\mathcal{K}_B$ are:

1. All axioms of propositional logic.
2. Axioms of Łukasiewicz logic for modal formulas (for instance, axioms of Hájek's Basic Logic (BL) [6] plus the axiom: $\neg\neg\Phi \to \Phi$.)
3. Probabilistic axioms, given $\varphi, \psi \in L$:
   - $B(\varphi \to \psi) \to_L (B\varphi \to B\psi)$
   - $B\varphi \equiv \neg_L B(\varphi \wedge \neg\psi) \to_L B(\varphi \wedge \psi)$

The deduction rules defining $\vdash_B$ of $\mathcal{K}_B$ are Modus ponens and Necessitation for B (from φ derive $B\varphi$).

Note that the truth function $\rho : L_B \to [0, 1]$ is defined by means of the truth-functions of Łukasiewicz logic and the probabilistic interpretation of beliefs as follows:

[2]We could use other logics as well by replacing the axioms.

- $\rho((B\varphi, r))^3 = r$ for all $r \in \mathbb{Q} \cap [0, 1]$
- $\rho(\varphi \,\&\, \psi) = max(\rho(\varphi) + \rho(\psi) - 1, 0)$ for all $\varphi, \psi \in L_B$
- $\rho(\varphi \rightarrow_L \psi) = min(1 - \rho(\varphi) + \rho(\psi), 1)$ for all $\varphi, \psi \in L_B$

Then a coherence graph over beliefs is defined over the belief logic $\mathcal{K}_B$ as follows:

Definition 3.2 *Given a belief logic $\mathcal{K}_B = \langle L_B, A_B, \vdash_B \rangle$ where L_B is a belief language, A_B are a set of axioms and $\vdash_B$ are a set of deduction rules, a belief coherence graph $g_B = \langle V_B, E_B, \zeta_B \rangle$ is a coherence graph defined over $\vdash_B$ and a finite theory $\mathcal{T}_B$ of L_B such that:*

- *$V_B \subseteq \mathcal{T}_B$*
- *E is a set of subsets of 2 elements of V_B*
- *$\zeta_{\mathcal{T}_B}$ is defined over $\vdash_B$ and $\mathcal{T}_B$.*

A belief coherence graph exclusively represents the graded beliefs of an agent and the associations among them. A desire coherence graph (g_D), and an intention coherence graph (g_I) over logics L_D, and L_I are similar.

3.2 Bridge Rules

Bridge rules are inference rules of the form $b = \frac{C_1:\psi, C_2:\psi}{C_3:\psi}$ whose premises and conclusion are labelled formulas where the labels denote the contexts they are taken from. They carry inferences between theories of different logics. Since our theories become coherence graphs, we need two functions to emulate the execution of bridge rules over coherence graphs. If $\mathcal{G}$ denote the set of all coherence graphs, then a graph node extension function ($\varepsilon : \mathcal{G}^n \rightarrow \mathcal{G}^n$) takes into account the influence of graphs (theories) on each other. An edge extension function ($\iota : \mathcal{G}^n \rightarrow \mathcal{G}$) joins a set of graphs by adding edges between the nodes participating in the inference. Since we treat bridge rules similar to any other implication relations, we use Definition 2.6 itself to calculate the coherence values on these edges. We now illustrate the concept of bridge rules when the contexts are coherence graphs (the formal definitions can be found in [8]).

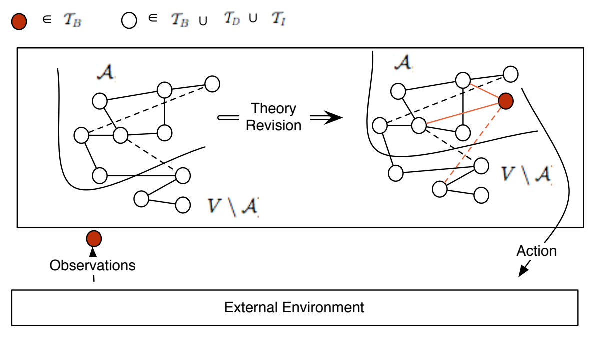
Example 3.1 *Let's assume, for instance, that an agent wants it to be the case that whenever it has an intention $(I\varphi, r)$ in the intention graph (a formula in the theory $\mathcal{T}_I$), then the corresponding belief $(B\varphi, r)$ is inferred in the belief graph (added to the theory $\mathcal{T}_B$). i.e., Given a bridge rule $b = \frac{C_B:(B\psi,r), C_D:(D\psi,s)}{C_I:(I\psi,\min(r,s))}$ where contexts C_B, C_D, and C_I have the coherence graphs g_B, g_D and g_I associated with them respectively and given $(B\psi, 0.95) \in g_B$, $(D\psi, 0.95) \in g_D$ function ε adds a node $(I\psi, 0.95)$ to g_I.*

Let's further assume that our agent further wants it to be the case that, the belief and the intention nodes are related and have a positive coherence between them. The edge extension function ι joins the graphs g_B, g_D and g_I associated with the contexts in the bridge rule by adding the edges $\{\{(I\psi, 0.95), (B\psi, 0.95)\}, \{(I\psi, 0.95), (D\psi, 0.95)\}\}$ with coherence values equal to $\frac{2\cdot\rho((I\psi,0.95))-1}{1} = 0.9$ from Definition 2.5.

$^3(B\varphi, r) \equiv \bar{r} \rightarrow B\varphi$

3.3 Architecture

Figure 2 shows the architecture of a coherence-driven agent. At any time, it can either perceive the environment (updates beliefs) or make a decision about a future action. In the event of a new information, an agent re-evaluates its theory, hence recomputes both the coherence graphs and the coherence maximising partition. If the new information falls in the accepted set then it reinforces the theory and the theory becomes more coherent. However, if it falls in the rejected set, then it contradicts some of the elements of the accepted theory to make the theory more coherent again in line with Lakatos. In the process, some of the existing elements may move from accepted to rejected or vice versa. An agent always bases its decisions on the accepted theory.



● ∈ $\mathcal{T}_B$
○ ∈ $\mathcal{T}_B \cup \mathcal{T}_D \cup \mathcal{T}_I$
$\mathcal{A}$
Theory Revision
$\mathcal{A}$
$V \setminus \mathcal{A}$
$V \setminus \mathcal{A}$
Observations
Action
External Environment

Figure 2: A coherence-driven agent architecture

3.4 Coherence-driven Action Selection

As discussed in the introduction, the philosophy behind a coherence-driven action selection is substantially different from other typical goal-driven approaches to action selection. A BDI-based agent, at anytime selects a goal to pursue, and looks for what actions would satisfy that goal. It is argued that, this would reduce the attention problem of the agent, giving it a stable behaviour. However, as argued in the introduction, this has many difficulties such as incorporating new perceptions, analysing conflicts between goals, and analysing feasibility of actions to achieve goals. This is due to the philosophical grounding of the theory for which the basis for an action is the expectation of a desired outcome.

As suggested in the introduction, coherence-driven reasoning offers a more holistic view on action selection. The philosophical theories of action suggest that an agent is influenced by a reason, and his action is consequently performed for that reason, when he is influenced by a representation of the action that makes it intelligible to him. Naturally, this representation may make the action intelligible precisely by setting it in the context of his desires and expectations, but his reason for action consists in this cognitively attractive representation of it rather than in the desires and expectations to which it alludes. A reason is a rationale, in the light of which an action makes sense to an agent, and promoting a desired outcome is one such rationale [18]. The coherence-driven approach we propose here attempts to capture this representation, which gives the agent the necessary rationale for action.

At any time a coherence-driven agent selects the most preferred action from its current accepted set of a coherence maximising partition. Any external stimuli interrupts the action selection process and forces the representation of the cognitions to go through a re-evaluation of coherence, resulting in some of the currently accepted cognitions to be rejected and vice versa. A procedure that a typical coherence-driven agent follows is outlined in the following.

Given the current coherence graphs g_B, g_D, and g_I and their composition ς_g and an external stimuli $(K\varphi, r)$ where $K \in \{B, D, I\}$

```
1: if $(K\varphi, r)$ then
2:     $v := (K\varphi, r)$
3:     $V_K := V_K \cup \{v\}$
4:     for all $w \in V_K$ do
5:         compute $\zeta(\{v, w\})$ using Definition 2.6.
6:         if $\zeta(\{v, w\})$ is defined then
7:             $E_K := E_K \cup \{\{v, w\}\}$
8:         end if
9:     end for
10:    compute a composite coherence graph $\varsigma_g$ as in [8] and
       Example 3.1.
11:    for all $(\mathcal{A}_i, V \setminus \mathcal{A}_i)$, $\mathcal{A}_i \subseteq V$ do
12:        calculate $\sigma(\varsigma_g, \mathcal{A}_i)$ using Equation 2.3
13:    end for
14:    $\kappa := \kappa(\varsigma_g)$ using Equation 2.4
15:    $\mathcal{A} := \mathcal{A}_i | \max(\sigma(\varsigma_g, \mathcal{A}_i))$
16: end if
17: current_action $:= \max_r\{(I\varphi, r) | (I\varphi, r) \in \mathcal{A}\}$
```

Line 1 checks for external stimuli. If there are any, then, lines from 2 to 9 updates the graphs by incorporating the stimuli and its influences on existing elements of the observed cognition. Line 10 builds up the reasoning across contexts by composing the coherence graphs. Lines from 11 to 14 determines the coherence maximising partition. This is done by first computing the strength of each partition using the function σ and choosing the partition $(\mathcal{A}, V \setminus \mathcal{A})$ for which $\sigma(g, \mathcal{A})$ is maximal. This part of the algorithm only gives the simplest solution, however, finding a maximising partition of a weighted graph is known to be an NP-complete problem. There are approximation algorithms exist to find the solution to this problem such as max-cut, neural network based algorithms. Line 17 determines the current action by selecting the action from $\mathcal{A}$ which has the highest preference.

4 Example

We consider a simple example to show how action selection works in our architecture. A coherence-driven robotic agent wants to choose between a set of possible actions(intentions) corresponding to a set of desires (goals) it has. The scenario is modelled like a grid in which at each cell the robot can chose between two possible actions: "plug" to restore its energy or "move" to earn points. It is further assumed that at every cell in the grid, it is possible to perform both actions. With every move the robot gains a point. Finally, the robot is equipped with an energy sensor, which measures the remaining energy at every time point, which influences the choices of the robot. The results are based on an implementation of a heuristic-based polynomial-time approximation algorithm to compute partitions and their corresponding coherences.

Since coherence maximisation dynamically choses the most coherent partition, the robot at any instant choses the set of beliefs, desires and intentions (actions) that it wants to pursue. There is one single persistent desire for the robot, which is to earn points. It has certain domain knowledge which indicates how to get its desire satisfied. This domain knowledge is encoded as a belief $(B(move \rightarrow points), 1)$ says that a move will fetch a point with a confidence degree 1. A bridge rule b_1 is used to reason with the beliefs and desires.

$$b_1 = \frac{C_B : (B(p \rightarrow q), \alpha), C_D : (Dq, \beta)}{C_D : (Dp, \min(\alpha, \beta))}$$

b_1 injects a new desire p given the desire of q and a belief that p facilitates q with appropriate degrees. Further, using b_1 and the belief that *having energy enables move, i.e.,* $(B(energy \rightarrow move), 1)$, a new desire to have "energy" is generated. A third desire to "plug" is generated using the bridge rule and the belief that *plugging gives energy, i.e,* $(B(plug \rightarrow energy), 1)$. The chain of desires and their coherence links are illustrated in Figure 3.

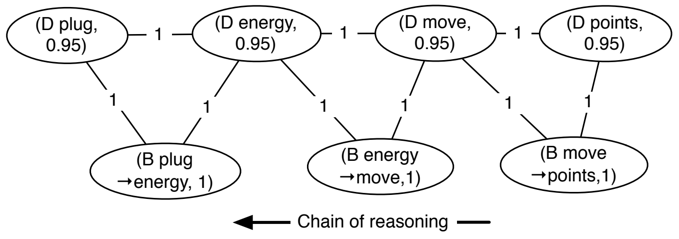


Figure 3: How one desire triggers another

The robot uses a second bridge rule b_2 that states that every desire with a corresponding belief that the desire is achievable, generates a corresponding intention (realistic agent).

$$b_2 = \frac{C_B : (Bp, \alpha), C_D : (Dp, \beta)}{C_I : (Ip, \min(\alpha, \beta))}$$

Using this rule, it has the intention to move, intention to have energy and intention to plug. Further, as in the case of desires, the intention to move is connected to the intention to have energy using another bridge rule b_3 which is used to reason across beliefs and intentions.

$$b_3 = \frac{C_B : (B(p \rightarrow q), \alpha), C_I : (Dq, \beta)}{C_I : (Dp, \min(\alpha, \beta))}$$

Note that bridge rules b_1 and b_3 are very similar and motivated from the well known practical syllogism, "If I want q and p realises q, then I should intend to do p". Using b_3, we have that *the belief* $(B(energy \rightarrow move), 1)$*,* $(Ienergy, x)$ *and* $(Imove, x)$ *are coherently related.* The same is true of $(Ienergy, x)$ and $(Iplug, x)$. Hence, similar to desires, a chain of intentions and their coherence links are generated (Figure 4).

As the only sensor for the robot (other essential sensors ignored) relevant to the problem is the *energy_sensor*

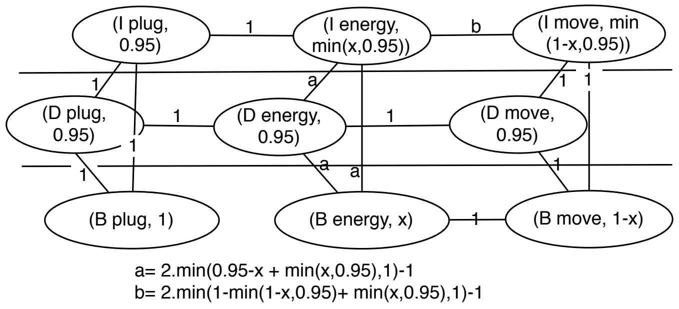


Figure 4: Desires and intentions trigger other intentions

(e_s), at every time point a few of the cognitions gets affected due to the changes in sensor readings. That is, we take that the grade on the belief that the quantity of energy needed changes inversely to the value of e_s . Further, the belief that move is possible changes proportionally with the value of e_s, using the modes ponens as $(B(energy \rightarrow move), 1), (Benergy, x) \rightarrow (Bmove, 1 - x)$. Finally, as it is assumed that the robot can perform only one action at a time, the essential conflict between intentions to "move" and "plug" are expressed as $(Imove, x) \Leftrightarrow (I\neg plug, x)$.

4.1 Action Selection

Given our robotic agent as described, we now pose the problem of action selection. That is, the robot has to decide what action to perform at every time point. We say an "energy-cycle" is the time between two consecutive "plug" actions. We take different energy levels and determine both the coherence of the robot and the coherence-driven choice of action. To understand how the coherence graph would look like, we show the graph with the partition when the $x = 0$ (just plugged) in Figure 5 and when $x = 1$ (there is no energy left) in Figure 6. In the case $x = 0$, there is a clear partition with the only intention selected is $(Imove, 0.95)$. Hence it is absolutely certain that, the robot should chose to move and earn points. The coherence of the graph for this partition is 0.6533.

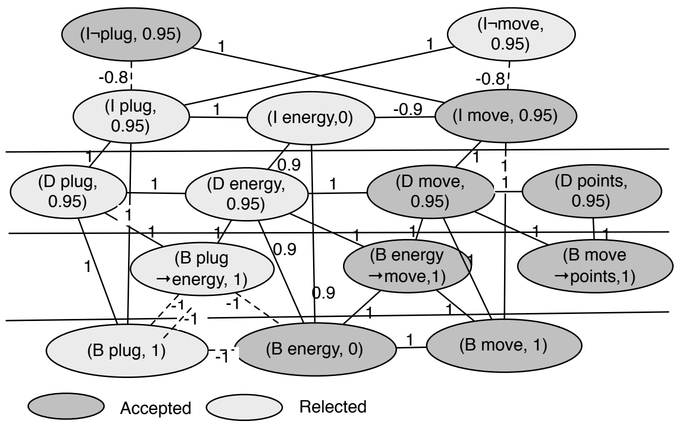


Figure 5: x=0 (Robot has maximum energy)

In the case $x = 1$ (Figure 6), there are no incoherence relations in the graph. However, it is due to the fact that, we are deriving the belief about move from the belief about energy needs. Though every node is part of the accepted set, notice that the node with the highest grade will be pursued. In this case the choice to plug will be pursued. The coherence of the graph for this partition is 0.97777. The increase in coherence is due to the fact that, there are no incoherence experienced by the robot and hence all nodes are accepted. Most of the individual coherence values are equal to the maximum possible $(= 1)$.

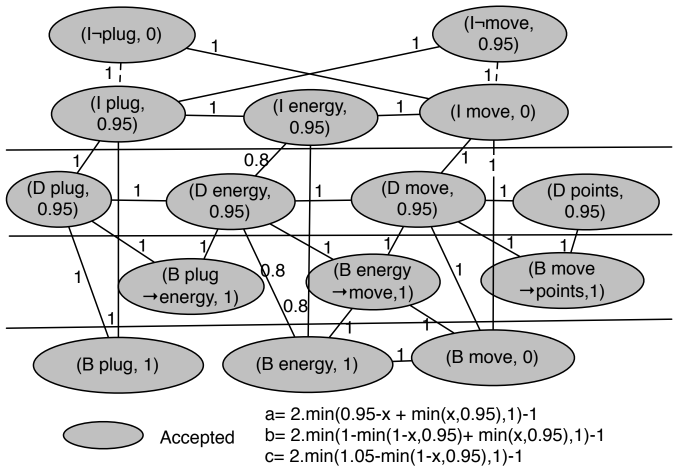


Figure 6: x=1 (Robot has no energy)

These are the two extreme cases, and now we plot the behaviour of the robot in terms of the choice of action, and the variation in coherence values at different energy levels in two *energy-cycles*. As seen in the coherence graphs in Figures 5 and 6, when the energy requirement is 0 or close to 0, the robot has some incoherences and selects only few of the cognitions as accepted. However, as the energy requirement increase, these incoherences disappear (due to the decreasing intention for action "move") and hence the robot becomes increasingly coherent with the action to plug.

Another graph which is interesting is the energy levels versus the choice of action as in Figure 8. This shows the expected actions of the robot at different energy levels. When the energy need is in the range $[0, 0.5]$ the robot choses to move. However, if the energy need is in the range $[0.6, 1]$, then the robot choses to plug and restore the energy. Then, its clear that as soon as the energy need raises to 0.6, the robot take the action to plug. Thus, the energy need never raises to a point beyond 0.6 (conservative behavior). Hence, the actual behavior of the robot will be a repeating sequence of $\{$*Move, Move, Move,* $\cdots$, *Plug, Move, Move,* $\cdots\}$, as the intuition would make us expect.

5 Discussion and Futurework

In this paper, we have introduced an alternative approach to action selection based on coherence maximisation. The interesting aspects of this approach over more traditional BDI

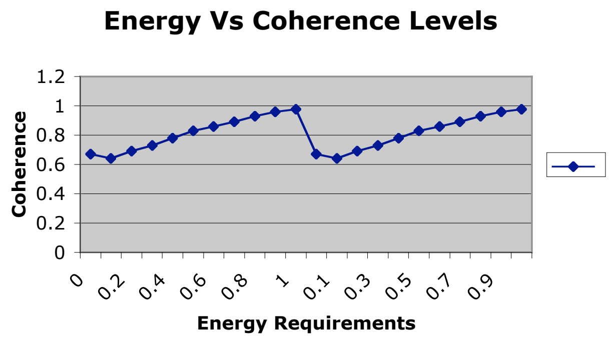


Figure 7: Variation in coherence with different energy levels

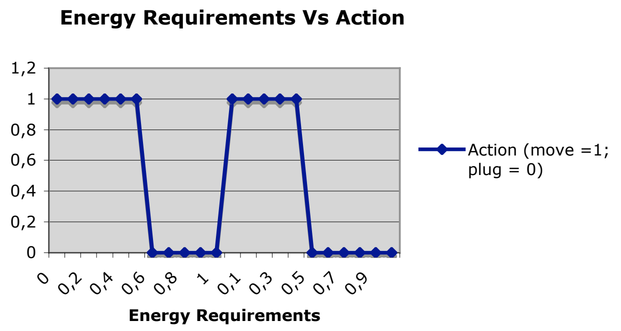


Figure 8: Action selection with different energy levels

approaches are that it takes a dynamic view of agent cognitions, can detect and resolve conflicts among cognitions, can perform uncertainty reasoning and can reason at a global level while also fully integrated into the BDI representation. Since we have discussed related work in the course of presenting the paper, we here make a brief comment on one related work, which is the only work known to us that uses coherence for agent reasoning. While the work of Pasquier et. al [13; 12], introduced coherence based reasoning in agents, there are significant differences with our proposal. In their work, coherence is like a utility maximising function, which is used to prioritise the intentions (dialogue moves), whereas reasoning about beliefs and desires are using the traditional BDI approach. This we imagine will retain all the difficulties we mentioned in the introduction. Another important difference is that, while we show how coherence can be computed using Thagard's principles, such mechanisms are missing from their approach.

In the future work, we plan to incorporate the representation of plans and study how plans can be included in the coherence maximising process. Further, we plan to explore the possibilities of evaluating our framework using empirical and mathematical proof.

References

[1] M. E. Bratman. *Intention, Plans, and Practical Reason*. CSLI publications, 1987.

[2] J. Broersen, M. Dastani, J. Hulstijn, Z. Huang, and L. van der Torre. The BOID architecture: conflicts between beliefs, obligations, intentions and desires. In *AGENTS '01*. ACM, 2001.

[3] A. Casali, L. Godo, and C. Sierra. Graded BDI models for agent architectures. In *Lecture Notes in Computer Science*, volume 3487, 2005.

[4] M. Dastani, F. de Boer, F. Dignum, and J.-J. Meyer. Programming agent deliberation: an approach illustrated using the 3apl language. In *AAMAS '03*, pages 97–104. ACM, 2003.

[5] L. Festinger. *A theory of cognitive dissonance*. Stanford University Press, 1957.

[6] P. Hájek. Metamathematics of fuzzy logic. In *Trends in Logic*, volume 4, 1998.

[7] S. Joseph, C. Sierra, and M. Schorlemmer. A coherence based framework for institutional agents. In *Lecture Notes in Computer Science*, volume 4870, 2007.

[8] S. Joseph, C. Sierra, M. Schorlemmer, and P. Dellunde. Formalising deductive coherence: An application to norm evaluation. In *Normas'08(Extended version)*, 2008.

[9] K.Popper. *Conjectures and Refutations: The Growth of Scientific Knowledge*. Basic Books, 1962.

[10] T. S. Kuhn. *Conjectures and Refutations: The Growth of Scientific Knowledge*. Chicago: University of Chicago Press, 1962.

[11] I. Lakatos. *Conjectures and Refutations: The Growth of Scientific Knowledge*. Cambridge University Press, 1976.

[12] P. Pasquier, N. Andrillon, M.-A. Labrie, and B. Chaib-draa. An exploration in using cognitive coherence theory to automate bdi agents communicational behavior. In *Advances in Agent Communication*. Springer, 2004.

[13] P. Pasquier and B. Chaib-draa. The cognitive coherence approach for agent communication pragmatics. In *Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003.

[14] A. S. Rao and M. P. Georgeff. Bdi agents: From theory to practice. In *ICMAS–95, First International Conference on Multi-Agent Systems: Proceedings*, pages 312–319. MIT Press, 1995.

[15] Y. Shoham. Agento: a simple agent language and its interpreter. In *Proceedings of AAAI*, 1993.

[16] P. Thagard. *Coherence in Thought and Action*. MIT Press, 2002.

[17] P. Thagard. *Hot Thought*. MIT Press, 2006.

[18] J. D. Velleman. *Self to Self: Selected Essays*. Cambridge University Press,, 2005.

Abductive Reasoning with Filtered Circumscription*

Martin Magnusson and Jonas Kvarnström and Patrick Doherty
Department of Computer and Information Science
Linköping University, 581 83 Linköping, Sweden
{marma,jonkv,patdo}@ida.liu.se

Abstract

For logical artificial intelligence to be truly useful, its methods must scale to problems of realistic size. An interruptible algorithm enables a logical agent to act in a timely manner to the best of its knowledge, given its reasoning so far. This seems necessary to avoid analysis paralysis, trying to think of every potentiality, however unlikely, beforehand. These considerations prompt us to look for alternative reasoning mechanisms for filtered circumscription, a nonmonotonic reasoning formalism used e.g. by Temporal Action Logic and Event Calculus. We generalize Ginsberg's circumscriptive theorem prover and describe an interruptible theorem prover based on abduction that has been used to unify planning and reasoning in a logical agent architecture.

1 Introduction

The world around us is uncertain. In fact, we have to cope with "pervasive ignorance" [Pollock, 2008] about most things. This is possible by reasoning defeasibly rather than purely deductively. But the world is also dynamic. Even when we *do* have all the relevant knowledge, we may not have time to think through all its consequences before the changing circumstances make our conclusions obsolete. This is most evident when planning our actions. Unless there is great risk involved, we most often carry out our plans after considering only a small subset of their consequences.

If we want to build logical agents that act autonomously to solve real world problems, we have to equip them with similar mechanisms to cope. Moving from simple benchmark problems to problems of realistic size has proven difficult due to the intractability of logical reasoning. An interruptible algorithm enables an agent to act in a timely manner, to the best of its knowledge given its reasoning so far. This seems a necessary feature of any nonmonotonic reasoning mechanism aimed to scale towards solving real world problems involving very large knowledge bases.

These considerations prompt us to look for alternative reasoning mechanisms for filtered circumscription, a nonmonotonic logic formalism for reasoning about action and change used e.g. by Temporal Action Logic (TAL). Regular theorem provers are not directly applicable to TAL's second-order circumscription axiom. This hinder has usually been overcome by applying predicate completion [Doherty and Kvarnström, 2007] to produce a first-order equivalent theory. But predicate completion involves a potentially costly computation applied to the entire knowledge base before any reasoning can begin. Moreover, the transformation must be recomputed whenever the agent's beliefs change, even e.g. when considering the effects of an action while planning. Finally, the reasoning involved is not interruptible. Predicate completion works by turning defeasible reasoning into deductive proof. These proofs must consider all potential objections to a defeasible conclusion before any answer can be given.

We extend Ginsberg's circumscriptive theorem prover [1989] to *filtered* circumscription. This forms the basis for an interruptible theorem prover based on abduction that operates on the Temporal Action Logic formulas directly, without any compilation step. We show how the same reasoning mechanism can be used to perform abductive planning, providing a unified planning and reasoning framework in a logical agent architecture. Such an agent could act in an any-time manner, using tentative answers based on defeasible assumptions if forced to act quickly, while still considering all potential objections given sufficient time for deliberation.

2 Preliminaries

While the results in this paper should be interesting for other logics of action and change, such as the Event Calculus, we focus on Temporal Action Logic and hence give a brief introduction to it. Similarly, while different proof systems could be used, our work implements the natural deduction system introduced below.

2.1 Temporal Action Logic

Temporal Action Logic is a highly expressive logic for reasoning about action and change. The origins of TAL are found in Sandewall's Features and Fluents framework [1994].

*This work is supported in part by the Swedish Foundation for Strategic Research (SSF) Strategic Research Center MOVIII, the Swedish Research Council Linnaeus Center CADICS, VR project (50405001, 50405002), and CENIIT, the Center for Industrial Information Technology.

Sandewall classified different variants of non-monotonic reasoning according to the types of reasoning problems for which they are applicable. TAL is a stand-alone logic based on one of the most general variant of these.

A central concept in TAL is *occlusion*. It is introduced as a flexible way to deal with the frame problem and related problems. The basic idea is to make fluent values, given by $Holds(time, fluent, value)$, persist over time by minimizing their opportunities for change. A predicate *Occlude(time,fluent)* represents the possibility of a fluent changing its value. This is a key difference from earlier attempts to use circumscription to minimize *change* rather than *potential for change*. Negated occlusion is then the property of a fluent not being able to change its value from the previous time point, i.e. persistence. A fluent f's default persistence at all time points (assuming unbound variables are implicitly universally quantified) is then axiomatized by:

$$\neg Occlude(t+1, f) \rightarrow (Holds(t, f, v) \leftrightarrow Holds(t+1, f, v))$$

Detailed control over fluent persistency can be exercised by adding similar persistence formulas, collectively denoted by T_{per}.

Situations that are known to cause a fluent's value to change must also occlude that fluent. E.g., actions must explicitly occlude affected fluents. We do not wish, however, to enumerate all situations in which a fluent is *not* occluded. The assumption is that, by default, things do not change without a reason. The Features and Fluents framework used *preferential entailment* to enforce this. Logical consequence is defined only w.r.t. models in which the extension of *Occlude* is minimal. Action occurrences, specified by the $Occurs(time_{start}, time_{end}, action)$ predicate, must also be minimized to prevent occlusion by spurious actions. But the question of how to compute the intended consequences was left open.

TAL provides a *syntactic* characterization using a form of circumscription called *filtered circumscription* [Doherty and Lukaszewicz, 1994], also referred to as *forced separation* [Shanahan, 1997]. Circumscription is used to minimize *Occlude* and *Occurs*, while fixing *Holds*, but T_{per} is forcedly separated from the rest of the theory T, outside the scope of the circumscription:

$$Circ(T; Occlude, Occurs) \land T_{\text{per}}$$

If the persistence formulas T_{per} had been included in the circumscription, then the extensions of *Occlude* had not been made any smaller. To see this, note e.g. that the contrapositives of formulas in T_{per} would have said that fluent change is *by itself* a reason for occlusion. Removing these formulas from the circumscription leaves only occlusion with some *explicit* cause, such as being occluded by an action's effects. The filtering occurs when we add T_{per} to the minimized theory, removing all models in which a fluent changes despite not being occluded.

This short introduction to TAL is intended to aid understanding of the rest of this paper. A more detailed presentation with a complete list of TAL's features is available elsewhere [Doherty and Kvarnström, 2007].

2.2 Natural Deduction

Example proofs will be presented in Suppes' style natural deduction [Pelletier, 1999]. Each proof row consists of a premise set, a row number, the formula, and a list of row numbers identifying the previous proof rows used by the current inference step (or empty for given input formulas). The premise set is a convenient bookkeeping device that keeps track of the assumptions that a formula depends on. This is important, not only for natural deduction's ability to construct proofs using temporary assumptions, but also during abductive proofs to label formulas by the set of ground instances of abducibles required. An assumption depends only on itself and thus its premise set only contains its own row number. Inference rules then combine their premises' dependency sets to form the conclusion's premise set, usually by taking the set union.

Another useful device is an explicit notation for proof goals. We write *Show P* when we adopt an interest in proving *P*, either because it is given as the overall proof goal, or as the result of reasoning backwards from the proof goal. Both devices are illustrated by the following simple abductive proof, where the conclusion is allowed to depend on a consistent set of ground instances of the abducible $\neg Ab(x)$:

$\{1\}$	1	$Bird(x) \land \neg Ab(x) \rightarrow Flies(x)$	
$\{2\}$	2	$Bird(tweety)$	
$\{\}$	3	$Show\ Flies(tweety)$	
$\{\}$	4	$Show\ \neg Ab(tweety)$	$1,2,3$
$\{5\}$	5	$\neg Ab(tweety)$	4
$\{1,2,5\}$	6	$Flies(tweety)$	$1,2,5$

3 Predicate Completion

TAL's syntactic characterization in terms of filtered circumscription produces a second-order theory to which regular theorem provers are not applicable. Fortunately, by placing certain syntactic restrictions on the TAL formulas one can ensure that the second-order circumscription formula can be compiled into an equivalent first-order characterization [Doherty, 1996]. The transformation is equivalent to Clarke's *predicate completion* [1978].

To see how it works, consider the Yale Shooting Problem formulated in TAL. There is an action for loading the gun, an action for firing the gun and killing Fred the turkey just in case the gun was loaded, the observation that Fred is initially alive, and a narrative consisting of the load and fire actions with a small wait in between. Note how the actions explicitly release the affected fluents from persistence by occluding them:

$$\begin{aligned}
&Occurs(t_1, t_2, load) \rightarrow \\
&\quad Occlude(t_2, loaded) \land Holds(t_2, loaded, true) \\
&Occurs(t_1, t_2, fire) \rightarrow \\
&\quad Holds(t_1, loaded, true) \rightarrow \\
&\qquad Occlude(t_2, loaded) \land Holds(t_2, loaded, false) \land \\
&\qquad Occlude(t_2, alive) \land Holds(t_2, alive, false) \\
&Holds(0, alive, true) \\
&Occurs(1, 2, load) \\
&Occurs(3, 4, fire)
\end{aligned}$$

Without saying anything about when fluents are *not* occluded, the above formulas do not predict the value of *alive* at time points other than 0, even if we add the persistence formulas T_{per}. We must first perform the predicate completion transformation step, minimizing fluent change by extending the above theory with additional formulas that correspond to the circumscription of *Occlude* and *Occurs*:

$$
\begin{aligned}
&\mathit{Occlude}(t,f) \leftrightarrow\\
&\quad f = \mathit{loaded} \land\\
&\quad \exists t_1[\mathit{Occurs}(t_1,t,\mathit{load}) \lor\\
&\qquad \mathit{Occurs}(t_1,t,\mathit{fire}) \land \mathit{Holds}(t_1,\mathit{loaded},\mathit{true})] \lor\\
&\quad f = \mathit{alive} \land\\
&\quad \exists t_1[\mathit{Occurs}(t_1,t,\mathit{fire}) \land \mathit{Holds}(t_1,\mathit{loaded},\mathit{true})]\\
&\mathit{Occurs}(t_1,t_2,\mathit{action}) \leftrightarrow\\
&\quad t_1 = 1 \land t_2 = 2 \land a = \mathit{load} \lor\\
&\quad t_1 = 3 \land t_2 = 4 \land a = \mathit{fire}
\end{aligned}
$$

The new theory makes it possible to derive non-occlusion deductively. Adding the T_{per} filter results in the intended consequences, e.g. $\neg \mathit{Holds}(4,\mathit{alive},\mathit{false})$.

This transformation works well for research benchmark problems. But the methodology has undesirable properties from the point of view of scalability. The transformation is applied to most of the entire knowledge base and is invalidated whenever some parts of the knowledge base change. A logical agent in a dynamic environment could have even simple queries stymied by potentially expensive computations.

Moreover, while the theorem prover can be used to reason about the consequences of *given* actions, it is not directly applicable to the more fundamental problem of reasoning about which actions to do in the first place. Even considering whether to do an action would require adding that action occurrence to the theory and repeating the transformation.

TALplanner [Kvarnström, 2005] avoids this problem by using special purpose planning algorithms to generate action occurrences. TAL is still used as a semantics for the finished plans, but the planning *process* is metatheoretical. In contrast, the next section will introduce an alternative abductive inference mechanism that naturally extends to planning, resulting in a unified planning and reasoning system without the need for a special purpose planning algorithm.

4 Abduction and Filtered Circumscription

Ginsberg [1989] presents a *circumscriptive theorem prover* (CTP) with properties conducive to scalability. The algorithm makes it possible to compute the logical consequences of a circumscribed theory without constructing the second-order circumscription axiom or compiling the theory beforehand. Of course, since circumscription is not even semi-decidable in the general case, some restrictions apply:

1. All formulas are universal, i.e. all its axioms can be written in the form $\forall \vec{x}\, P(\vec{x})$ where P is quantifier free.
2. The theory includes unique names and domain closure axioms.
3. The circumscription policy does not fix predicates.
4. The entire theory is circumscribed.

In the rest of the paper we assume that the theories we are interested in satisfy Restrictions 1 and 2, including only finitely many objects and time points.

Restriction 3 is not satisfied by TAL's circumscription policy as defined in Section 2.1. This, however, is not as troublesome as it might seem. As de Kleer and Konolige have shown [1989], any predicate P can be fixed by simultaneously minimizing both P and $\neg P$. Along with their proof they provide the intuition that this works since any attempt to make P smaller will automatically make $\neg P$ larger, and vice versa. In the end, therefore, the extension of P remains fixed. Using this equivalence we can eliminate the fixation of the *Holds* predicate from TAL's circumscription policy:

$$\mathit{Circ}(T;\mathit{Occlude},\mathit{Occurs},\mathit{Holds},\neg \mathit{Holds}) \land T_{\text{per}}$$

Unfortunately, Restriction 4 is not as easily remedied. The formulas belonging to T_{per} were kept outside the scope of the circumscription for a reason. They were not to affect the minimization of *Occlude* and *Occurs*, while still acting as a filter to remove models in which fluents change without being occluded.

4.1 Filtered Circumscription

In order to extend Ginsberg's method to a filtered circumscriptive theorem prover (FCTP) we first note that we can simplify the formulas to which it is applied.

Lemma 1. *Regular theorem proving can be used to reserve the FCTP for proving negative literals of minimized predicates, without loss of generality.*

Proof. Let T denote a theory and M the set of predicates to be minimized. According to Restriction 1 above, any proof goal G must be universal. If the theory is first put in negation normal form, the following serves as an example of a set of logical equivalences that can be used to reduce the filtered circumscriptive proof goal G to a literal:

$$F,\mathit{Circ}(T;M) \models \forall x P(x) \quad\Leftrightarrow\quad F,\mathit{Circ}(T;M) \models P(c)$$

Where c does not occur in $P(x)$ nor any premise that $P(c)$ depends on.

$$F,\mathit{Circ}(T;M) \models P \leftrightarrow Q \quad\Leftrightarrow\quad \begin{cases} F,\mathit{Circ}(T;M) \models P \rightarrow Q\\ F,\mathit{Circ}(T;M) \models Q \rightarrow P\end{cases}$$

$$F,\mathit{Circ}(T;M) \models P \land Q \quad\Leftrightarrow\quad \begin{cases} F,\mathit{Circ}(T;M) \models P\\ F,\mathit{Circ}(T;M) \models Q\end{cases}$$

$$F,\mathit{Circ}(T;M) \models P \lor Q \quad\Leftrightarrow\quad F,\mathit{Circ}(T;M) \models \neg P \rightarrow Q$$

$$F,\mathit{Circ}(T;M) \models P \rightarrow Q \quad\Leftrightarrow\quad F,\mathit{Circ}(T;M),P \models Q$$

The only remaining case is when G is a literal $(\neg)P$. Proposition 12 in [Lifschitz, 1994] tells us that if $(\neg)P$ is positive w.r.t. M (or is not one of the predicates in M), then $\mathit{Circ}(T;M) \models (\neg)P$ iff $T \models (\neg)P$, in which case we can continue using regular first-order theorem proving. Thus we need only resort to the FCTP when trying to prove literals that are negations of minimized predicates. □

While Ginsberg's implementation is based on an assumption-based truth maintenance system [de Kleer, 1986], the CTP algorithm can now be formulated in terms of abduction. Let T denote our theory, M be the set of predicates to be minimized, and the goal formula G be the negation of a predicate in M. The CTP then corresponds to the following algorithm [Brewka *et al.*, 1997]:

1. Let the set of abducibles be negations of predicates in M.
2. Abduce an explanation E for the goal G.
3. Check that there is no counter-explanation of E, i.e. that there is no explanation CE for $\neg E$.

This can be reformulated in terms of an inference rule. Explanations E and counter-explanations CE are conjunctions of ground abducible literals from Step 1 of the abductive algorithm. (Since they are the result of abductive proof, we always require them to be consistent with the theory T.) Step 2 of the algorithm is represented by the rule's premise, Step 3 by the rule's qualification, and the fact that the algorithm computes circumscription is stated by the rule's conclusion:

$$\frac{T, E \models G}{Circ(T;M) \models G}\ \text{CTP}$$

Where there is no CE consistent with T such that T,CE ⊨ ¬E.

Ginsberg [1989] shows this sound and complete for circumscription. Furthermore, as we prove next, the only explanation we need is the goal itself.

Lemma 2. *When G is the negation of a predicate in M, the CTP can use E = G without loss of generality.*

Proof. Constraining the set of explanations E does not affect soundness. Let us consider completeness. Using a stronger explanation than $E = G$ would gain us nothing since it can only decrease the applicability of the CTP. Assume a weaker explanation $T, E' \models G$. Then $T \models E' \rightarrow G$ and (because $G = E$) $T \models E' \rightarrow E$. Since E' is weaker than E, we know $E \rightarrow E'$, and consequently $T \models E \leftrightarrow E'$. □

We want to extend this to filtered circumscription, which adds a *filter* formula F. Since the filter is outside the scope of circumscription, it should not invalidate any conclusion drawn from the original theory by the inference rule above. However, it might allow us to draw new conclusions. As an intermediate step, we reformulate the CTP so that any counter-explanations consistent with T are listed explicitly in the conclusion.

Lemma 3. *The following inference rule is sound and complete for circumscription:*

$$\frac{\begin{array}{l} T, E \models G \\ T, CE_1 \models \neg E \\ \vdots \\ T, CE_n \models \neg E \end{array}}{Circ(T;M) \models \neg CE_1 \wedge \cdots \wedge \neg CE_n \rightarrow G}$$

Where $CE_1, \ldots, CE_n$ are all minimal counter-explanations consistent with T.

Proof. Completeness follows since if the CTP can be used to prove G, there are no consistent counter-explanations, and the implication antecedent collapses to true. To prove soundness, assume that we can use the above rule to prove G. It must be the case that $Circ(T;M) \models \neg CE_1 \wedge \cdots \wedge \neg CE_n$. Since each $\neg CE_i$ is a disjunction of minimized predicates, and circumscription never makes the extension of a minimized predicate larger, we have $T \models \neg CE_i$. But the rule assumes every CE_i consistent with T. This is only possible if $n = 0$, in which case G follows from the original CTP.

Note that it suffices to consider minimal counter-explanations. Suppose that $CE_i \subset CE'_i$. If we can prove $\neg CE_i$, then we can also prove the weaker condition $\neg CE'_i$.

□

The new rule makes it clear that when there *are* counter-explanations consistent with T, these could become inconsistent after adding the filter F, and the implication used to conclude G after all. A naïve implementation could simply add all (finitely many) implications produced by the above proof rule to T, creating a first-order equivalent[1] of $Circ(T;M)$, and append F. By running a sound and complete theorem prover one could derive G using Modus Ponens on the implications whose antecedent counter-explanations are inconsistent with this new filtered circumscription $F, Circ(T;M)$:

$$\frac{\begin{array}{l} Circ(T;M) \models \neg CE_1 \wedge \cdots \wedge \neg CE_n \rightarrow G \\ F, Circ(T;M) \models \neg CE_1 \wedge \cdots \wedge \neg CE_n \end{array}}{F, Circ(T;M) \models G}\ \text{MP}$$

It would, however, be very inefficient to add all implications when we only care about those implications that are relevant in a proof of G. Instead, we can get exactly the same result by adding an FCTP inference rule that allows us to conclude G directly, whenever all counter-explanations consistent with T are inconsistent with the filtered circumscription $F, Circ(T;M)$:

$$\frac{\begin{array}{l} T, E \models G \\ T, CE_1 \models \neg E \\ \vdots \\ T, CE_n \models \neg E \\ F, Circ(T;M) \models \neg CE_1 \\ \vdots \\ F, Circ(T;M) \models \neg CE_n \end{array}}{F, Circ(T;M) \models G}\ \text{FCTP}$$

Where $CE_1, \ldots, CE_n$ are all minimal counter-explanations consistent with T.

Theorem 1. *The FCTP inference rule is sound and complete for filtered circumscription.*

[1]It is always possible to construct a first-order equivalent of $Circ(T;M)$ given Ginsberg's assumption of universal theories with unique names and domain closure axioms.

Proof. Each proof produced by the naïve algorithm corresponds to a proof using the FCTP rule, obtained by replacing applications of Modus Ponens on one of the added implications by an application of the FCTP rule. Likewise, any application of the FCTP rule can be replaced by an implication and an application of Modus Ponens of the naïve algorithm. □

5 Examples

Let us use some TAL reasoning problems to illustrate the use of the FCTP. The proofs are abbreviated compared to the output of the implementation described in Section 6. E.g., we use the same (incremental and interruptible) consistency checking mechanism as Poole's THEORIST [1991] but do not display these steps below. All of the examples refer to the following filter formula as F:

$$\neg Occlude(t+1,f) \rightarrow (Holds(t,f,v) \leftrightarrow Holds(t+1,f,v))$$

Consider first the simplest case of fluent persistency through F. The fluent *alive* should persist from 0 to 1:

$\{1\}$	1	$Holds(0,alive,true)$	
$\{\}$	2	$Show\ Holds(1,alive,true)$	

The only way to show this is to use the filter F's persistence axiom:

$\{\}$	3	$Show\ \neg Occlude(1,alive)$	$F,1,2$

While none of the given formulas entail non-occlusion, we can apply the FCTP. First, $\neg Occlude(1,alive)$ can be used as its own explanation:

$\{4\}$	4	$\neg Occlude(1,alive)$	3

Next, we must find all counter-explanations consistent with T, i.e. all abductive explanations of $Occlude(1,alive)$ using T:

$\{\}$	5	$Show\ Occlude(1,alive)$	3

Given the theory T consisting of Row 1, it is impossible to prove $Occlude(1,alive)$. Consequently there are no counter-explanations and the proof succeeds:

$\{1,4\}$	6	$Holds(1,alive,true)$	$F,1,4$

The following example illustrates how the simultaneous minimization of *Holds* and $\neg$*Holds* can provide counter-examples that prevent credulous conclusions in the case of incomplete information of *loaded* in the initial state:

$\{1\}$	1	$Holds(0,alive,true)$	
$\{2\}$	2	$Holds(0,loaded,true) \rightarrow Occlude(1,alive)$	
$\{\}$	3	$Show\ Holds(1,alive,true)$	
$\{\}$	4	$Show\ \neg Occlude(1,alive)$	$F,1,3$
$\{5\}$	5	$\neg Occlude(1,alive)$	4
$\{\}$	6	$Show\ Occlude(1,alive)$	4
$\{\}$	7	$Show\ Holds(0,loaded,true)$	2,6
$\{8\}$	8	$Holds(0,loaded,true)$	7
$\{2,8\}$	9	$Occlude(1,alive)$	2,8

Since any attempt to apply FCTP recursively to prove $Holds(0,loaded,false)$ fails, we have a consistent Row 8 that counter-explains Row 5, and the proof fails. In other words, since it is possible that the gun is loaded, it is not safe to rely on the persistence of alive.

Here is an example in which Ginsberg's CTP does not give the expected result due to TAL's filtered circumscription:

$\{1\}$	1	$Holds(0,alive,true)$	
$\{2\}$	2	$Holds(0,loaded,false)$	
$\{3\}$	3	$Holds(1,loaded,true) \rightarrow Occlude(2,alive)$	
$\{\}$	4	$Show\ Holds(2,alive,true)$	

The goal follows if *alive* is not occluded. The first invocation of FCTP comes up with the explanation for $\neg Occlude(2,alive)$ in Row 6, but also a potential counter-explanation for $Occlude(2,alive)$ consistent with T in Row 9:

$\{\}$	5	$Show\ \neg Occlude(2,alive)$	$F,1,4$
$\{6\}$	6	$\neg Occlude(2,alive)$	5
$\{\}$	7	$Show\ Occlude(2,alive)$	5
$\{\}$	8	$Show\ Holds(1,loaded,true)$	3,7
$\{9\}$	9	$Holds(1,loaded,true)$	8

A recursive invocation of FCTP attempts to disprove the counter-explanation by showing that its negation (where $\neg Holds(t,f,true) \leftrightarrow Holds(t,f,false)$) follows from F,$Circ(T;M)$:

$\{\}$	10	$Show\ Holds(1,loaded,false)$	8
$\{\}$	11	$Show\ \neg Occlude(1,loaded)$	$F,2,9$
$\{12\}$	12	$\neg Occlude(1,loaded)$	11
$\{\}$	13	$Show\ Occlude(1,loaded)$	11
$\{2,12\}$	14	$Holds(1,loaded,false)$	$F,2,12$

This proof succeeds in Row 14 since the explanation in Row 12 is not counter-explained. Row 9 is thus not consistent with the *filtered* circumscription and the original conclusion follows after all:

$\{\}$	15	$Show\ \neg Occlude(1,alive)$	$F,1,4$
$\{16\}$	16	$\neg Occlude(1,alive)$	15
$\{\}$	17	$Show\ Occlude(1,alive)$	15
$\{1,16\}$	18	$Holds(1,alive,true)$	$F,1,16$
$\{1,2,6,12,16\}$	19	$Holds(2,alive,true)$	$F,6,18$

Finally, let us consider a disjunctive example. Suppose that activating a lamp either causes a change to the bulb's lit state or its broken state:

$\{1\}$	1	$Occurs(t_1,t_2,activate) \rightarrow Occlude(t_2,lit) \vee Occlude(t_2,broken)$
$\{2\}$	2	$Occurs(0,1,activate)$

Predicate completion requires that the theory can be put in the form $\Phi(t,f) \rightarrow Occlude(t,f)$ where $\Phi(t,f)$ does not contain occurrences of *Occlude*. But this is not possible given the above disjunctive action effect. TAL's predicate completion has been applied to actions with non-deterministic effects, but never when the occlusion itself is non-deterministic. The FCTP, however, has no problems proving e.g. that one of the fluents will not be occluded:

$\{\}$	3	$\mathit{Show}\ \neg \mathit{Occlude}(1, \mathit{lit}) \vee \neg \mathit{Occlude}(1, \mathit{broken})$	
$\{4\}$	4	$\mathit{Occlude}(1, \mathit{lit})$	3
$\{4\}$	5	$\mathit{Show}\ \neg \mathit{Occlude}(1, \mathit{broken})$	3
$\{4,6\}$	6	$\neg \mathit{Occlude}(1, \mathit{broken})$	5
$\{4\}$	7	$\mathit{Show}\ \mathit{Occlude}(1, \mathit{broken})$	5

The equivalences in Section 4.1 reduce the problem using the assumption in Row 4 and the new goal in Row 5, which has an explanation in Row 6.

From the action and its occurrence in Row 1 and 2 we know that one of the fluents are occluded. A counter-explanation to Row 6 is therefore $\neg \mathit{Occlude}(1, \mathit{lit})$:

$\{1,2\}$	8	$\mathit{Occlude}(1, \mathit{lit}) \vee \mathit{Occlude}(1, \mathit{broken})$	$1,2,7$
$\{4\}$	9	$\mathit{Show}\ \neg \mathit{Occlude}(1, \mathit{lit})$	$7,8$

However, the assumption in Row 4 is not part of the abductive explanation. It was introduced by the previous goal reduction and is still in force when trying to prove the counter-explanation. Consequently, assuming $\neg \mathit{Occlude}(1, \mathit{lit})$ would be inconsistent, the only counter-explanation fails, and the conclusion follows:

$\{1,6\}$	10	$\neg \mathit{Occlude}(1, \mathit{lit}) \vee \neg \mathit{Occlude}(1, \mathit{broken})$	$3,6$

Given the same action specification and action occurrence, one of the fluents has to be occluded at time 1. Attempting to prove both not occluded fails:

$\{\}$	3	$\mathit{Show}\ \neg \mathit{Occlude}(1, \mathit{lit}) \wedge \neg \mathit{Occlude}(1, \mathit{broken})$	
$\{\}$	4	$\mathit{Show}\ \neg \mathit{Occlude}(1, \mathit{lit})$	3
$\{5\}$	5	$\neg \mathit{Occlude}(1, \mathit{lit})$	4
$\{\}$	6	$\mathit{Show}\ \mathit{Occlude}(1, \mathit{lit})$	4
$\{1,2\}$	7	$\mathit{Occlude}(1, \mathit{lit}) \vee \mathit{Occlude}(1, \mathit{broken})$	$1,2,6$
$\{\}$	8	$\mathit{Show}\ \neg \mathit{Occlude}(1, \mathit{broken})$	$6,7$
$\{9\}$	9	$\neg \mathit{Occlude}(1, \mathit{broken})$	8
$\{1,2,9\}$	10	$\mathit{Occlude}(1, \mathit{lit})$	$7,9$

This time $\neg \mathit{Occlude}(1, \mathit{broken})$ is a consistent counter-explanation to Row 5 since there is no way to prove that its negation follows. The proof of the conjunction fails since the proof of the first conjunct fails. The failure would repeat if trying to prove the second conjunct first.

6 Unified Planning and Reasoning

Let us turn now to the task of implementing a planning and reasoning system based on the theory presented above. A commonly used implementation tool is logic programming. Indeed, earlier work with TAL made planning and reasoning possible through a compilation of TAL formulas into Prolog programs [Magnusson, 2007]. Proofs were *deductive* and instantiated a plan variable, similarly to the instantiation of the situation term in deductive Situation Calculus planning. Other work extends Prolog's inference mechanism to *abduction* by means of a meta-interpreter. This has been the de facto standard in work on abductive planning in Event Calculus, e.g. in [Shanahan, 2000; Denecker *et al.*, 1992; Missiaen *et al.*, 1995].

6.1 Pattern-Directed Inference System

We have explored a different avenue with a theorem prover based on *natural deduction*, inspired by similar systems by Rips [1994], Pelletier [1998], and Pollock [2000]. This is an interesting alternative to the more common resolution method used by most theorem provers, including Prolog. A natural deduction prover works with the formulas of an agent's knowledge base in their "natural form" directly, rather than first compiling them into clause form. This fits perfectly with the algorithm in Section 4 that has already eliminated the need for a compilation step for nonmonotonic reasoning.

The system uses pattern-directed inference similar to Forbus and de Kleer's fast tiny rule engine [1993]. To see how this works let us look at the inference rules. Applicable rules are added to a queue. By controlling which rule application the prover selects next we can implement e.g. depth-first, breadth-first, or best-first search.

Rules are divided into forward and backward rules. Forward rules are triggered whenever possible. They are therefore designed to be convergent, so as not to generate new inferences forever. An example is the modus ponens rule, which concludes Q whenever both P and $P \rightarrow Q$ are present in the knowledge base. The results in this paper generalizes our previous work that relied on forward rules to implement an incomplete consistency check [Magnusson, 2007; Magnusson and Doherty, 2008a; 2008b]. By explicitly trying to counter-explain abductive assumptions we no longer have to rely on forward rules being strong enough to detect inconsistent assumptions.

Backward rules, in contrast, are used to search backwards from the current proof goal and thus exhibit goal direction. An example is the goal chaining rule, which adds *Show* P as a new goal whenever both *Show* Q and $P \rightarrow Q$ are present in the knowledge base.

Combining forward and backward rules results in a bidirectional search for proofs that is pattern-directed since the prover's current goals are explicitly represented (by *Show* formula "patterns") in the knowledge base. This further contributes to the incremental nature of the reasoner. Inference can be interrupted at any time and later resumed since the knowledge base keeps track of what the prover was about to do.

6.2 Abductive Planning

The same reasoning mechanism can be used for abductive planning. Instead of reasoning about the effects of a given narrative, we reason about what narrative would have the desired effects.

Given a domain, we define a planning goal G as a conjunction of ground *Holds* literals. A plan for G is then a set T_{occ} of ground atomic *Occurs* formulas such that:

$$T_{\mathrm{per}} \wedge \mathit{Circ}(T \wedge T_{\mathrm{occ}}; \mathit{Occlude}, \mathit{Occurs}, \mathit{Holds}, \neg \mathit{Holds}) \vDash G$$

where the left hand side is consistent.

To generate T_{occ} we simply add *Occurs* to the set of abducibles for the proof goal. (Note however that we do not add *Occurs* as an abducible to explanations and counter-explanations. Doing so would amount to planning to thwart

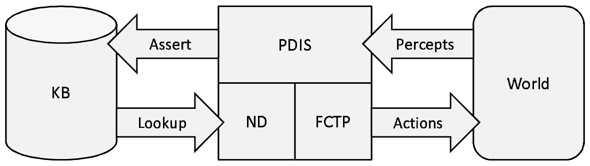


Figure 1: A logical agent architecture.

our own plans!) The soundness of this planning method follows directly from the soundness of the FCTP. Its implementation is already implicit in the theorem prover described above.

6.3 Logical Agents

Using the same reasoning mechanisms for many problems faced by autonomous agents results in a particularly simple agent architecture, as illustrated by Figure 1. The agent is equipped with a knowledge base containing formulas encoding knowledge about actions, world laws, and memory of recent events and percepts. The knowledge is used by the pattern-directed inference system (PDIS), with the help of natural deduction (ND) and the abductive algorithm from Section 4, to plan its actions.

But when plans meet the World they often fail. Executing a plan will achieve the goal only if the plan's assumptions hold up. The agent can detect some failures early through execution monitoring. In particular, persistence assumptions are represented in the plan by non-occlusion assumptions and can be continually evaluated. When a failure is perceived, that percept constitutes a counter-explanation to the assumption. Neither the assumption nor the planning goal derived from it are justified conclusions given the new percept. This immediately makes the pattern-directed goal-chaining inference rules applicable in trying to find an alternative proof of the goal. The result is an automatic plan revision and failure recovery process as the agent uses abductive planning to reestablish goals that lost their justification and execute an alternative plan.

7 Discussion

Predicate completion and the filtered circumscriptive theorem prover (FCTP) are two methods for automated reasoning in logics that use filtered circumscription. E.g., they both satisfy the circumscriptive characterization of TAL and produce the same end result, given the restrictions in Section 4. But in practical problem solving we believe the FCTP to have a number of desirable properties and advantages.

7.1 Abductive Planning and Reasoning

The FCTP uses abductive proof methods to reason with the original TAL formulas directly. As noted by Brewka, Dix, and Konolige [1997]:

> Abduction offers several benefits from a knowledge representation viewpoint. It does not require the assumption of complete knowledge of causation, and it is not necessary to assert the explanatory closures (which can lead to inconsistency and is computationally discouraging since it is performed globally on the theory).

Since the method does not presume complete knowledge of action occurrences, applications in planning open up. We exemplified this by using the reasoner for abductive planning. Furthermore, this unification of planning and reasoning forms the basis of a logical agent architecture that is highly capable despite its simplicity.

7.2 Reasoning Without Compilation

Predicate completion works by compiling the theory into a first-order equivalent with which reasoning proceeds. Lifschitz [1994] comments:

> But it should be observed that this approach to the automation of circumscription is not the only one possible. In fact, it may be unattractive, in view of the fact that it requires preprocessing the circumscription in a manner that is not related in any way to the goal formula.

The FCTP reasons with the TAL formulas directly, without first transforming the theory. Its efforts are spent only on those formulas of the knowledge base that are potentially relevant to the goal. The resulting proofs are also easier to comprehend since they refer directly to the formulas that were given to the system as input. Comprehension is also improved by the mechanism's similarity to argumentation and thereby to human reasoning. It would be interesting to further investigate the relation between FCTP and argumentation-theoretic systems.

7.3 Doubly Defeasible Reasoning

Finally, let us adopt a long term view and consider logical agents with commonsense knowledge. With the manual development or automated learning of very large knowledge bases, which are presumably needed for commonsense reasoning, it will be impractical or even impossible to search through all conceivable counter-explanations to a defeasible inference before taking action. It becomes necessary to consider what Pollock [2008] refers to as "doubly defeasible" reasoning. Not only can the reasoner change its mind with new information, it can also change its mind with more time to reason with its current information.

Predicate completion forces the agent to prove conclusions deductively in a first-order equivalent to the circumscribed theory. There is no way to interrupt the reasoning and act to the best of one's current knowledge. The incremental nature of the FCTP makes this possible. If counter-explanations are tested in the order of their likelyhood, it implements, in effect, an any-time algorithm that is always able to respond with the current best answer.

8 Conclusion

We are interested in building logical agents that use knowledge and reasoning to achieve goals. Observing that the world is both uncertain and dynamic motivates our choice of reasoning mechanisms that are incremental in nature. The computational effort of pondering a question should be related to the extent of relevant knowledge and the time available, not the total size of the knowledge base nor a potentially unbounded time requirement. Only then will the technology have the potential to scale up to very large knowledge bases.

One step in this direction is reported here in our investigation of Temporal Action Logic and its application in a logical agent architecture. By extending Ginsberg's circumscriptive theorem prover we have made it applicable to logics defined in terms of filtered circumscription. The abduction-based filtered circumscriptive theorem prover reasons directly with the input formulas, removing the need for a compilation step involving the entire knowledge base. Its interruptible nature enables an agent to act to the best of its knowledge given only limited time for reasoning. Finally, its double duty as an abductive planner makes possible a particularly simple agent architecture.

An agent architecture based exclusively on logical reasoning will necessarily suffer somewhat in efficiency compared to less general methods, despite being designed with scalability in mind. But achieving satisfactory performance in certain domains is already possible. E.g., we have applied the architecture to the control of computer game characters that require real-time interaction [Magnusson and Doherty, 2008b]. We believe computer games to be an excellent domain for empirical studies of logical agents on the road from tiny benchmark problems towards larger real world applications.

Acknowledgements

We thank Andrzej Szalas for helpful suggestions.

References

[Brewka *et al.*, 1997] Gerhard Brewka, Jürgen Dix, and Kurt Konolige. *Nonmonotonic Reasoning: An Overview*, volume 73 of *CSLI Lecture Notes*. CSLI Publications, 1997.

[Clark, 1978] Keith L. Clark. Negation as failure. In *Logic and Data Bases*, pages 293–322, 1978.

[de Kleer and Konolige, 1989] Johan de Kleer and Kurt Konolige. Eliminating the fixed predicates from a circumscription. *Artificial Intelligence*, 39(3):391–398, 1989.

[de Kleer, 1986] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2):127–162, 1986.

[Denecker *et al.*, 1992] Marc Denecker, Lode Missiaen, and Maurice Bruynooghe. Temporal reasoning with abductive event calculus. In *Proc. of ECAI'92*, pages 384–388, 1992.

[Doherty and Kvarnström, 2007] Patrick Doherty and Jonas Kvarnström. Temporal action logics. In *Handbook of Knowledge Representation*. Elsevier, 2007.

[Doherty and Lukaszewicz, 1994] Patrick Doherty and Witold Lukaszewicz. Circumscribing features and fluents: A fluent logic for reasoning about action and change. In *Proc. of ISMIS'94*, pages 521–530, 1994.

[Doherty, 1996] Patrick Doherty. PMON+: A fluent logic for action and change. Formal specification, version 1.0. Technical report, Linköping University, 1996.

[Forbus and de Kleer, 1993] Kenneth D. Forbus and Johan de Kleer. *Building Problem Solvers*. MIT Press, 1993.

[Ginsberg, 1989] Matthew L. Ginsberg. A circumscriptive theorem prover. In *Nonmonotonic reasoning*, pages 100–114. Springer, 1989.

[Kvarnström, 2005] Jonas Kvarnström. *TALplanner and Other Extensions to Temporal Action Logic*. PhD thesis, Linköping University, 2005.

[Lifschitz, 1994] Vladimir Lifschitz. Circumscription. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 298–352. Oxford University Press, New York, NY, USA, 1994.

[Magnusson and Doherty, 2008a] Martin Magnusson and Patrick Doherty. Deductive planning with inductive loops. In *Proc. of KR 2008*, pages 528–534, 2008.

[Magnusson and Doherty, 2008b] Martin Magnusson and Patrick Doherty. Logical agents for language and action. In *Proc. of AIIDE-08*, 2008.

[Magnusson, 2007] Martin Magnusson. *Deductive Planning and Composite Actions in Temporal Action Logic*. Licentiate thesis, Linköping University, 2007.

[Missiaen *et al.*, 1995] Lode Missiaen, Maurice Bruynooghe, and Marc Denecker. CHICA, an abductive planning system based on event calculus. *Journal of Logic and Computation*, 5(5):579–602, 1995.

[Pelletier, 1998] Francis Jeffry Pelletier. Natural deduction theorem proving in THINKER. *Studia Logica*, 60:3–43, 1998.

[Pelletier, 1999] Francis Jeffry Pelletier. A brief history of natural deduction. *History and Philosophy of Logic*, 20:1–31, 1999.

[Pollock, 2000] John Pollock. Rational cognition in OSCAR. In *Intelligent Agents VI: Agent Theories, Architectures, and Languages*, volume 1757 of *Lecture Notes in AI*, pages 71–90. Springer Verlag, 2000.

[Pollock, 2008] John L. Pollock. OSCAR: An architecture for generally intelligent agents. In *Proc. of AGI 2008*, pages 275–286. IOS Press, 2008.

[Poole, 1991] David Poole. Compiling a default reasoning system into prolog. *New Generation Computing*, 9(1):3–38, 1991.

[Rips, 1994] Lance J. Rips. *The psychology of proof: deductive reasoning in human thinking*. MIT Press, 1994.

[Sandewall, 1994] Erik Sandewall. *Features and Fluents: The Representation of Knowledge about Dynamical Systems*, volume 1. Oxford University Press, 1994.

[Shanahan, 1997] Murray Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. MIT Press, 1997.

[Shanahan, 2000] Murray Shanahan. An abductive event calculus planner. *The Journal of Logic Programming*, 44:207–239, 2000.

A Belief Revision Approach to Inconsistency Handling in Multi-Agent Systems

Luciano H. Tamargo **Alejandro J. García**
Marcelo A. Falappa **Guillermo R. Simari**
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Artificial Intelligence Research and Development Laboratory,
Department of Computer Science and Engineering - Universidad Nacional del Sur (UNS),
Bahía Blanca, ARGENTINA,
e-mail: {lt, ajg, maf, grs}@cs.uns.edu.ar

Abstract

In this paper, two methods that follow different attitudes when facing inconsistent knowledge will be analyzed. One is to use a revision operator over the knowledge base in order to preserve its consistency, the other is to cope with inconsistency. For the first case we will present a formalism for knowledge representation and consistency maintenance in a multi-agent system where deliberative agents can receive new information from others through communication. For the latter, an argumentative consequence in layered knowledge bases will be presented. Furthermore, we will compare these methods and we will show that they have several differences. However, we will show that they are equivalent under certain restrictions.

1 Introduction

One of the emerging important problems in the management of knowledge based systems is inconsistency handling. Inconsistency may be present for instance when there are exceptions to general rules, or there are several possibly disagreeing sources feeding the knowledge base. There are two attitudes that an agent can adopt when facing inconsistent knowledge. One is to revise the knowledge base with the main goal of preserving consistency, and the other is to cope with the inconsistency. In the first approach, the problem is that part of the information has to be thrown away and we no longer have access to it. Coping with inconsistency bypasses this difficulty.

As mentioned in [Benferhat *et al.*, 1993], the use of priorities among formulas is very important to appropriately revise inconsistent knowledge bases. In [Gärdenfors, 1988] it is shown that any revision process that satisfies natural requirements is implicitly based on such a set of priorities. Handling priorities has been shown to be completely in agreement with possibilistic logic (see [Dubois *et al.*, 1992] and [Benferhat *et al.*, 1992]).

In [Tamargo *et al.*, 2008], we have defined a revision operator which is based on a credibility ordering among agents. There, we address the problem of knowledge representation in a collaborative multi-agent system (MAS) where agents can obtain new information from others through communication. Informant agents are ranked by their credibility. This credibility order is used when new incoming information is contradictory with the current agent's belief base. We proposed a method for analyzing the information received; if inconsistency arises, the credibility order is used to decide which information prevails. The proposed operator satisfies the minimal change principle, and incoming information can be rejected when the agent has more credible beliefs that contradict the new information.

In [Benferhat *et al.*, 1993] several methods for coping with inconsistency are investigated by suitably defining notions of consequence capable of inferring non trivial conclusions from an inconsistent knowledge base. These consequence relations coincide with the classical definition when the knowledge base is consistent.

In this paper, the methods proposed in [Tamargo *et al.*, 2008] and in [Benferhat *et al.*, 1993] will be analyzed. It is clear that both methods follow different attitudes when facing inconsistent knowledge. The main difference between these approaches is that in [Benferhat *et al.*, 1993] inconsistency-tolerant consequence relations in layered knowledge bases are proposed, whereas in [Tamargo *et al.*, 2008] a revision operator is defined. In this work, we will show that the systems used in both approaches are equivalent under certain restrictions. That is, in [Tamargo *et al.*, 2008] a plausibility function is used to calculate the plausibility of a sentence, and in [Benferhat *et al.*, 1993] a procedure is proposed to determine if a sentence is an argumentative consequence of a knowledge base. We will show that the plausibility function and the procedure are equivalent.

As we will show in the next section, in [Tamargo *et al.*, 2008] agents' epistemic states will be represented by belief bases [Fuhrmann, 1991; Hansson, 1992]. The most widely studied method of changing a belief state is *partial meet contraction-revision*, also known as the AGM model [Alchourrón *et al.*, 1985]. The AGM model represents epistemic states by means of belief sets, that is, sets of sentences closed under logical consequence. However, our epistemic model uses belief bases, that is, sets of sentences not necessarily closed. As it will be explained below, we will adapt the notion of kernel contraction [Hansson, 1994; 1999] to our epistemic model, in which the beliefs are pro-

Partially supported by PIP-CONICET and UNS.

vided by agents.

It is important to note that the revision operator proposed in [Tamargo *et al.*, 2008] is similar to the revision operator proposed in [Benferhat *et al.*, 2002]. However, these operators are built in a different way. In [Benferhat *et al.*, 2002] the epistemic state is represented by a possibility distribution which is a mapping from the set of classical interpretations or worlds to the [0,1] interval. This distribution represents the degree of compatibility of the interpretations with the available information. In [Benferhat *et al.*, 2002] the revision is done over the possibility distribution. This revision modifies the ranking of interpretations so as to give priority to the input information. The input must be incorporated in the epistemic state; in other words, it takes priority over information in the epistemic state. They discuss the revision with respect to uncertain information; the input is of the form (ϕ, a), which means that the classical formula ϕ should be believed to a degree of certainty of exactly a.

Both approaches differ in some interesting ways. A first difference occurs in the way they handle the epistemic state. In [Benferhat *et al.*, 2002] the authors use belief sets, whereas in [Tamargo *et al.*, 2008] they use belief bases. The use of belief bases makes the representation of the agent's cognitive state more natural and computationally tractable. That is, following [Hansson, 1999], we consider that agents' beliefs could be represented by a limited number of sentences that correspond to the explicit beliefs of the agent. Another important difference, related to the intention of using the operator in a MAS environment, is the additional information that is added to each belief. Here, as in [Tamargo *et al.*, 2008], to decide whether to reject or accept a new belief, a comparison criterion among beliefs was defined. This criterion (called plausibility) is based on the credibility order among agents. We have assumed that this order is fixed; however, this order can be changed without affecting the behavior of the operator. This characteristic is one of the motivations for using agent identifiers instead of representing the plausibility of a sentence as a number as in [Benferhat *et al.*, 2002]. Furthermore, since in [Tamargo *et al.*, 2008] the revision process is based on the credibility order among agents, it is possible to define an operator to revise the credibility order. This will allow to represent changes over the credibility order. Moreover, in [Tamargo *et al.*, 2008] a total order among agents is necessary, but this assumption can be relaxed considering a partial order among agents.

A variety of notations have been adopted in the literature of Belief Revision (BR) in Multi-Agent Systems. In [Liu and Williams, 1999] an exhaustive analysis of these approaches is presented and a very interesting hierarchy is introduced (see Figure 1). Observe that in the hierarchy, *Multi-Agent Belief Revision* (MABR) and *Belief Revision using information from Multiple Sources* (MSBR) are distinguished.

As stated in [Liu and Williams, 1999], BR could be considered as part of the agent's skills to maintain the consistency of its own epistemic state. In this case, an individual BR process is carried out in a multi-agent environment, where the new information may come from multiple sources and maybe conflict. BR in this sense is called MSBR by [Dragoni *et al.*, 1994]. Cantwell [Cantwell, 1998] tries to resolve conflicting information by ordering the information sources on the basis of their trustworthiness. This could be served as a rational way of generating the new information credibility based on the source reliability using the terms of MSBR.

On the other hand, as discussed in [Liu and Williams, 1999], BR could also be used to achieve a society's or team's mutual belief goals ($e.g.$, reaching consensus before carrying out plans). In this setting, more than one agent takes part in the process. In order to pursue the mutual goal, the agents involved need to communicate, cooperate, coordinate, and negotiate with one another. A MABR system is a MAS whose mutual goal involves BR. Different formalisms have been presented to deal with MABR [Liu and Williams, 1999; 2001; Kfir-Dahav and Tennenholz, 1996].

MSBR studies individual agent revision behaviors, *i.e.*, when an agent receives information from multiple agents towards whom it has social opinions. MABR investigates the overall BR behavior of agent teams or a society. MSBR is one of the essential components of MABR. In our approach, as in [Tamargo *et al.*, 2008], we focus on MSBR.

The AGM paradigm [Alchourrón *et al.*, 1985] has been widely accepted as a standard framework for BR. But it is only capable of prescribing revision behaviors of a single agent. The BR process is more complex in the multi-agent case. Besides the principle of minimal change, there exist other requirements due to the sophisticated agent interactions.

An agent is capable of carrying out Individual Belief Revision (IBR), while an agent society or team is capable of MABR. IBR in a single agent environment (Single Belief Revision, SBR) could be achieved using classical BR satisfying or adapting AGM postulates. IBR in a multiple agent environment is MSBR, *i.e.*, a single agent will have to process information coming from more than one source.

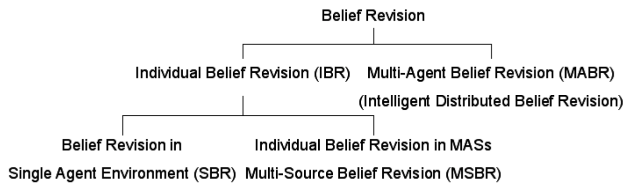

Belief Revision
Individual Belief Revision (IBR)
Multi-Agent Belief Revision (MABR)
(Intelligent Distributed Belief Revision)
Belief Revision in
Single Agent Environment (SBR)
Individual Belief Revision in MASs
Multi-Source Belief Revision (MSBR)

Figure 1: Belief Revision Hierarchy

The paper is organized as follows. The next section is a summary of plausible belief bases based on agent credibility. Section 3 contains a summary of argumentative inference in layered knowledge bases where layers express degrees of certainty as in possibilistic logic [Dubois *et al.*, 1992]. Then, in Section 4 we will show the equivalence between the systems proposed in [Benferhat *et al.*, 1993] and [Tamargo *et al.*, 2008]. In Section 5 a non-prioritized revision operator that uses the plausibility order is introduced. A discussion follows, and finally we offer some interesting conclusions.

2 Plausible Belief Bases Based on Agent Credibility

In this section we will show the epistemic model used in [Tamargo *et al.*, 2008]. Furthermore, we will describe the *plausibility function* (Definition 8) which uses a system that is equivalent to the one based on argumentative inference as defined in [Benferhat *et al.*, 1993]. Throughout the rest of this paper, a propositional language $\mathcal{L}$ with a complete set of boolean connectives will be adopted, namely $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$. Also, we assume the existence of an operator Cn that satisfies *inclusion* ($B \subseteq Cn(B)$), *iteration* ($Cn(B) = Cn(Cn(B))$), and *monotonicity* (if $B \subseteq C$ then $Cn(B) \subseteq Cn(C)$) and includes the classical consequence operator. In general, we will write $\alpha \in Cn(B)$ as $B \vdash \alpha$.

2.1 Epistemic Model

We consider a finite set of agent identifiers that is denoted as $\mathcal{A}gents$. Since agents can obtain information from other agents, agents' beliefs are represented as tuples (ϕ, A), where ϕ is a sentence of a propositional language $\mathcal{L}$ and $A \in \mathcal{A}gents$. Let $\mathcal{K} = 2^{\mathcal{L}\times\mathcal{A}gents}$, each agent $A \in \mathcal{A}gents$ has a consistent belief base $K_A \in \mathcal{K}$. As stated above, informant agents will be ranked by their credibilities. Hence, a Credibility Order over the set $\mathcal{A}gents$ is introduced:

Definition 1 *A* ***credibility order among agents****, denoted by '$\leq_{Co}$', is a total order over $\mathcal{A}gents$, where $A_1 \leq_{Co} A_2$ means that A_2 is at least as credible as A_1. The strict relation $A_1 <_{Co} A_2$, denoting that A_2 is strictly more credible than A_1, is defined as $A_1 \leq_{Co} A_2$ and $A_2 \not\leq_{Co} A_1$. Moreover, $A_1 =_{Co} A_2$ means that A_1 is as credible as A_2, and it holds when $A_1 \leq_{Co} A_2$ and $A_2 \leq_{Co} A_1$.*

Since this order is assumed to be total, the following properties hold for all $A_1, A_2, A_3 \in \mathcal{A}gents$:
– Totality or Completeness: $A_1 \leq_{Co} A_2$ or $A_2 \leq_{Co} A_1$.
– Transitivity: if $A_1 \leq_{Co} A_2$ and $A_2 \leq_{Co} A_3$ then $A_1 \leq_{Co} A_3$.
– Antisymmetry: if $A_1 \leq_{Co} A_2$ and $A_2 \leq_{Co} A_1$ then $A_1 =_{Co} A_2$.

Example 1 *Consider a set $\mathcal{A}gents = \{A_1, A_2, A_3, A_4\}$ where the credibility order is $A_1 \leq_{Co} A_2$, $A_2 =_{Co} A_3$, $A_3 \leq_{Co} A_4$. The belief base $K_{A_1} = \{(\psi, A_3), (\phi, A_2), (\omega, A_3), (\phi \rightarrow \psi, A_2), (\phi \rightarrow \psi, A_1), (\omega \rightarrow \psi, A_1), (\phi \rightarrow \varphi, A_2), (\varphi \rightarrow \psi, A_3), (\rho, A_1), (\omega \rightarrow \rho, A_2)\}$. Observe that K_{A_1} has two tuples with the sentence $\phi \rightarrow \psi$.*

Next, two auxiliary functions are introduced in order to obtain the set of sentences and the set of agents that belong to a belief base $K \in \mathcal{K}$.

Definition 2 *The* ***sentence function****, $Sen : \mathcal{K} \rightarrow 2^{\mathcal{L}}$, is a function such that for a given belief base $K \in \mathcal{K}$, $Sen(K) = \{\phi : (\phi, A) \in K$ for any $A \in \mathcal{A}gents\}$.*

Definition 3 *The* ***agent identifier function****, $Ag : \mathcal{K} \rightarrow 2^{\mathcal{A}gents}$, is a function such that for a given belief base $K \in \mathcal{K}$, $Ag(K) = \{A : (\phi, A) \in K$ for any $\phi \in \mathcal{L}\}$.*

We say that a belief base K is consistent if $Sen(K)$ is a consistent set. As stated above, agents can receive new beliefs from other informants. This new information can be contradictory with their current beliefs. For instance, consider again the belief base (K_{A_1}) of Example 1, where $Sen(K_{A_1}) \vdash \psi$ (observe that there are several derivations for ψ). Suppose now that agent A_1 receives the input $(\neg\psi, A_4)$. It is clear that adding $(\neg\psi, A_4)$ to K_{A_1} will produce an inconsistent belief base. Therefore, the agent has to revise its beliefs and decide whether it rejects $(\neg\psi, A_4)$ or it withdraws ψ. The credibility order will be used to decide which information prevails. However, since there can be several derivations of ψ, then we have to "cut" all of them. For doing that, all the minimal subsets of K_{A_1} that entail ψ will be obtained, using an extension of *Kernel contractions* [Hansson, 1994]. These are based on a selection among the sentences that are relevant to derive the sentence to be retracted. In order to perform a contraction, kernel contractions use incision functions which cut into the minimal subsets that entail the information to be given up. We will adapt the notion of kernel contraction to our epistemic model. First, we will define kernel set and then we will present incision functions that cut beliefs according to their plausibility.

Definition 4 *Let $K \in \mathcal{K}$ and $\phi \in \mathcal{L}$. Then H is a ϕ-kernel of K if and only if* 1) *$H \subseteq K$;* 2) *$Sen(H) \vdash \phi$;* 3) *if $H' \subset H$, then $Sen(H') \nvdash \phi$.*

Note that a *ϕ-kernel* is a minimal set of tuples from K that entails ϕ. The set of ϕ-$kernels$ of K is denoted $K^{\perp\!\!\perp}\phi$ and is called *kernel set*.

The information (ϕ, A_p) that an agent A_i receives from A_p is consistent with its current belief base K_{A_i} if $Sen(K_{A_i}) \nvdash \neg\phi$. If $Sen(K_{A_i}) \nvdash \neg\phi$, then it is clear that (ϕ, A_p) is added to K_{A_i}. If $Sen(K_{A_i}) \vdash \phi$ then (ϕ, A_p) is also added to K_{A_i} because the plausibility of ϕ may be increased (see Section "Sentence Plausibility" below). Therefore, a belief base $K \in \mathcal{K}$ may contain the same belief in two tuples with different agent identifiers. For instance, in Example 1 the sentence $\phi \rightarrow \psi$ is in two tuples. From the tuples point of view there is no redundancy, due to the fact that each tuple represents a different informant.

In the next section we will show how our operator uses the additional information (agent identifiers) in order to guide the revision process. The plausibility of the sentences will be defined by using the agent identifiers stored in the belief bases of the agents and the credibility order among agents.

2.2 Sentence Plausibility

We will use the agent identifier to compute the plausibility of the beliefs, *i.e.*, each of the agent's beliefs will have an associated plausibility that will depend on the agent identifier and the *credibility order among agents*. The behavior of the plausibility is similar to that of epistemic entrenchment defined in [Gärdenfors and Makinson, 1988]. That is, if ϕ and ψ are sentences in $\mathcal{L}$, the notation $\phi \preceq_{K_A} \psi$ means "ψ is at least as plausible as ϕ relative to the belief base K of agent A".

A belief base $K \in \mathcal{K}$ may support either explicit or entailed sentences. The explicit sentences are those contained in $Sen(K)$, while the entailed sentences are those that are not in $Sen(K)$ but they are entailed by sentences in $Sen(K)$. In order to obtain the entailed sentences from a belief base K we will use the following function:

Definition 5 *The* ***belief function****, $Bel : \mathcal{K} \rightarrow 2^{\mathcal{L}}$, is a function such that for a given belief base $K \in \mathcal{K}$, $Bel(K) = \{\phi : \phi \in \mathcal{L}$ and $Sen(K) \vdash \phi\}$.*

In general, there may be several proofs for ψ from K. Therefore, to calculate the plausibility of a sentence (ψ) we must analyze all its proofs. In order to do this, the kernel sets are used. We consider that this calculation should be cautious, that is, from each ψ-kernel, we want to obtain the lesser-plausibility tuples. This gives us the plausibility of each proof. Then, the plausibility of a derived sentence ψ will be the greater plausibility among those of each ψ-kernel. In order to define this, two functions will be given next.

Definition 6 *The **lesser-credibility sources function**, $min : \mathcal{K} \rightarrow 2^{\mathcal{K}}$, is a function such that for a given belief base $K \in \mathcal{K}$, $min(K) = \{(\phi, A_i) : (\phi, A_i) \in K$ and for all $(\varphi, A_j) \in K, A_i \leq_{Co} A_j\}$.*

Definition 7 *The **greater-credibility sources function**, $max : \mathcal{K} \rightarrow 2^{\mathcal{K}}$, is a function such that for a given belief base $K \in \mathcal{K}$, $max(K) = \{(\phi, A_i) : (\phi, A_i) \in K$ and for all $(\varphi, A_j) \in K, A_j \leq_{Co} A_i\}$.*

Example 2 *Consider $\mathcal{A}gents = \{A_1, A_2, A_3\}$ where $A_1 \leq_{Co} A_2 \leq_{Co} A_3$. Let $K_{A_1} = \{(\phi, A_1), (\phi, A_2), (\psi, A_1), (\rho, A_1), (\phi \rightarrow \rho, A_3)\}$ be the belief base of A_1. Then, $Bel(K_{A_1})$ will contain the sentences from $Sen(K_{A_1})$ plus the sentences derived by "$\vdash$". For instance, $\phi, \psi, \rho, \phi \vee \psi, \phi \vee \psi \vee \rho, \ldots$, and so on. Then, $min(K_{A_1}) = \{(\phi, A_1), (\psi, A_1), (\rho, A_1)\}$ and $max(K_{A_1}) = \{(\phi \rightarrow \rho, A_3)\}$.*

Next, we will introduce a function that returns the plausibility of a sentence that can be explicitly in K or entailed from K.

Definition 8 *The **Plausibility function**, $Pl : \mathcal{L} \times \mathcal{K} \rightarrow \mathcal{A}gents$, is a function such that for a $\phi \in \mathcal{L}$ and a belief base $K \in \mathcal{K}$, $Pl(\phi, K) = Ag(max(\bigcup_{X \in K^{\perp\!\!\perp}\phi} min(X)))$.*

Note that if $(\rho, A_1) \in K_{A_1}$ then $Pl(\rho, K_{A_1})$ could be different from A_1. For instance, considering Example 2, we have $Pl(\phi, K_{A_1}) = A_2$, $Pl(\psi, K_{A_1}) = A_1$ and $Pl(\rho, K_{A_1}) = A_2$. In Example 4 we will describe (step by step) how the *plausibility function* returns an agent identifier.

Definition 9 (Plausibility Criterion) *Let $K_A \in \mathcal{K}$ be the belief base of agent A and let $\{\phi, \psi\} \subseteq Bel(K_A)$, then $\phi \preceq_{K_A} \psi$ if and only if $Pl(\phi, K_A) \leq_{Co} Pl(\psi, K_A)$.*

The strict relation $\phi \prec_{K_A} \psi$, representing "ψ is more plausible than ϕ", is defined as "$\phi \preceq_{K_A} \psi$ and $\psi \not\preceq_{K_A} \phi$". Moreover, $\phi \simeq_{K_A} \psi$ means that ϕ is as plausible as ψ, and it holds when $\phi \preceq_{K_A} \psi$ and $\psi \preceq_{K_A} \phi$. From the previous definition we can observe that the plausibility of the sentences inherits the properties of the *credibility order among agents* ('$\preceq_{K_A}$' is a total order on $\mathcal{L}$). Furthermore, note that the relation '$\preceq_{K_A}$' is only defined with respect to a given K_A (different belief bases may be associated with different orderings of plausibility as shown in Example 3).

Example 3 *Consider a set $\mathcal{A}gents = \{A_1, A_2, A_3\}$ where the credibility order is $A_1 \leq_{Co} A_2, A_2 \leq_{Co} A_3$. Suppose that agent A_2 has the following belief base $K_{A_2} = \{(\phi, A_1), (\psi, A_2), (\rho, A_3)\}$, and suppose that agent A_3 has the following belief base $K_{A_3} = \{(\phi, A_1), (\psi, A_3), (\rho, A_2)\}$. Then, for both agents, ψ is more plausible than ϕ (i.e., $\phi \preceq_{K_{A_2}} \psi$ and $\phi \preceq_{K_{A_3}} \psi$). However, for A_2, ρ is more plausible than ψ ($\psi \preceq_{K_{A_2}} \rho$) whereas for A_3, ψ is more plausible than ρ ($\rho \preceq_{K_{A_3}} \psi$).*

Example 4 *Consider a set $\mathcal{A}gents = \{A_1, A_2, A_3\}$ where the credibility order is $A_3 \leq_{Co} A_2$, $A_2 \leq_{Co} A_1$. The belief base of agent A_1 is $K_{A_1} = \{(\psi, A_3), (\phi, A_2), (\phi \rightarrow \psi, A_2), (\phi \rightarrow \psi, A_1), (\omega, A_3), (\omega \rightarrow \psi, A_1), (\phi \rightarrow \varphi, A_2), (\varphi \rightarrow \psi, A_3), (\rho, A_1), (\omega \rightarrow \rho, A_2)\}$. Suppose that agent A_1 needs to calculate the plausibility of ψ. In order to do so, A_1 will perform the following steps.*

• Step 1. *Obtain the minimal subsets that derive ψ from belief base K_{A_1}.*
$K_{A_1}^{\perp\!\!\perp}\psi = \{H_a, H_b, H_c, H_d, H_e\}$ where $H_a = \{(\psi, A_3)\}$, $H_b = \{(\phi, A_2), (\phi \rightarrow \psi, A_2)\}$, $H_c = \{(\phi, A_2), (\phi \rightarrow \psi, A_1)\}$, $H_d = \{(\omega, A_3), (\omega \rightarrow \psi, A_1)\}$ and $H_e = \{(\phi, A_2), (\phi \rightarrow \varphi, A_2), (\varphi \rightarrow \psi, A_3)\}$.

• Step 2. *Obtain from each ψ-kernel $\in K_{A_1}^{\perp\!\!\perp}\psi$ the set containing the lesser plausibility tuples determined by the lesser-credibility sources function:*
$min(H_a) = \{(\psi, A_3)\}$, $min(H_b) = \{(\phi, A_2), (\phi \rightarrow \psi, A_2)\}$, $min(H_c) = \{(\phi, A_2)\}$, $min(H_d) = \{(\omega, A_3)\}$ and $min(H_e) = \{(\varphi \rightarrow \psi, A_3)\}$.

• Step 3. *Obtain from the tuples of the previous step the set containing the greater plausibility tuples determined by the greater-credibility sources function: $max(\{(\psi, A_3), (\phi, A_2), (\phi \rightarrow \psi, A_2), (\omega, A_3), (\varphi \rightarrow \psi, A_3)\}) = \{(\phi, A_2), (\phi \rightarrow \psi, A_2)\}$.*

• Step 4. *Obtain from the tuples of the previous step the set containing the identifiers determined by the get agent identifier function: $Ag(\{(\phi, A_2), (\phi \rightarrow \psi, A_2)\}) = \{A_2\}$.*

Therefore, $Pl(\psi, K_{A_1}) = A_2$. Hence, when ψ is compared with other beliefs, A_2 will be used as the informant of ψ (the plausibility of ψ will be given by A_2).

3 Arguments in Prioritized Knowledge Bases

In this section we will summarize an argumentative consequence relation proposed in [Benferhat *et al.*, 1993]. Furthermore, we will show the procedure which is used to determine if a sentence is an argumentative consequence of a knowledge base. Next, in Section 4 we will show that this procedure is equivalent to the one discussed in the previous section.

In the prioritized case proposed in [Benferhat *et al.*, 1993], a knowledge base can be viewed as a layered knowledge base $\Sigma = B_1 \cup \ldots \cup B_n$, such that formulas in B_i have the same level of priority or certainty and are more reliable than the ones in B_j where $j > i$. This stratification is modeled by attaching a weight $\alpha \in [0, 1]$ to each formula with the convention that $(\phi\ \alpha_i) \in B_i, \forall i$ and $\alpha_1 = 1 > \alpha_2 > \alpha_n > 0$.

A sub-base $\Sigma_i = E_1 \cup \ldots \cup E_n$ of $\Sigma = B_1 \cup \ldots \cup B_n$ where $E_j \subseteq B_j$, $j = 1, \ldots, n$, is said to be consistent if $\Sigma_i \nvdash \perp$, and is said to be maximally consistent if adding any formula from $(\Sigma - \Sigma_i)$ to Σ_i produces an inconsistent knowledge base. Before introducing the notion of argumentation in prioritized knowledge bases, the notion of entailment in a layered base is defined.

Definition 10 *Let $\Sigma = B_1 \cup \ldots \cup B_n$ be a layered knowledge base. A formula ϕ is said to be a **π-consequence** of Σ with weight α_i, denoted by $\Sigma \vdash_\pi (\phi\ \alpha_i)$, if and only if $B_1 \cup \ldots \cup B_i$ is consistent, $B_1 \cup \ldots \cup B_i \vdash \phi$, and $\forall j < i, B_1 \cup \ldots \cup B_j \nvdash \phi$.*

In a stratified base, a consistent sub-base of Σ (in general not maximal), denoted by $\pi(\Sigma)$, is induced by the levels of priority and defined in this way: $\pi(\Sigma) = B_1 \cup \ldots \cup B_i$, such that $\pi(\Sigma)$ is consistent and $B_1 \cup \ldots \cup B_{i+1}$ is inconsistent.

The remaining sub-base $\Sigma - \pi(\Sigma)$ is simply inhibited. Then, in [Benferhat *et al.*, 1993] an extension of the argumentative inference to layered knowledge bases is proposed.

Definition 11 *A sub-base Σ_i of Σ is said to be **an argument for a formula** ϕ with a weight α if it satisfies the following conditions:* 1) $\Sigma_i \nvdash \perp$, 2) $\Sigma_i \vdash_\pi (\phi\ \alpha)$, *and* 3) $\forall(\psi\ \beta) \in \Sigma_i$, $\Sigma_i - \{(\psi\ \beta)\} \nvdash_\pi (\phi\ \alpha)$.

Definition 12 *A formula ϕ is said to be an **argumentative consequence** of Σ, denoted by $\Sigma \vdash_{\mathcal{A}} (\phi\ \alpha)$, if and only if there exists an argument for $(\phi\ \alpha)$ in Σ, and for each argument of $(\neg\phi\ \beta)$ in Σ, we have $\beta < \alpha$.*

Then, in [Benferhat *et al.*, 1993] a procedure is given which determines if ϕ is an argumentative consequence of a stratified knowledge base $\Sigma = B_1 \cup \ldots \cup B_n$. The procedure assumes the existence of an algorithm which checks if there exists an argument for a given formula in some base.

The procedure is based on the construction of the maximal argument of ϕ and its contradiction. First the procedure starts with the sub-base B_1, and it checks if there is a consistent sub-base of B_1 which entails ϕ or $\neg\phi$. If the response is respectively Yes-No then ϕ is an argumentative consequence of Σ with a weight $\alpha_1 = 1$; by symmetry, if the response is No-Yes then $\neg\phi$ is in this case the argumentative consequence of Σ. Now, if the response is Yes-Yes, then we have a conflict. If the answer corresponds to one of the answers given above then the algorithm stops.

In the last case, if the response is No-No we repeat the same cycle described above with $B_1 \cup B_2$, and so on. The algorithm stops when all the base Σ is checked.

Example 5 *Let Σ be the following knowledge base:* $\Sigma = \{(\psi\ 0.4), (\phi\ 0.7), (\phi{\rightarrow}\psi\ 0.7), (\phi \rightarrow \psi\ 0.9), (\omega\ 0.4), (\omega{\rightarrow}\psi\ 0.9), (\phi{\rightarrow}\varphi\ 0.7), (\varphi{\rightarrow}\psi\ 0.4), (\gamma\ 0.9), (\omega{\rightarrow}\ \gamma\ 0.7)\}$. *In the stratified fashion, the knowledge base Σ can be viewed as follows:* $\Sigma = B_1 \cup B_2 \cup B_3$ *where,* $B_1 = \{(\phi \rightarrow \psi, 0.9), (\omega \rightarrow \psi, 0.9), (\gamma, 0.9)\}$, $B_2 = \{(\phi, 0.7), (\phi \rightarrow \psi, 0.7), (\phi \rightarrow \varphi, 0.7), (\omega \rightarrow \gamma, 0.7)\}$ *and* $B_3 = \{(\psi, 0.4), (\omega, 0.4), (\varphi \rightarrow \psi, 0.4)\}$. *Since $B_1 \nvdash \psi$ and $B_1 \cup B_2 \vdash \psi$ then, the weight of ψ is 0.7.*

4 Equivalence Between the Two Methods

In [Tamargo *et al.*, 2008] we have presented a formalism for knowledge representation and consistency maintenance in a MAS where deliberative agents can receive new information from other agents, *i.e.*, we adopt Multi-Source Belief Revision (MSBR). Recall that since agents can obtain information from other agents, agents' beliefs are represented as tuples (ϕ, A_i), where ϕ is a sentence of a propositional language $\mathcal{L}$ and A_i is an agent identifier from $\mathcal{A}gents$. Each agent $A \in \mathcal{A}gents$ will have a belief base $\Sigma^{MS}_A \in 2^{\mathcal{L}\times\mathcal{A}gents}$. For instance, $\Sigma^{MS}_{A_1} = \{(\phi, A_1), (\psi, A_2), (\varphi, A_3)\}$. Informant agents will be ranked and this rank represents the credibility of agents. Observe that an agent belief base Σ^{MS}_A can be viewed as a layered knowledge base $\Sigma^{MS}_A = B_1 \cup \ldots \cup B_n$, such that formulas in B_i are attached with agent identifiers to the same level of credibility and are more plausible than the ones in B_j where $j > i$.

In [Benferhat *et al.*, 1993], a knowledge base can be viewed as a **layered knowledge base** $\Sigma = B_1 \cup \ldots \cup B_n$, such that formulas in B_i have the same level of priority or certainty and are more reliable than the ones in B_j where $j > i$. This stratification is modeled by attaching a weight $\alpha \in [0, 1]$ to each formula with the convention that $(\phi\ \alpha_i) \in B_i, \forall i$ and $\alpha_1 = 1 > \alpha_2 > \alpha_n > 0$. There, a formula ϕ is said to be an **argumentative consequence** of Σ, denoted by $\Sigma \vdash_{\mathcal{A}} (\phi\ \alpha)$, if and only if there exists an argument for $(\phi\ \alpha)$ in Σ, and for each argument of $(\neg\phi\ \beta)$ in Σ, we have $\beta < \alpha$.

Then, in [Benferhat *et al.*, 1993], a procedure which determines if ϕ is an argumentative consequence of a stratified knowledge base $\Sigma = B_1 \cup \ldots \cup B_n$ is proposed. From this procedure, if Σ is consistent, it is clear that $\Sigma \vdash_{\mathcal{A}} (\phi\ \alpha_i)$ if and only if $B_1 \cup \ldots \cup B_i \vdash \phi$ and $B_1 \cup \ldots \cup B_{i-1} \nvdash \phi$, where α_i is the weight of all formulas that belong to B_i. Note that only consistent belief bases are considered because in our approach a revision operator was defined, and therefore we do not allow inconsistency.

Considering the assumptions stated below, we will prove that: given a belief base Σ^{MS}_A such that $\Sigma^{MS}_A = B_1 \cup \ldots \cup B_n$, and a formula ϕ, the agent identifier returned by $Pl(\phi, \Sigma^{MS}_A)$ is A_i whenever $B_1 \cup \ldots \cup B_i \vdash \phi$ and $B_1 \cup \ldots \cup B_{i-1} \nvdash \phi$.

Assumption 1: The formulas' weights are represented by agent identifiers.

Assumption 2: Σ^{MS}_A is consistent (in our approach we do not allow inconsistency).

Assumption 3: There is no pair of agents A_i and A_j such that $A_i =_{Co} A_j$.

Assumption 4: $\forall\omega(\omega \in Sen(B_i)$ if and only if $(\omega, A_i) \in B_i)$. Hence, by the stratification of the knowledge base on MAS, $\forall i\ A_{i+1} \leq_{Co} A_i$.

Assumption 5: $\Sigma^{MS}_i = E_1 \cup \ldots \cup E_n$ is said to be a sub-base of $\Sigma^{MS} = B_1 \cup \ldots \cup B_n$ if $E_j \subseteq B_j, j = 1, \ldots, n$. Note that E_j may be $\emptyset$.

Theorem: Let $\Sigma^{MS}_{A_p} = B_1 \cup \ldots \cup B_n$ be a layered knowledge base on a MAS. Then, $Pl(\phi, \Sigma^{MS}_{A_p}) = A_i$ if and only if $B_1 \cup \ldots \cup B_i \vdash \phi$ and $B_1 \cup \ldots \cup B_{i-1} \nvdash \phi$.

Proof: For simplicity we will use K to represent $\Sigma^{MS}_{A_p}$.

To prove the statement of the theorem we must show the following:

(a) If $Pl(\phi, K) = A_i \Rightarrow B_1 \cup \ldots \cup B_i \vdash \phi$ and $B_1 \cup \ldots \cup B_{i-1} \nvdash \phi$

(b) If $B_1 \cup \ldots \cup B_i \vdash \phi$ and $B_1 \cup \ldots \cup B_{i-1} \nvdash \phi \Rightarrow Pl(\phi, K) = A_i$

a) Let $K = \Sigma^{MS}_{A_p} = B_1 \cup \ldots \cup B_n$. If $Pl(\phi, K) = A_i$, then by Definition 8, $A_i = Ag(max(\bigcup_{X\in K^{\perp\!\!\perp}\phi} min(X)))$ and by Definition 3 it holds that $\exists\ \omega \in Sen(K)$ such that $(\omega, A_i) \in max(\bigcup_{X\in K^{\perp\!\!\perp}\phi} min(X))$. Thus, by Definition 7, we note that $(\omega, A_i) \in \bigcup_{X\in K^{\perp\!\!\perp}\phi} min(X)$. Therefore, there exists $X_q \in K^{\perp\!\!\perp}\phi$ such that $(\omega, A_i) \in X_q$ and $Ag(min(X_q)) = A_i$. Since $X_q \subseteq K$, it follows by assumption 5 that $X_q = E_1 \cup \ldots \cup E_i$ such that $\forall j, 1 \leq j \leq i, E_j \subseteq B_j$. Note that E_i is the stratum less reliable by Definition 6. Furthermore, since X_q is a kernel, then $E_1 \cup \ldots \cup E_i \vdash \phi$. It is for that reason and by assumption 2, that $B_1 \cup \ldots \cup B_i \vdash \phi$. In addition, since $(\omega, A_i) \in B_i$ it follows from assumption 4 and the fact that $Sen(X_q \backslash \{(\omega, A_i)\}) \nvdash \phi$ (X_q is a *ϕ-kernel*), that $B_1 \cup \ldots \cup B_{i-1} \nvdash \phi$. Hence, $B_1 \cup \ldots \cup B_i \vdash \phi$ and

$B_1 \cup \ldots \cup B_{i-1} \nvdash \phi$. □

b) Let $\Sigma^{MS}_{A_p} = B_1 \cup \ldots \cup B_n$. If $B_1 \cup \ldots \cup B_i \vdash \phi$ and $B_1 \cup \ldots \cup B_{i-1} \nvdash \phi$, then there exists $X \subseteq B_1 \cup \ldots \cup B_i$ such that $Sen(X) \vdash \phi$ and let $\omega \in Sen(X)$, $Sen(X)\backslash\{\omega\} \nvdash \phi$. By the definition of *kernel*, $X \in K^{\perp\!\!\perp}\phi$. Then since $B_1 \cup \ldots \cup B_i \vdash \phi$ and $B_1 \cup \ldots \cup B_{i-1} \nvdash \phi$ it follows by Definition 3 and Definition 6 that $Ag(min(X)) = A_i$. Suppose that $Z \in K^{\perp\!\!\perp}\phi$ such that $Z \neq X$, then we can note that $Z \nsubseteq B_1 \cup \ldots \cup B_{i-1}$. Hence, $Z \subseteq B_1 \cup \ldots \cup B_j$ where $i \leq j \leq n$ and $Ag(min(Z)) = A_j$. Then, by assumption 4 and since $i \leq j \leq n$, it holds $A_j \leq_{Co} A_i$. Hence, $Ag(min(Z)) \leq_{co} Ag(min(X))$, so $Ag(max(min(X) \cup min(Z))) = A_i$. Therefore, $Ag(max(\bigcup_{X \in K^{\perp\!\!\perp}\phi} min(X))) = A_i$ that is the same $Pl(\phi, K) = A_i$. □

Example 6 *Let us consider again Example 4. As stated above, we can see K_{A_1} in a stratified fashion. That is, $K_{A_1} = \Sigma^{MS}_{A_1}$ can be viewed as a layered knowledge base $\Sigma^{MS}_{A_1} = B_1 \cup B_2 \cup B_3$, s. t. $B_1 = \{(\phi \rightarrow \psi, A_1), (\omega \rightarrow \psi, A_1), (\rho, A_1)\}$, $B_2 = \{(\phi, A_2), (\phi \rightarrow \psi, A_2), (\phi \rightarrow \varphi, A_2), (\omega \rightarrow \rho, A_2)\}$ and $B_3 = \{(\psi, A_3), (\omega, A_3), (\varphi \rightarrow \psi, A_3)\}$.*

Suppose that agent A_1 needs to calculate the plausibility of ψ. Following the procedure proposed in [Benferhat et al.*, 1993], we can note that $Sen(B_1) \nvdash \psi$ and $Sen(B_1 \cup B_2) \vdash \psi$. Hence, the weight of ψ is based on agent identifier A_2. This result is the same returned in Example 4 when our system was used.*

5 Non Prioritized Revision Using Plausibility

In this section, the behavior of a new revision operator based on sentence plausibility will be shown. When a belief base $K \in \mathcal{K}$ is revised by a tuple (ϕ, A_i) we will have two cases:

$-$ ϕ is consistent with $Bel(K)$. This is the most simple case to characterize from the logical point of view because it consists only in the addition of new tuples. In the limit case in which $\phi \in Bel(K)$ then this operation could increase the plausibility of ϕ.

$-$ ϕ is inconsistent with $Bel(K)$, that is $\neg\phi \in Bel(K)$. This case requires a deeper analysis because: a) it is necessary to determine whether the sentence will be accepted; and b) if the input is accepted then it is necessary to erase some tuples from K. For the second case we need to define an incision function on each ϕ-kernel.

We will adapt the incision function definition proposed by [Hansson, 1994] to our framework.

Definition 13 *An **incision function** σ for $K \in \mathcal{K}$ is a function such that for all ϕ:* 1) *$\sigma(K^{\perp\!\!\perp}\phi) \subseteq \cup(K^{\perp\!\!\perp}\phi)$, and* 2) *if $X \in K^{\perp\!\!\perp}\phi, X \neq \emptyset$, then $X \cap \sigma(K^{\perp\!\!\perp}\phi) \neq \emptyset$.*

The incision function selects the sentences to be discarded from K. Therefore, the subset of K that is not affected by the incision should equal the outcome of the contraction of K by ϕ. In the definition of *incision function* in Hansson's work it is not specified how the function selects the sentences that will be discarded from each ϕ-kernel. This can be solved with the sentence plausibility that we defined above. The incision function σ will select the lesser plausibility sentences from each ϕ-kernel. Hence, the new operator differs from the kernel revision operator defined by Hansson in the following issues:

1. The new operator makes an analysis to determine if the revision is necessary.
2. The sentence selection for the incision function is defined.

According to 1, the new operator permits two options: completely accept all the input, or completely reject all the input. For this reason the new operator is non prioritized. An example of non prioritized operators of the literature that completely accept or reject the input is Semi-Revision [Hansson, 1997]. Other operators may partially accept the new information, for instance Revision by a Set of Sentences [Falappa *et al.*, 2002] and Selective Revision [Fermé and Hansson, 1999].

Next we will define a specific incision function, based on belief plausibility, that will select the lesser plausibility sentences of each ϕ-kernel (following the principle of minimal change).

Definition 14 *$\sigma_\downarrow$ is a **bottom incision function** for K if $\sigma_\downarrow$ is an incision function such that, $\sigma_\downarrow(K^{\perp\!\!\perp}\phi) = \{(\varphi, A_i) : (\varphi, A_i) \in X \in K^{\perp\!\!\perp}\phi$ and for all $(\psi, A_j) \in X$ it holds that $A_i \leq_{Co} A_j\}$.*

Example 7 *Consider a set $\mathcal{A}gents = \{A_1, A_2, A_3\}$ where the credibility order is $A_1 \leq_{Co} A_2$, $A_2 \leq_{Co} A_3$. Suppose that agent A_2 has the following belief base $K_{A_2} = \{(\phi, A_3), (\psi, A_2), (\psi \rightarrow \phi, A_1), (\omega, A_1), (\omega \rightarrow \phi, A_3), (\varphi, A_1)\}$. Then, $K^{\perp\!\!\perp}_{A_2}\phi = \{H_a, H_b, H_c\}$ where $H_a = \{(\phi, A_3)\}$, $H_b = \{(\psi, A_2), (\psi \rightarrow \phi, A_1)\}$, $H_c = \{(\omega, A_1), (\omega \rightarrow \phi, A_3)\}$. Then, $\sigma_\downarrow(K^{\perp\!\!\perp}_{A_2}\phi) = \{(\phi, A_3), (\psi \rightarrow \phi, A_1), (\omega, A_1)\}$.*

Now that we have given the necessary background on the behavior of the new operator, *Non-Prioritized Revision Using Plausibility* will be defined.

Definition 15 *Let $K \in \mathcal{K}$, $\phi \in \mathcal{L}$, and $K^{\perp\!\!\perp}\phi$ be the set of ϕ-kernels of K. Let $\sigma_\downarrow$ be a bottom incision function* [1] *for K. The operator "$\circ$", called **Non-Prioritized Revision Using Plausibility**, is defined as follows:*

$$K \circ (\phi, A_i) = \begin{cases} K \cup \{(\phi, A_i)\} & \text{if } \neg\phi \notin Bel(K) \\ K & \text{if } \neg\phi \in Bel(K) \\ & \text{and } A_i \leq_{Co} Pl(\neg\phi, K) \\ (K \backslash X) \cup \{(\phi, A_i)\} & \text{if } \neg\phi \in Bel(K) \\ & \text{and } Pl(\neg\phi, K) <_{Co} A_i \end{cases}$$

where: $X = \{(\omega, A_j) : \omega \in Sen(\sigma_\downarrow(K^{\perp\!\!\perp}\neg\phi))$ and $(\omega, A_j) \in K\}$.

Example 8 *Consider a set $\mathcal{A}gents = \{A_1, A_2, A_3, A_4\}$ where the credibility order is $A_4 \leq_{Co} A_3$, $A_3 \leq_{Co} A_2$, $A_2 \leq_{Co} A_1$. Suppose that agent A_4 has the following belief base $K_{A_4} = \{(\psi, A_4), (\phi, A_3), (\phi \rightarrow \psi, A_3), (\phi \rightarrow \psi, A_2), (\omega \rightarrow \psi, A_2), (\omega, A_4), (\phi \rightarrow \varphi, A_3), (\varphi \rightarrow \psi, A_4), (\rho, A_2), (\omega \rightarrow \rho, A_3)\}$. Furthermore, suppose A_4 receives the tuple $(\neg\psi, A_1)$. Then, A_4 should revise K_{A_4} by $(\neg\psi, A_1)$. Next we will describe the behavior of the new operator step by step.*

• Step 1. *Obtain the minimal subsets that derive ψ from belief base K_{A_4}.*

[1] Observe that the outcome of a *bottom incision function* would be similar to that of a "*safe contraction*" [Alchourrón and Makinson, 1985].

$K^{\perp\!\!\perp}_{A_4}\psi = \{H_a, H_b, H_c, H_d, H_e\}$, where $H_a = \{(\psi, A_4)\}$, $H_b = \{(\phi, A_3), (\phi \rightarrow \psi, A_3)\}$, $H_c = \{(\phi, A_3), (\phi \rightarrow \psi, A_2)\}$, $H_d = \{(\omega, A_4), (\omega \rightarrow \psi, A_2)\}$ and $H_e = \{(\phi, A_3), (\phi \rightarrow \varphi, A_3), (\varphi \rightarrow \psi, A_4)\}$.

• Step 2. *Apply the bottom incision function "$\sigma_\downarrow$" to $K^{\perp\!\!\perp}_{A_4}\psi$ to obtain the set containing the lesser plausibility tuples from each ψ-kernel.*

$\sigma_\downarrow(K^{\perp\!\!\perp}_{A_4}\psi) = \{(\psi, A_4), (\phi, A_3), (\phi \rightarrow \psi, A_3), (\omega, A_4), (\varphi \rightarrow \psi, A_4)\}$.

• Step 3. *Obtain from the tuples of the previous step the set containing the greater plausibility tuples determined by the greater-credibility sources function: $max(\{(\psi, A_4), (\phi, A_3), (\phi \rightarrow \psi, A_3), (\omega, A_4), (\varphi \rightarrow \psi, A_4)\}) = \{(\phi, A_3), (\phi \rightarrow \psi, A_3)\}$.*

• Step 4. *Compare the agent identifier of the input tuple with the agent identifier of any tuple obtained in the previous step. Since $A_3 \leq_{Co} A_1$, then $K_{A_4} \circ (\neg\psi, A_1) = \{(\omega \rightarrow \psi, A_2), (\phi \rightarrow \varphi, A_3), (\rho, A_2), (\omega \rightarrow \rho, A_3), (\neg\psi, A_1)\}$. If the input is (ψ, A_4) rather than (ψ, A_1), then the revision will not have effect because $A_4 \leq_{Co} A_3$.*

Note that, since the belief base may be redundant, in Step 4 of Example 8 if the revision gives rise to a contraction then we will discard from K all those tuples whose sentences were selected by the bottom incision function without regarding the respective informants. Besides, note that our operator will never discard more plausible sentences than the input. This control can be seen in Step 4 of Example 8.

In this approach, we have assumed that the credibility order among agents is fixed. However, this order may be replaced and this will not affect the behavior of the operator. If the credibility order among agents changes, then the plausibility of all sentences may also change without changing the belief bases of the agents. This feature was one of the motivations for using agent identifiers instead of representing explicitly the plausibility of sentences as a number. For instance, consider a set $\mathcal{A}gents = \{A_1, A_2\}$ where the credibility order is $A_2 \leq_{Co} A_1$, $K_{A_1} = \{(\phi, A_2), (\psi, A_1)\}$ and $K_{A_2} = \{(\omega, A_1), (\rho, A_2)\}$. Hence, $\phi \preceq_{K_{A_1}} \psi$ and $\rho \preceq_{K_{A_2}} \omega$. If the credibility order changes to $A_1 \leq_{Co} A_2$ then $\psi \preceq_{K_{A_1}} \phi$ and $\omega \preceq_{K_{A_2}} \rho$. Note that the tuples in K_1 and K_2 remain unchanged.

If the behavior of the new operator follows Definition 15 then the operator follows the principles stated below. These are:

– Maintenance of Consistency [Dalal, 1988]: If a belief base K and a belief α are both consistent, then K revised by α is consistent.

– Minimal changes: As much old knowledge as possible should be retained in the revised knowledge. Namely, in this work, if the revision gives rise to a contraction, then the revision should discard those beliefs that are less plausible.

– Non-Prioritization: If a belief base K is revised by a belief α, the new belief is not necessarily accepted in the revised belief base. In other words, in this paper the revision could have no effect if some beliefs, that will be possibly discarded, are more plausible than the input.

Lemma 1 The non-prioritized revision using plausibility follows the principle of Maintenance of Consistency.

Proof: Let $K \in \mathcal{K}$ be a belief base and let $\phi \in \mathcal{L}$. Suppose that K and ϕ are consistent. By Definition 15 (*Non-Prioritized Revision Using Plausibility*) we have three cases:

– If $\neg\phi \notin Bel(K)$ then $K \circ (\phi, A_i) = K \cup \{(\phi, A_i)\}$.
ϕ and K are consistent, hence since $\neg\phi \notin Bel(K)$ then $K \cup \{(\phi, A_i)\}$ is consistent.

– If $\neg\phi \in Bel(K)$ and $A_i \leq_{Co} Pl(\neg\phi, K)$, then $K \circ (\phi, A_i) = K$ and we are done.

– If $\neg\phi \in Bel(K)$ and $Pl(\neg\phi, K) <_{Co} A_i$ then $K \circ (\phi, A_i) = (K \setminus X) \cup \{(\phi, A_i)\}$ where $X = \{(\omega, A_j) : \omega \in Sen(\sigma_\downarrow(K^{\perp\!\!\perp}\neg\phi))$ and $(\omega, A_j) \in K\}$.

We must show that $\neg\phi \notin Bel(K \setminus X)$.
Suppose that $\neg\phi \in Bel(K \setminus X)$. Then the $\neg\phi$-kernels were not cut by the *bottom incision function*. That is, no sentences from $\neg\phi$-kernels were removed. However, this is absurd by Definition 14 (*bottom incision function.*) The contradiction comes from supposing $\neg\phi \in Bel(K \setminus X)$. Then $\neg\phi \notin Bel(K \setminus X)$.

Hence, the non-prioritized revision using plausibility follows the principle of Maintenance of Consistency. □

The following two lemmas are straightforward from Definitions 14 and 15.

Lemma 2 The new operator follows the principle of Minimal Change.

Lemma 3 The new operator follows the principle of Non-Prioritization.

6 Discussion and Conclusions

In this paper we have analyzed and compared two methods that follow different attitudes when facing inconsistent knowledge. One attitude is to revise the knowledge base in order to preserve its consistency, while the other is to cope with the inconsistency. In this work we have presented an approach proposed in [Tamargo *et al.*, 2008] that follows the first attitude. Furthermore, we have shown an approach proposed in [Benferhat *et al.*, 1993] that follows the second attitude. The work in [Tamargo *et al.*, 2008] presents a formalism for knowledge representation and consistency maintenance in a MAS where deliberative agents can receive new information from others through communication. This type of Belief Revision is called *Belief Revision using information from Multiple Sources* (MSBR). Similarly to Dragoni [Dragoni *et al.*, 1994] and Cantwell [Cantwell, 1998], informant agents can have different levels of credibility and these levels are used to decide which information prevails when a contradiction arises. However, our work has differences with that of these authors as was discussed in [Tamargo *et al.*, 2008].

We have also analyzed the argumentative consequence proposed in [Benferhat *et al.*, 1993]. Furthermore, we have shown a procedure that can be used to determine if a sentence is an argumentative consequence of a knowledge base. Our proposal, which is based on the definition of a revision operator, differs from the one proposed in [Benferhat *et al.*, 1993] which is based on the definition of inconsistency-tolerant consequence relations in inconsistent stratified knowledge bases. This difference lies in the posture adopted when facing inconsistent knowledge. In [Benferhat *et al.*, 1993], inconsistency is accepted obtaining argumentative consequences, whereas in our approach we start from consistent knowledge bases and through a revision operator based on the kernel contraction defined in [Hansson, 1994] we preserve consistency.

Furthermore, in [Benferhat *et al.*, 1993] the stratification ($\Sigma = B_1 \cup \ldots \cup B_n$) is modeled by attaching a weight $\alpha \in [0,1]$ to each formula with the convention that $(\phi\, \alpha_i) \in B_i, i = 1, \ldots, n$, and $\alpha_1 = 1 > \alpha_2 > \ldots > \alpha_n > 0$. Here tuples are stored in a similar way to [Benferhat *et al.*, 1993]. However, in [Tamargo *et al.*, 2008] an agent identifier is attached to each formula (the source). When the agent identifiers are totally ordered, it is equivalent to use weights (as in possibilistic logic) or agent identifiers. However, in our framework it is possible to define a *partial* order among agent identifiers and this is part of our future work.

Here, as in [Tamargo *et al.*, 2008], we have assumed that the credibility order among agents is fixed, as stated in Section 5. This ordering can be changed without affecting the behavior of the operator. If the credibility order among agents changes, then the plausibility of all sentences will also change without having to modify the belief bases of the agents. This characteristic is one of the motivations for using agent identifiers instead of representing the plausibility of sentences as a number. As future work we propose to define a revision operator capable of revising the credibility order. This will allow us to represent changes over the credibility order.

The revision operator presented in this article could use stratified bases as in [Benferhat *et al.*, 1993] and a new revision operator could be defined based on the procedure proposed in the mentioned article. This operator would have a similar behavior to the partial meet revision defined in [Alchourrón *et al.*, 1985], due to the fact that in the procedure for argumentative consequence as well as in partial meet, maximal subsets are obtained. As future work we will propose a revision operator with these features.

As we can see there exist several differences between [Benferhat *et al.*, 1993] and the work presented here. However, in this article we showed, in Section 4, that the systems used in each approach are equivalent under certain reasonable restrictions.

References

[Alchourrón and Makinson, 1985] C. Alchourrón and D. Makinson. On the logic of theory change: Safe contraction. *St. Logica*, 44:405–422, 1985.

[Alchourrón *et al.*, 1985] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *J. of Symbolic Logic*, 50(2):510–530, 1985.

[Benferhat *et al.*, 1992] S. Benferhat, D. Dubois, and H. Prade. Representing default rules in possibilistic logic. In *3rd KR'92*, pages 673–684, Cambridge, MA, 1992.

[Benferhat *et al.*, 1993] S. Benferhat, D. Dubois, and H. Prade. Argumentative inference in uncertain and inconsistent knowledge bases. In *In Proc. of Uncertainty in Artificial Intelligence*, pages 411–419, 1993.

[Benferhat *et al.*, 2002] S. Benferhat, D. Dubois, H. Prade, and M. A. Williams. A practical approach to revising prioritized knowledge bases. *Studia Logica*, 70(1):105–130, 2002.

[Cantwell, 1998] J. Cantwell. Resolving conflicting information. *Journal of Logic, Language and Information*, 7(2):191–220, 1998.

[Dalal, 1988] M. Dalal. Investigations into a theory of knowledge base revision. *Proceedings of AAAI*, pages 475–479, 1988.

[Dragoni *et al.*, 1994] A. Dragoni, P. Giorgini, and P. Puliti. Distributed belief revision versus distributed truth maintenance, 1994.

[Dubois *et al.*, 1992] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In *H. of Logic in Artificial Intelligent and Logict Programming*, volume 3, 1992.

[Falappa *et al.*, 2002] M. A. Falappa, G. Kern-Isberner, and G. R. Simari. Explanations, belief revision and defeasible reasoning. *Artificial Intelligence*, 141(1):1–28, 2002.

[Fermé and Hansson, 1999] E. L. Fermé and S. O. Hansson. Selective revision. *Studia Logica*, 63(3):331–342, 1999.

[Fuhrmann, 1991] A. Fuhrmann. Theory contraction through base contraction. *Journal of Philosophical Logic*, 20(2):175–203, may 1991.

[Gärdenfors and Makinson, 1988] P. Gärdenfors and D. Makinson. Revisions of knowledge systems using epistemic entrenchment. In *Second Conference on Theoretical Aspects of Reasoning about Knowledge Conference*, pages 83–95, 1988.

[Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux - Modeling the Dynamic of Epistemic States*. The MIT Press, Cambridge, Mass, 1988.

[Hansson, 1992] S. O. Hansson. In defense of base contraction. *Synthese*, 91(3):239–245, june 1992.

[Hansson, 1994] S. O. Hansson. Kernel contraction. *Journal of Symbolic Logic*, 59(3):845–859, 1994.

[Hansson, 1997] S. O. Hansson. Semi-revision. *J. of Applied Non-Classical Logic*, pages 151–175, 1997.

[Hansson, 1999] S. O. Hansson. *A Textbook of Belief Dynamics: Theory Change and Database Updating*. 1999.

[Kfir-Dahav and Tennenholz, 1996] N. E. Kfir-Dahav and M. Tennenholz. Multi-agent belief revision. In *Theoretical Aspects of Rationality and Knowledge: Pro. of the Sixth Conf. (TARK 1996)*, pages 175–196. San Francisco, 1996.

[Liu and Williams, 1999] W. Liu and M. A. Williams. A framework for multi-agent belief revision, part i: The role of ontology. In *Australian Joint Conference on A. I.*, pages 168–179, 1999.

[Liu and Williams, 2001] W. Liu and M. A. Williams. A framework for multi-agent belief revision. *Studia Logica*, 67(2):291–312, 2001.

[Tamargo *et al.*, 2008] L. H. Tamargo, A. J. García, M. A. Falappa, and G. R. Simari. Consistency maintenance of plausible belief bases based on agents credibility. *12th International Workshop on Non-Monotonic Reasoning (NMR)*, pages 50–58, 2008.

On the Revision of Action Laws: An Algorithmic Approach

Ivan José Varzinczak

Meraka Institute
CSIR, Pretoria, South Africa
ivan.varzinczak@meraka.org.za

Abstract

Domain descriptions in reasoning about actions are logical theories and as such they may also evolve. Given that, knowledge engineers also need revision tools to incorporate new incoming laws about the dynamic environment. Here we fill this gap by providing an algorithmic approach for revision of action laws. We give a well defined semantics that ensures minimal change w.r.t. the original models, and show correctness of our algorithms w.r.t. the semantic constructions.

1 Introduction

Like any logical theory, action theories in reasoning about actions may evolve, and thus need revision methods to adequately accommodate new information about the behavior of actions. Recently, update and contraction-based methods for action theory change have been defined [Eiter *et al.*, 2005; Herzig *et al.*, 2006; Varzinczak, 2008]. Here we continue this important though quite new thread of investigation and develop a minimal change approach for *revising* laws of an action domain description.

The motivation is as follows. Consider an agent designed to interact with a coffee machine (Figure 1).

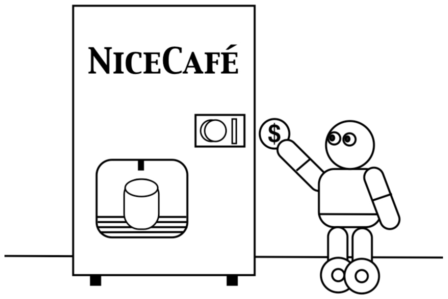


Figure 1: The coffee deliverer agent.

Among her beliefs, the agent may know that a coffee is a hot drink, that after buying she gets a coffee, and that with a token it is possible to buy. We can see the agent's beliefs about the behavior of actions in this scenario as a transition system (Figure 2).

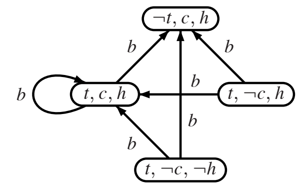


Figure 2: A transition system depicting the agent's knowledge about the dynamics of the coffee machine. b, t, c, and h stand for, respectively, *buy*, *token*, *coffee*, and *hot*.

Now, it may be the case that the agent learns that coffee is the only hot drink available at the machine, or that even without a token she can still buy, or that all possible executions of *buy* should lead to states where $\neg$*token* is the case. These are examples of *revision* with new laws about the dynamics of the environment under consideration. And here we are interested in exactly these kinds of theory modification.

The contributions of the present work are as follows:

- What is the semantics of revising an action theory by a law? How to get minimal change, i.e., how to keep as much knowledge about other laws as possible?
- How to syntactically revise an action theory so that its result corresponds to the intended semantics?

Here we answer these questions.

2 Logical Preliminaries

Our base formalism is multimodal logic K_n [Popkorn, 1994].

2.1 Action Theories in Multimodal K

Let $\mathfrak{A} = \{a_1, a_2, \ldots, a_n\}$ be the set of *atomic actions* of a domain. To each action a there is associated a modal operator $[a]$. $\mathfrak{P} = \{p_1, p_2, \ldots, p_n\}$ denotes the set of *propositions*, or *atoms*. $\mathfrak{L} = \{p, \neg p \ : \ p \in \mathfrak{P}\}$ is the set of literals. ℓ denotes a literal and $|\ell|$ the atom in ℓ.

We use $\varphi, \psi, \ldots$ to denote *Boolean formulas*. $\mathfrak{F}$ is the set of all Boolean formulas. A propositional valuation v is a *maximally consistent* set of literals. We denote by $v \Vdash \varphi$ the fact that v satisfies φ. By $val(\varphi)$ we denote the set of all valuations satisfying φ. $\models_{\mathsf{CPL}}$ is the classical consequence relation. $Cn(\varphi)$ denotes all logical consequences of φ.

With $IP(\varphi)$ we denote the set of *prime implicants* [Quine, 1952] of φ. By π we denote a prime implicant, and $atm(\pi)$ is the set of atoms occurring in π. Given ℓ and π, $\ell \in \pi$ abbreviates 'ℓ is a literal of π'. For a given set A, $\bar{A}$ denotes its complement. Hence $\overline{atm(\pi)}$ denotes $\mathfrak{P} \setminus atm(\pi)$.

We use $\Phi, \Psi, \ldots$ to denote complex formulas (possibly with modal operators). $\langle a \rangle$ is the dual operator of $[a]$ ($\langle a \rangle \Phi =_{\text{def}} \neg[a]\neg\Phi$).

A K_n*-model* is a tuple $\mathscr{M} = \langle W, R \rangle$ where W is a set of valuations, and R maps action constants a to accessibility relations $R_a \subseteq W \times W$. Given $\mathscr{M}$, $\models^{\mathscr{M}}_{w} p$ (p is true at world w of model $\mathscr{M}$) if $w \Vdash p$; $\models^{\mathscr{M}}_{w} [a]\Phi$ if $\models^{\mathscr{M}}_{w'} \Phi$ for every w' s.t. $(w, w') \in R_a$; truth conditions for the other connectives are as usual. By $\mathcal{M}$ we will denote a set of K_n-models.

$\mathscr{M}$ is a model of Φ (noted $\models^{\mathscr{M}} \Phi$) if and only if $\models^{\mathscr{M}}_{w} \Phi$ for all $w \in W$. $\mathscr{M}$ is a model of a set of formulas Σ (noted $\models^{\mathscr{M}} \Sigma$) if and only if $\models^{\mathscr{M}} \Phi$ for every $\Phi \in \Sigma$. Φ is a *consequence of the global axioms* Σ in all K_n-models (noted $\Sigma \models_{\mathsf{K}_n} \Phi$) if and only if for every $\mathscr{M}$, if $\models^{\mathscr{M}} \Sigma$, then $\models^{\mathscr{M}} \Phi$.

In K_n we can state laws describing the behavior of actions. Here we distinguish three types of them.

Static Laws A *static law* is a formula $\varphi \in \mathfrak{F}$ that characterizes the possible states of the world. An example is *coffee* $\rightarrow$ *hot*: if the agent holds a coffee, then she holds a hot drink. The set of static laws of a domain is denoted by $\mathcal{S}$.

Effect Laws An *effect law for a* has the form $\varphi \rightarrow [a]\psi$, with $\varphi, \psi \in \mathfrak{F}$. Effect laws relate an action to its effects, which can be conditional. The consequent ψ is the effect that always obtains when a is executed in a state where the antecedent φ holds. An example is *token* $\rightarrow$ $[buy]hot$: whenever the agent has a token, after buying, she has a hot drink. If ψ is inconsistent we have a special kind of effect law that we call an *inexecutability law*. For example, $\neg token \rightarrow [buy]\bot$ says that *buy* cannot be executed if the agent has no token. The set of effect laws of a domain is denoted by $\mathcal{E}$.

Executability Laws An *executability law for a* has the form $\varphi \rightarrow \langle a \rangle \top$, with $\varphi \in \mathfrak{F}$. It stipulates the context in which a is guaranteed to be executable. (In K_n $\langle a \rangle \top$ reads "a's execution is possible".) For instance, $token \rightarrow \langle buy \rangle \top$ says that buying can be executed whenever the agent has a token. The set of executability laws of a domain is denoted by $\mathcal{X}$.

Given a, $\mathcal{E}_a$ (resp. $\mathcal{X}_a$) will denote the set of only those effect (resp. executability) laws about a.

Action Theories $\mathcal{T} = \mathcal{S} \cup \mathcal{E} \cup \mathcal{X}$ is an *action theory*.

2.2 The Frame, Ramification and Qualification Problems

To make the presentation more clear to the reader, we here assume that the agent's theory contains all frame axioms. However, all we shall say here can be defined within a formalism with a solution to the frame and ramification problems like done by Herzig *et al*. [2006].

Given the acknowledged difficulty of the qualification problem, we do not assume here any a priori solution to it. Instead, we suppose the knowledge engineer may want to state some (not necessarily fully specified) executability laws for some actions. These may be incorrect at the starting point, but revising wrong executability laws is an approach towards its solution and one of the aims of this work. With further information the knowledge engineer will have the chance to change them so that eventually they will correspond to the intuition (cf. Section 3).

The action theory of our running example will thus be:

$$\mathcal{T} = \left\{ \begin{array}{c} coffee \rightarrow hot, token \rightarrow \langle buy \rangle \top, \\ \neg coffee \rightarrow [buy]coffee, \neg token \rightarrow [buy]\bot, \\ coffee \rightarrow [buy]coffee, hot \rightarrow [buy]hot \end{array} \right\}$$

Figure 2 above shows a K_n-model for the theory $\mathcal{T}$.

2.3 Supra-models

Sometimes it will be useful to consider models whose possible worlds are *all* the possible states allowed by $\mathcal{S}$:

Definition 1 *$\mathscr{M} = \langle W, R \rangle$ is a* big frame *of $\mathcal{T}$ if and only if:*

- *$W = val(\mathcal{S})$; and*
- *$R = \bigcup_{a \in \mathfrak{A}} R_a$, where*

$$R_a = \{(w, w') : \forall.\varphi \rightarrow [a]\psi \in \mathcal{E}_a,\ if\ \models^{\mathscr{M}}_{w} \varphi\ then\ \models^{\mathscr{M}}_{w'} \psi\}$$

Big frames of $\mathcal{T}$ are not unique and not always models of $\mathcal{T}$.

Definition 2 *$\mathscr{M}$ is a* supra-model *of $\mathcal{T}$ iff $\models^{\mathscr{M}} \mathcal{T}$ and $\mathscr{M}$ is a big frame of $\mathcal{T}$.*

Figure 3 depicts a supra-model of our example $\mathcal{T}$.

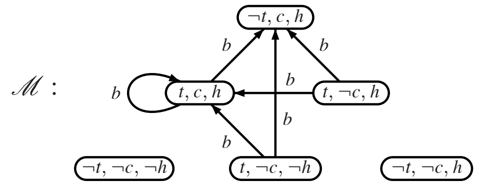


Figure 3: Supra-model for the coffee machine scenario.

2.4 Prime Valuations

An atom p is *essential* to φ if and only if $p \in atm(\varphi')$ for all φ' such that $\models_{\mathsf{CPL}} \varphi \leftrightarrow \varphi'$. For instance, p_1 is essential to $\neg p_1 \wedge (\neg p_1 \vee p_2)$. $atm!(\varphi)$ will denote the essential atoms of φ. (If φ is a tautology or a contradiction, then $atm!(\varphi) = \emptyset$.)

For $\varphi \in \mathfrak{F}$, $\varphi*$ is the set of all $\varphi' \in \mathfrak{F}$ such that $\varphi \models_{\mathsf{CPL}} \varphi'$ and $atm(\varphi') \subseteq atm!(\varphi)$. For instance, $p_1 \vee p_2 \notin p_1*$, as $p_1 \models_{\mathsf{CPL}} p_1 \vee p_2$ but $atm(p_1 \vee p_2) \not\subseteq atm!(p_1)$. Clearly, $atm(\bigwedge \varphi*) = atm!(\bigwedge \varphi*)$. Moreover, whenever $\models_{\mathsf{CPL}} \varphi \leftrightarrow \varphi'$, then $atm!(\varphi) = atm!(\varphi')$ and also $\varphi* = \varphi'*$.

Theorem 1 ([Parikh, 1999]) *$\models_{\mathsf{CPL}} \varphi \leftrightarrow \bigwedge \varphi*$, and $atm(\varphi*) \subseteq atm(\varphi')$ for every φ' s.t. $\models_{\mathsf{CPL}} \varphi \leftrightarrow \varphi'$.*

Thus for every $\varphi \in \mathfrak{F}$ there is a unique least set of elementary atoms such that φ may equivalently be expressed using only atoms from that set. Hence, $Cn(\varphi) = Cn(\varphi*)$.

Given a valuation v, $v' \subseteq v$ is a *subvaluation*. For W a set of valuations, a subvaluation v' *satisfies* $\varphi \in \mathfrak{F}$ modulo W (noted $v' \Vdash_{W} \varphi$) if and only if $v \Vdash \varphi$ for all $v \in W$ such that $v' \subseteq v$. A subvaluation v *essentially satisfies* φ modulo W ($v \Vdash^{!}_{W} \varphi$) if and only if $v \Vdash_{W} \varphi$ and $\{|\ell| : \ell \in v\} \subseteq atm!(\varphi)$.

Definition 3 *Let $\varphi \in \mathfrak{F}$ and W be a set of valuations. A subvaluation v is a* prime subvaluation *of φ (modulo W) if and only if $v \overset{!}{\Vdash}_W \varphi$ and there is no $v' \subseteq v$ s.t. $v' \overset{!}{\Vdash}_W \varphi$.*

A prime subvaluation of a formula φ is one of the weakest states of truth in which φ is true. (Notice the similarity with the syntactical notion of prime implicant [Quine, 1952].) We denote all prime subvaluations of φ modulo *W* by $base(\varphi, W)$.

Theorem 2 *Let $\varphi \in \mathfrak{F}$ and W be a set of valuations. Then for all $w \in W$, $w \Vdash \varphi$ if and only if $w \Vdash \bigvee_{v \in base(\varphi,W)} \bigwedge_{\ell \in v} \ell$.*

2.5 Closeness Between Models

When revising a model, we perform a change in its structure. Because there can be several ways of modifying a model (not all of them minimal), we need a notion of distance between models to identify those that are closest to the original one.

As we are going to see in more depth in the sequel, changing a model amounts to modifying its possible worlds or its accessibility relation. Hence, the distance between two K_n-models will depend upon the distance between their sets of worlds and accessibility relations. These here will be based on the *symmetric difference* between sets, defined as $X \dot{-} Y = (X \setminus Y) \cup (Y \setminus X)$.

Definition 4 *Let $\mathscr{M} = \langle W, R \rangle$. $\mathscr{M}' = \langle W', R' \rangle$ is* at least as close to *$\mathscr{M}$ as $\mathscr{M}'' = \langle W'', R'' \rangle$, noted $\mathscr{M}' \preceq_{\mathscr{M}} \mathscr{M}''$, iff*

- *either $W \dot{-} W' \subseteq W \dot{-} W''$*
- *or $W \dot{-} W' = W \dot{-} W''$ and $R \dot{-} R' \subseteq R \dot{-} R''$*

This is an extension of Burger and Heidema's [2002] relation to our modal case. Note that other distance notions are also possible, like e.g. the *cardinality* of symmetric differences or Hamming distance. (See Section 7 for an explanation on why we have chosen this particular distance notion here.)

3 Semantics of Revision

Contrary to contraction, where we want the negation of a law to be *satisfiable*, in revision we want a new law to be *valid*. Thus we must eliminate all cases satisfying its negation.

The idea in our semantics is as follows: we initially have a set of models $\mathcal{M}$ in which a given formula Φ is (potentially) not valid, i.e., Φ is (possibly) not true in every model in $\mathcal{M}$. In the result we want to have only models of Φ. Adding Φ-models to $\mathcal{M}$ is of no help. Moreover, adding models makes us lose laws: the resulting theory would be more liberal.

One solution amounts to deleting from $\mathcal{M}$ those models that are not Φ-models. Of course removing only some of them does not solve the problem, we must delete every such a model. By doing that, all resulting models will be models of Φ. (This corresponds to *theory expansion*, when the resulting theory is satisfiable.) However, if $\mathcal{M}$ contains no model of Φ, we will end up with $\emptyset$. Consequence: the resulting theory is inconsistent. (This is the main revision problem.) In this case the solution is to *substitute* each model $\mathscr{M}$ in $\mathcal{M}$ by its *nearest modifications* $\mathscr{M}^{\star}_{\Phi}$ that makes Φ true. This lets us to keep as close as possible to the original models we had.

Before defining revision of sets of models, we present what modifications of (individual) models are.

3.1 Revising a Model by a Static Law

Suppose that our agent discovers that the only hot drink that is served on the machine is coffee. In this case, we might want to revise her beliefs with the new static law *coffee* $\leftrightarrow$ *hot*.

Considering the model in Figure 3, we see that $\neg$*coffee* $\wedge$ *hot* is satisfiable. As we do not want this, the first step is to *remove* all worlds in which $\neg$*coffee* $\wedge$ *hot* is true. The second step is to guarantee all the remaining worlds satisfy the new law. This issue has been largely addressed in the literature on belief revision and update [Gärdenfors, 1988; Winslett, 1988; Katsuno and Mendelzon, 1992; Herzig and Rifi, 1999]. Here we can achieve that with a semantics similar to that of classical revision operators: basically one can change the set of possible valuations, by removing or adding worlds.

In our example, removing the possible worlds $\{t, \neg c, h\}$ and $\{\neg t, \neg c, h\}$ would do the job (there is no need to add new valuations since the new static law is satisfied in at least one world of the original model).

The delicate point in removing worlds is that it may result in the loss of some executability laws: in the example, if there were only one arrow leaving some world w and pointing to $\{\neg t, \neg c, h\}$, then removing the latter from the model would make the action under concern no longer executable in w. Here we claim that this is intuitive: if the state of the world to which we could move is no longer possible, then we do not have a transition to that state anymore. Hence, if that transition was the only one we had, it is natural to lose it.

One could also ask what to do with the accessibility relation if new worlds must be added (revision case). We claim that it is reckless to blindly add new elements to *R*. Instead, we shall postpone correction of executability laws, if needed. This approach is debatable, but with the information we have at hand, it is the safest way of changing static laws.

The semantics for revision of one model by a static law is as follows:

Definition 5 *Let $\mathscr{M} = \langle W, R \rangle$. $\mathscr{M}' = \langle W', R' \rangle \in \mathscr{M}^{\star}_{\varphi}$ iff $W' = (W \setminus val(\neg\varphi)) \cup val(\varphi)$ and $R' \subseteq R$.*

Clearly $\models^{\mathscr{M}'} \varphi$ for all $\mathscr{M}' \in \mathscr{M}^{\star}_{\varphi}$. The minimal models of the revision of $\mathscr{M}$ by φ are those closest to $\mathscr{M}$ w.r.t. $\preceq_{\mathscr{M}}$:

Definition 6 $rev(\mathscr{M}, \varphi) = \bigcup \min\{\mathscr{M}^{\star}_{\varphi}, \preceq_{\mathscr{M}}\}$.

In the example of Figure 3, $rev(\mathscr{M}, coffee \leftrightarrow hot)$ is the singleton $\{\mathscr{M}'\}$, with $\mathscr{M}'$ as shown in Figure 4.

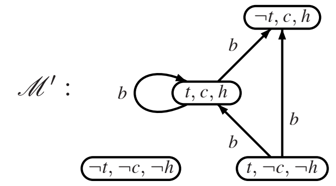


Figure 4: Revising model $\mathscr{M}$ in Figure 3 with *coffee* $\leftrightarrow$ *hot*.

3.2 Revising a Model by an Effect Law

Let's suppose now that our agent eventually discovers that after buying coffee she does not keep her token. This means that her theory should now be revised by the new effect law *token* $\rightarrow$ [*buy*]$\neg$*token*. Looking at model $\mathscr{M}$ in Figure 3, this amounts to guaranteeing that *token* $\wedge \langle buy \rangle$*token* is satisfiable in none of its worlds. To do that, we have to look at all the worlds satisfying this formula (if any) and

- either make *token* false in each of these worlds,
- or make $\langle buy \rangle$*token* false in all of them.

If we chose the first option, we will essentially flip the truth value of literal *token* in the respective worlds, which changes the set of valuations of the model. If we chose the latter, we will basically remove *buy*-arrows leading to *token*-worlds, which amounts to changing the accessibility relation.

In our example, the worlds $w_1 = \{token, coffee, hot\}$, $w_2 = \{token, \neg coffee, hot\}$ and $w_3 = \{token, \neg coffee, \neg hot\}$ satisfy *token* $\wedge \langle buy \rangle$*token*. Flipping *token* in all of them to $\neg$*token* would do the job, but would also have as consequence the introduction of a new static law: $\neg$*token* would now be valid, i.e., the agent never has a token! Do we want this?

We claim that changing action laws should not have as side effect a change in the static laws. These have a special status, and should change only if required (see Section 3.1). Hence each world satisfying *token* $\wedge \langle buy \rangle$*token* has to be changed so that $\langle buy \rangle$*token* becomes untrue in it. In the example, we thus should remove (w_1, w_1), (w_2, w_1) and (w_3, w_1) from *R*.

The semantics of one model revision for the case of a new effect law is:

Definition 7 *Let* $\mathscr{M} = \langle W, R \rangle$. $\mathscr{M}' = \langle W', R' \rangle \in \mathscr{M}^{\star}_{\varphi \rightarrow [a]\psi}$ *iff:*

- $W' = W$, $R' \subseteq R$, $\models^{\mathscr{M}'} \varphi \rightarrow [a]\psi$, *and*
- *If* $(w, w') \in R \setminus R'$, *then* $\models^{\mathscr{M}}_{w} \varphi$

The minimal models resulting from the revision of a model $\mathscr{M}$ by a new effect law are those closest to $\mathscr{M}$ w.r.t. $\preceq_{\mathscr{M}}$:

Definition 8 $rev(\mathscr{M}, \varphi \rightarrow [a]\psi) = \bigcup \min\{\mathscr{M}^{\star}_{\varphi \rightarrow [a]\psi}, \preceq_{\mathscr{M}}\}$.

Taking $\mathscr{M}$ as in Figure 3, $rev(\mathscr{M}, token \rightarrow [buy]\neg token)$ will be the singleton $\{\mathscr{M}'\}$ depicted in Figure 5.

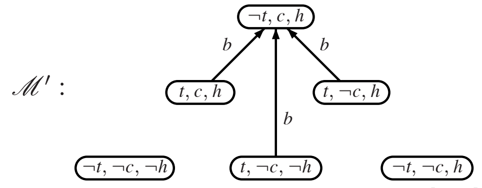


Figure 5: Revising $\mathscr{M}$ in Figure 3 with *token* $\rightarrow$ [*buy*]$\neg$*token*.

Note that adding effect laws will never require new arrows. This is the job of executability-revision.

3.3 Revising a Model by an Executability Law

Let us now suppose that at some stage it has been decided to grant free coffee to everybody. Faced with this information, we have to revise the agent's laws to reflect the fact that *buy* can also be executed in $\neg$*token*-contexts: $\neg$*token* $\rightarrow \langle buy \rangle \top$ is a new executability law (and hence we will have $\langle buy \rangle \top$ in all new models of the agent's beliefs).

Considering model $\mathscr{M}$ in Figure 3, we observe that $\neg$*token*$\wedge$[*buy*]$\bot$ is satisfiable. Hence we must throw $\neg$*token*$\wedge$ [*buy*]$\bot$ away to ensure the new law becomes true.

To remove $\neg$*token* $\wedge$ [*buy*]$\bot$ we have to look at all worlds satisfying it and modify $\mathscr{M}$ so that they no longer satisfy that formula. Given worlds $w_4 = \{\neg token, \neg coffee, \neg hot\}$ and $w_5 = \{\neg token, \neg coffee, hot\}$, we have two options: change the interpretation of *token* in both or add new arrows leaving these worlds. A question that arises is 'what choice is more drastic: change a world or an arrow'? Again, here we claim that changing the world's content (the valuation) is more drastic, as the existence of such a world is foreseen by some static law and is hence assumed to be as it is, unless we have enough information supporting the contrary, in which case we explicitly change the static laws (see Section 3.1). Thus we shall add a new *buy*-arrow from each of w_4 and w_5.

Having agreed on that, the issue now is: which worlds should the new arrows point to? In order to comply with minimal change, the new arrows shall point to worlds that are relevant targets of each of the $\neg$*token*-worlds in question.

Definition 9 *Let* $\mathscr{M} = \langle W, R \rangle$, $w, w' \in W$, *and* $\mathcal{M}$ *be a set of models s.t.* $\mathscr{M} \in \mathcal{M}$. *Then* w' *is a* relevant target world of w *w.r.t.* $\varphi \rightarrow \langle a \rangle \top$ *for* $\mathscr{M}$ *in* $\mathcal{M}$ *iff* $\models^{\mathscr{M}}_{w} \varphi$ *and*

- *If there is* $\mathscr{M}' = \langle W', R' \rangle \in \mathcal{M}$ *such that* $R'_a(w) \neq \emptyset$:
  - *for all* $\ell \in w' \setminus w$, *there is* $\psi' \in \mathfrak{F}$ *s.t. there is* $v' \in base(\psi', W)$ *s.t.* $v' \subseteq w'$, $\ell \in v'$, *and* $\models^{\mathscr{M}_i}_{w} [a]\psi'$ *for every* $\mathscr{M}_i \in \mathcal{M}$
  - *for all* $\ell \in w \cap w'$, *either there is* $\psi' \in \mathfrak{F}$ *s.t. there is* $v' \in base(\psi', W)$ *s.t.* $v' \subseteq w'$, $\ell \in v'$, *and* $\models^{\mathscr{M}_i}_{w} [a]\psi'$ *for all* $\mathscr{M}_i \in \mathcal{M}$; *or there is* $\mathscr{M}_i \in \mathcal{M}$ *s.t.* $\not\models^{\mathscr{M}_i}_{w} [a]\neg\ell$
- *If* $R'_a(w) = \emptyset$ *for every* $\mathscr{M}' = \langle W', R' \rangle \in \mathcal{M}$:
  - *for all* $\ell \in w' \setminus w$, *there is* $\mathscr{M}_i = \langle W_i, R_i \rangle \in \mathcal{M}$ *s.t. there is* $u, v \in W_i$ *s.t.* $(u, v) \in R_{i_a}$ *and* $\ell \in v \setminus u$
  - *for all* $\ell \in w \cap w'$, *there is* $\mathscr{M}_i = \langle W_i, R_i \rangle \in \mathcal{M}$ *s.t. there is* $u, v \in W_i$ *s.t.* $(u, v) \in R_{i_a}$ *and* $\ell \in u \cap v$, *or for all* $\mathscr{M}_i = \langle W_i, R_i \rangle \in \mathcal{M}$, *if* $(u, v) \in R_{i_a}$, *then* $\neg\ell \notin v \setminus u$

By $rt(w, \varphi \rightarrow \langle a \rangle \top, \mathscr{M}, \mathcal{M})$ *we denote the set of all relevant target worlds of* w *w.r.t.* $\varphi \rightarrow \langle a \rangle \top$ *for* $\mathscr{M}$ *in* $\mathcal{M}$.

In our example, $w_6 = \{\neg token, coffee, hot\}$ is the only relevant target world here: the two other $\neg$*token*-worlds violate the direct effect *coffee* of action *buy*, while the three *token*-worlds would make us violate the frame axiom $\neg$*token* $\rightarrow$ [*buy*]$\neg$*token*.

The semantics for one model revision by a new executability law is as follows:

Definition 10 *Let* $\mathscr{M} = \langle W, R \rangle$. $\mathscr{M}' = \langle W', R' \rangle \in \mathscr{M}^{\star}_{\varphi \rightarrow \langle a \rangle \top}$ *iff:*

- $W' = W$, $R \subseteq R'$, $\models^{\mathscr{M}'} \varphi \rightarrow \langle a \rangle \top$, *and*
- *If* $(w, w') \in R' \setminus R$, *then* $w' \in rt(w, \varphi \rightarrow [a]\bot, \mathscr{M}, \mathcal{M})$

The minimal models resulting from revising a model $\mathscr{M}$ by a new executability law are those closest to $\mathscr{M}$ w.r.t. $\preceq_{\mathscr{M}}$:

Definition 11 $rev(\mathscr{M}, \varphi \rightarrow \langle a \rangle \top) = \bigcup \min\{\mathscr{M}^{\star}_{\varphi \rightarrow \langle a \rangle \top}, \preceq_{\mathscr{M}}\}$.

In our running example, $rev(\mathscr{M}, \neg token \rightarrow \langle buy \rangle \top)$ is the singleton $\{\mathscr{M}'\}$, where $\mathscr{M}'$ is as shown in Figure 6.

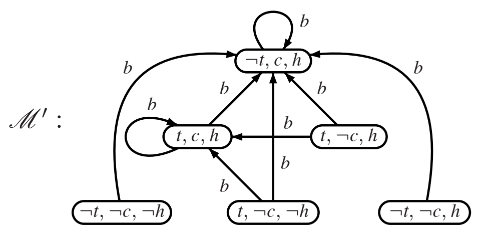


Figure 6: The result of revising model $\mathscr{M}$ in Figure 3 by the new executability law $\neg token \rightarrow \langle buy \rangle \top$.

3.4 Revising Sets of Models

Up until now we have seen what the revision of single models means. Now we are ready for a unified definition of revision of a set of models $\mathcal{M}$ by a new law Φ (cf. Section 5):

Definition 12 *Let $\mathcal{M}$ be a set of models and Φ a law. Then*

$$\mathcal{M}^{\star}_{\Phi} = (\mathcal{M} \setminus \{\mathscr{M} : \not\models^{\mathscr{M}} \Phi\}) \cup \bigcup_{\mathscr{M} \in \mathcal{M}} rev(\mathscr{M}, \Phi)$$

Definition 12 comprises both *expansion* and *revision*: in the former, addition of the new law gives a satisfiable theory; in the latter a deeper change is needed to get rid of inconsistency.

4 Algorithms for Revision of Laws

We now turn our attention to the syntactical counterpart of revision. Our endeavor here is to perform minimal change also at the syntactical level. By $\mathcal{T}^{\star}_{\Phi}$ we denote the result of revising an action theory $\mathcal{T}$ with a new law Φ.

4.1 Revising a Theory by a Static Law

Looking at the semantics of revision by Boolean formulas, we see that revising an action theory by a new static law may conflict with the executability laws: some of them may be lost and thus have to be changed as well.

The approach here is to preserve as many executability laws as we can in the old possible states. To do that, we look at each possible valuation that is common to the new $\mathcal{S}$ and the old one. Every time an executability used to hold in that state and no inexecutability holds there now, we make the action executable in such a context. For those contexts not allowed by the old $\mathcal{S}$, we make a inexecutable (cf. Section 3.1). Algorithm 1 deals with that (here $\mathcal{S} \star \varphi$ denotes the classical revision of $\mathcal{S}$ by φ built upon some well established method from the literature [Winslett, 1988; Katsuno and Mendelzon, 1992; Herzig and Rifi, 1999]. The choice of a particular operator for classical revision/update is not the main matter here, but rather whether it gives us a modified set of static laws entailing the new one).

In our example, revising the action theory $\mathcal{T}$ with a new static law *coffee* $\leftrightarrow$ *hot* will give us

$$\mathcal{T}^{\star}_{coffee \leftrightarrow hot} = \left\{ \begin{array}{c} coffee \leftrightarrow hot, \\ (token \wedge coffee \wedge hot) \rightarrow \langle buy \rangle \top, \\ (token \wedge \neg coffee \wedge \neg hot) \rightarrow \langle buy \rangle \top, \\ \neg coffee \rightarrow [buy] coffee, \neg token \rightarrow [buy] \bot, \\ coffee \rightarrow [buy] coffee, hot \rightarrow [buy] hot \end{array} \right\}$$

Algorithm 1 Revision by a Static Law

```
input: $\mathcal{T}, \varphi$
output: $\mathcal{T}^{\star}_{\varphi}$
  $\mathcal{S}' := \mathcal{S} \star \varphi$ /* classically revise $\mathcal{S}$ */
  $\mathcal{E}' := \mathcal{E}$ /* effect laws remain unchanged */
  $\mathcal{X}' := \emptyset$ /* executability laws will be ‘recovered’ from old $\mathcal{T}$ */
  for all $\pi \in IP(\mathcal{S}')$ do
    for all $A \subseteq \overline{atm(\pi)}$ do
      $\varphi_A := \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \in A}} p_i \wedge \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \notin A}} \neg p_i$
      /* by extending $\pi$ with $\varphi_A$ we get a valuation */
      if $\mathcal{S}' \not\models_{\mathsf{CPL}} (\pi \wedge \varphi_A) \rightarrow \bot$ /* context not removed */ then
        if $\mathcal{S} \not\models_{\mathsf{CPL}} (\pi \wedge \varphi_A) \rightarrow \bot$ then
          if $\mathcal{T} \models_{\mathsf{K}_n} (\pi \wedge \varphi_A) \rightarrow \langle a \rangle \top$ and $\mathcal{S}', \mathcal{E}', \mathcal{X} \not\models_{\mathsf{K}_n} \neg(\pi \wedge \varphi_A)$
          then
            $\mathcal{X}_a' := \{(\varphi_i \wedge \pi \wedge \varphi_A) \rightarrow \langle a \rangle \top : \varphi_i \rightarrow \langle a \rangle \top \in \mathcal{X}_a\}$
            /* preserve executability law in state not removed */
        else
          $\mathcal{E}' := \mathcal{E}' \cup \{(\pi \wedge \varphi_A) \rightarrow [a]\bot\}$
  $\mathcal{T}^{\star}_{\varphi} := \mathcal{S}' \cup \mathcal{E}' \cup \mathcal{X}'$
```

4.2 Revising a Theory by an Effect Law

When revising a theory by a new effect law $\varphi \rightarrow [a]\psi$, we want to eliminate all possible executions of a leading to $\neg\psi$-states. To achieve that, we look at all φ-contexts and every time a transition to some $\neg\psi$-context is not always the case, i.e., $\mathcal{T} \not\models_{\mathsf{K}_n} \varphi \rightarrow \langle a \rangle \neg\psi$, we can safely force $[a]\psi$ for that context. On the other hand, if in such a context there is always an execution of a to $\neg\psi$, then we should strengthen the executability laws to make room for the new effect in that context we want to add. Algorithm 2 below does the job.

Algorithm 2 Revision by an Effect Law

```
input: $\mathcal{T}, \varphi \rightarrow [a]\psi$
output: $\mathcal{T}^{\star}_{\varphi \rightarrow [a]\psi}$
  $\mathcal{T}' := \mathcal{T}$
  for all $\pi \in IP(\mathcal{S} \wedge \varphi)$ do
    for all $A \subseteq \overline{atm(\pi)}$ do
      $\varphi_A := \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \in A}} p_i \wedge \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \notin A}} \neg p_i$
      /* by extending $\pi$ with $\varphi_A$ we get a valuation */
      if $\mathcal{S} \not\models_{\mathsf{CPL}} (\pi \wedge \varphi_A) \rightarrow \bot$ /* is an allowed context */ then
        for all $\pi' \in IP(\mathcal{S} \wedge \neg \psi)$ do
          if $\mathcal{T}' \models_{\mathsf{K}_n} (\pi \wedge \varphi_A) \rightarrow \langle a \rangle \pi'$ /* $\neg\psi$ is achievable */
          then
            $\mathcal{T}' := (\mathcal{T}' \setminus \mathcal{X}_a') \cup \{(\varphi_i \wedge \neg(\pi \wedge \varphi_A)) \rightarrow \langle a \rangle \top : \varphi_i \rightarrow \langle a \rangle \top \in \mathcal{X}_a'\}$
            /* weaken executability laws */
        $\mathcal{T}' := \mathcal{T}' \cup \{(\pi \wedge \varphi_A) \rightarrow [a]\psi\}$ /* safely add the law */
        if $\mathcal{T}' \not\models_{\mathsf{K}_n} (\pi \wedge \varphi_A) \rightarrow [a]\bot$ then
          $\mathcal{T}' := \mathcal{T}' \cup \{(\varphi_i \wedge \pi \wedge \varphi_A) \rightarrow \langle a \rangle \top : \varphi_i \rightarrow \langle a \rangle \top \in \mathcal{T}\}$
          /* preserve other previous transitions */
  $\mathcal{T}^{\star}_{\varphi \rightarrow [a]\psi} := \mathcal{T}'$
```

In our running example, revision of the action theory $\mathcal{T}$ with the new effect law *token* $\rightarrow$ [*buy*]$\neg$*token* would give us

$\mathcal{T}^{\star}_{token \rightarrow [buy]\neg token} =$

$$\left\{\begin{array}{c} coffee \rightarrow hot, \\ (token \wedge \neg(token \wedge coffee \wedge hot)) \rightarrow \langle buy\rangle\top, \\ (token \wedge coffee \wedge hot) \rightarrow \langle buy\rangle\top, \\ (token \wedge \neg(token \wedge \neg coffee \wedge hot)) \rightarrow \langle buy\rangle\top, \\ (token \wedge \neg coffee \wedge hot) \rightarrow \langle buy\rangle\top, \\ (token \wedge \neg(token \wedge \neg coffee \wedge \neg hot)) \rightarrow \langle buy\rangle\top, \\ (token \wedge \neg coffee \wedge \neg hot) \rightarrow \langle buy\rangle\top, \\ \neg coffee \rightarrow [buy]coffee, \neg token \rightarrow [buy]\bot, \\ coffee \rightarrow [buy]coffee, hot \rightarrow [buy]hot, \\ token \rightarrow [buy]\neg token \end{array}\right\}$$

Regarding the bunch of new executability laws introduced in the resulting theory, observe that they can be easily simplified to the single one $token \rightarrow \langle buy\rangle\top$.

4.3 Revising a Theory by an Executability Law

Revision of a theory by a new executability law has as consequence a change in the effect laws: all those laws preventing the execution of a shall be weakened. Moreover, to comply with minimal change, we must ensure that in all models of the resulting theory there will be at most *one* transition by a from those worlds in which $\mathcal{T}$ precluded a's execution.

Let $(\mathcal{E}_a^{\varphi,\bot})_1, \ldots, (\mathcal{E}_a^{\varphi,\bot})_n$ denote minimum subsets (w.r.t. set inclusion) of $\mathcal{E}_a$ such that $\mathcal{S}, (\mathcal{E}_a^{\varphi,\bot})_i \models_{\mathsf{K}_n} \varphi \rightarrow [a]\bot$. (According to Herzig and Varzinczak [2007], one can ensure at least one such a set always exists.) Let $\mathcal{E}_a^- = \bigcup_{1\leq i\leq n}(\mathcal{E}_a^{\varphi,\bot})_i$. The effect laws in $\mathcal{E}_a^-$ will serve as guidelines to get rid of $[a]\bot$ in each φ-world allowed by $\mathcal{T}$: they are the laws to be weakened to allow for $\langle a\rangle\top$ in φ-contexts.

Our algorithm works as follows. To force $\varphi \rightarrow \langle a\rangle\top$ to be true in all models of the resulting theory, we visit every possible φ-context allowed by it and make the following operations to ensure $\langle a\rangle\top$ is the case for that context: Given a φ-context, if $\mathcal{T}$ does not always preclude a from being executed in it, we can safely force $\langle a\rangle\top$ without modifying other laws. On the other hand, if a is always inexecutable in that context, then we should weaken the laws in $\mathcal{E}_a^-$. The first thing we must do is to preserve all old effects in all other φ-worlds. To achieve that we specialize the above laws to each possible valuation (maximal conjunction of literals) satisfying φ but the actual one. Then, in the current φ-valuation, we must ensure that action a may have any effect, i.e., from this φ-world we can reach any other possible world. We achieve that by weakening the *consequent* of the laws in $\mathcal{E}_a^-$ to the exclusive disjunction of all possible contexts in $\mathcal{T}$. Finally, to get minimal change, we must ensure that all literals in this φ-valuation that are not forced to change are preserved. We do this by stating a conditional frame axiom of the form $(\varphi_k \wedge \ell) \rightarrow [a]\ell$, where φ_k is the above-mentioned φ-valuation.

Algorithm 3 gives the pseudo-code for that.

Algorithm 3 Revision by an Executability Law

```
input: $\mathcal{T}, \varphi \rightarrow \langle a\rangle\top$
output: $\mathcal{T}^{\star}_{\varphi \rightarrow \langle a\rangle\top}$
  $\mathcal{T}' := \mathcal{T}$
  for all $\pi \in IP(\overline{\mathcal{S} \wedge \varphi})$ do
    for all $A \subseteq \overline{atm(\pi)}$ do
      $\varphi_A := \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \in A}} p_i \wedge \bigwedge_{\substack{p_i \in \overline{atm(\pi)} \\ p_i \notin A}} \neg p_i$
      /* by extending $\pi$ with $\varphi_A$ we get a valuation */
      if $\mathcal{S} \not\models_{\mathsf{CPL}} (\pi \wedge \varphi_A) \rightarrow \bot$ /* is an allowed context */ then
        if $\mathcal{T}' \models_{\mathsf{K}_n} (\pi \wedge \varphi_A) \rightarrow [a]\bot$ then
          $\mathcal{T}' := \begin{array}{l} (\mathcal{T}' \setminus \mathcal{E}'^{-}_a) \cup \{(\varphi_i \wedge \neg(\pi \wedge \varphi_A)) \rightarrow [a]\psi_i : \\ \qquad \varphi_i \rightarrow [a]\psi_i \in \mathcal{E}'^{-}_a\} \cup \\ \{(\varphi_i \wedge \pi \wedge \varphi_A) \rightarrow [a] \bigoplus_{\substack{\pi' \in IP(\mathcal{S}) \\ A' \subseteq \overline{atm(\pi')}}} (\pi' \wedge \varphi_{A'}) : \\ \qquad \varphi_i \rightarrow [a]\psi_i \in \mathcal{E}'^{-}_a\} \end{array}$
          /* weaken the effect laws */
          for all $L \subseteq \mathfrak{L}$ do
            if $\mathcal{S} \models_{\mathsf{CPL}} (\pi \wedge \varphi_A) \rightarrow \bigwedge_{\ell \in L} \ell$ then
              for all $\ell \in L$ do
                if $\mathcal{T} \models_{\mathsf{K}_n} \ell \rightarrow [a]\bot$ or ($\mathcal{T} \not\models_{\mathsf{K}_n} \ell \rightarrow [a]\neg\ell$ and
                $\mathcal{T} \models_{\mathsf{K}_n} \ell \rightarrow [a]\ell$) then
                  $\mathcal{T}' := \mathcal{T}' \cup \{(\pi \wedge \varphi_A \wedge \ell) \rightarrow [a]\ell\}$
                /* preserve non-affected literals */
      $\mathcal{T}' := \mathcal{T}' \cup \{(\pi \wedge \varphi_A) \rightarrow \langle a\rangle\top\}$ /* safely add the law */
  $\mathcal{T}^{\star}_{\varphi \rightarrow \langle a\rangle\top} := \mathcal{T}'$
```

In our example, revising the action theory $\mathcal{T}$ with the executability law $\neg token \rightarrow \langle buy\rangle\top$ gives us $\mathcal{T}^{\star}_{\neg token \rightarrow \langle buy\rangle\top} =$

$$\left\{\begin{array}{c} coffee \rightarrow hot, token \rightarrow \langle buy\rangle\top, \\ \neg coffee \rightarrow [buy]coffee, \\ (\neg token \wedge \neg(\neg token \wedge coffee \wedge hot)\wedge \\ \neg(\neg token \wedge \neg coffee \wedge hot)\wedge \\ \neg(\neg token \wedge \neg coffee \wedge \neg hot)) \rightarrow [buy]\bot, \\ coffee \rightarrow [buy]coffee, hot \rightarrow [buy]hot, \\ (\neg token \wedge coffee \wedge hot) \rightarrow [buy]\neg token, \\ (\neg token \wedge \neg coffee \wedge hot) \rightarrow [buy]\neg token, \\ (\neg token \wedge \neg coffee \wedge \neg hot) \rightarrow [buy]\neg token, \\ \neg token \rightarrow \langle buy\rangle\top \end{array}\right\}$$

Again, the resulting theory can be post-processed to give us a much more compact representation of the new laws that have been added.

4.4 Complexity Issues

Algorithms 1–3 terminate. However, they come with a considerable computational cost: the K_n-entailment test with global axioms is known to be EXPTIME-complete. Computing all possible contexts allowed by the theory is clearly exponential. Moreover, the computation of $IP(.)$ might result in exponential growth [Marquis, 2000].

Given that theory change can be done offline, from the knowledge engineer's perspective what is more important is the size of the computed contracted theories. In that matter, our results are positive:

Theorem 3 *Let $\mathcal{T}$ be an action theory, and Φ be a law. Then the size (number of formulas) of $\mathcal{T}^{\star}_{\Phi}$ is linear in that of $\mathcal{T}$.*

5 Correctness of the Algorithms

Suppose we have two atoms p_1 and p_2, and one action a. Let $\mathcal{T}_1 = \{\neg p_2, p_1 \to [a]p_2, \langle a \rangle \top\}$. The only model of $\mathcal{T}_1$ is $\mathscr{M}$ in Figure 7. Revising such a model by $p_1 \vee p_2$ gives us the models $\mathscr{M}'_i$, $1 \leq i \leq 3$, in Figure 7. Now, revising $\mathcal{T}_1$ by $p_1 \vee p_2$ will give us $\mathcal{T}_1{}^{\star}_{p_1 \vee p_2} = \{p_1 \wedge \neg p_2, p_1 \to [a]p_2\}$. The only model of $\mathcal{T}_1{}^{\star}_{p_1 \vee p_2}$ is $\mathscr{M}'_1$ in Figure 7. This means that the semantic revision may produce models (viz. $\mathscr{M}'_2$ and $\mathscr{M}'_3$ in Figure 7) that are not models of the revised theories.

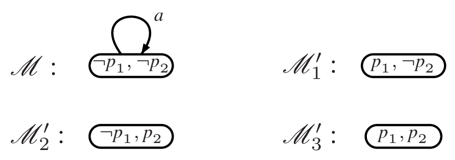


Figure 7: Model $\mathscr{M}$ of $\mathcal{T}_1$ and revision of $\mathscr{M}$ by $p_1 \vee p_2$.

The other way round the algorithms may give theories whose models do not result from revision of models of the initial theory: let $\mathcal{T}_2 = \{(p_1 \vee p_2) \to [a]\bot, \langle a \rangle \top\}$. Its only model is $\mathscr{M}$ (Figure 7). Revising $\mathscr{M}$ by $p_1 \vee p_2$ is as above. But $\mathcal{T}_2{}^{\star}_{p_1 \vee p_2} = \{p_1 \vee p_2, (p_1 \vee p_2) \to [a]\bot\}$ has a model $\mathscr{M}'' = \langle \{\{p_1, p_2\}, \{p_1, \neg p_2\}, \{\neg p_1, p_2\}\}, \emptyset \rangle$ not in $\mathscr{M}^{\star}_{p_1 \vee p_2}$.

All this happens because the possible states are not completely characterized by the static laws. Fortunately, concentrating on supra-models of $\mathcal{T}$, we get the right result.

Theorem 4 *If* $\mathcal{M} = \{\mathscr{M} : \mathscr{M}$ *is a supra-model of* $\mathcal{T}\}$ *and there is* $\mathscr{M}' \in \mathcal{M}$ *s.t.* $\models^{\mathscr{M}'} \Phi$, *then* $\bigcup_{\mathscr{M} \in \mathcal{M}} rev(\mathscr{M}, \Phi) \subseteq \mathcal{M}$.

Then, revision of models of $\mathcal{T}$ by a law Φ in the semantics produces models of the output of the algorithms $\mathcal{T}^{\star}_{\Phi}$:

Theorem 5 *If* $\mathcal{M} = \{\mathscr{M} : \mathscr{M}$ *is a supra-model of* $\mathcal{T}\} \neq \emptyset$, *then for every* $\mathscr{M}' \in \mathcal{M}^{\star}_{\Phi}$, $\models^{\mathscr{M}'} \mathcal{T}^{\star}_{\Phi}$.

Also, models of $\mathcal{T}^{\star}_{\Phi}$ result from revision of models of $\mathcal{T}$ by Φ:

Theorem 6 *If* $\mathcal{M} = \{\mathscr{M} : \mathscr{M}$ *is a supra-model of* $\mathcal{T}\} \neq \emptyset$, *then for every* $\mathscr{M}'$, *if* $\models^{\mathscr{M}'} \mathcal{T}^{\star}_{\Phi}$, *then* $\mathscr{M}' \in \mathcal{M}^{\star}_{\Phi}$.

Sticking to supra-models of $\mathcal{T}$ is not a big deal. We can use existent algorithms in the literature [Herzig and Varzinczak, 2007] to ensure that $\mathcal{T}$ is characterized by its supra-models and that $\mathcal{M} \neq \emptyset$.

6 Related Work

The problem of action theory change has only recently received attention in the literature, both in action languages [Baral and Lobo, 1997; Eiter *et al.*, 2005] and in modal logic [Herzig *et al.*, 2006; Varzinczak, 2008].

Baral and Lobo [1997] introduce extensions of action languages that allow for some causal laws to be stated as defeasible. Their work is similar to ours in that they also allow for weakening of laws: in their setting, effect propositions can be replaced by what they call defeasible (weakened versions of) effect propositions. Our approach is different from theirs in the way executability laws are dealt with. Here executability laws are explicit and we are also able to change them. This feature is important when the qualification problem is considered (cf. the Introduction).

The work by Eiter *et al.* [2005; 2006] is similar to ours in that they also propose a framework that is oriented to updating action laws. They mainly investigate the case where e.g. a new effect law is added to the description (and then has to be true in all models of the modified theory).

In Eiter *et al.*'s framework, action theories are described in a variant of a narrative-based action description language. Like in the present work, the semantics is also in terms of transition systems: directed graphs having arrows (action occurrences) linking nodes (configurations of the world). Contrary to us, however, the minimality condition on the outcome of the update is in terms of inclusion of sets of laws, which means the approach is more syntax oriented.

In their setting, during an update an action theory $\mathcal{T}$ is seen as composed of two pieces, $\mathcal{T}_u$ and $\mathcal{T}_m$, where $\mathcal{T}_u$ stands for the part of $\mathcal{T}$ that is not supposed to change and $\mathcal{T}_m$ contains the laws that may be modified. In our terms, when revising by a static law we would have $\mathcal{T}_m = \mathcal{S} \cup \mathcal{X}_a$, when revising by an effect law $\mathcal{T}_m = \mathcal{E}_a \cup \mathcal{X}_a$, and when revising with executability laws $\mathcal{T}_m = \mathcal{E}^{-}_a \cup \mathcal{X}_a$. The difference here is that in our approach it is always clear what laws should not change in a given type of revision, and $\mathcal{T}_u$ and $\mathcal{T}_m$ do not need to be explicitly specified prior to the update.

Their approach and ours can both be described as *constraint-based* update, in that the theory change is carried out relative to some restrictions (a set of laws that we want to hold in the result). In our framework, for example, all changes in the action laws are relative to the set of static laws $\mathcal{S}$ (and that is why we concentrate on supra-models: models of $\mathcal{T}$ having *val*$(\mathcal{S})$ as worlds). When changing a law, we want to keep the same set of states. The difference w.r.t. Eiter *et al.*'s approach is that there it is also possible to update a theory relatively to e.g. executability laws: when expanding $\mathcal{T}$ with a new effect law, one may want to constrain the change so that the action under concern is guaranteed to be executable in the result.[1] As shown in the referred work, this may require the withdrawal of some static law. Hence, in Eiter *et al.*'s framework, static laws do not have the same status as in ours.

7 Discussion and Perspectives

Here we have studied what revising action theories by a law means, both in the semantics and at the syntactical (algorithmic) level. We have defined a semantics based on distances between models that also captures minimal change w.r.t. the preservation of effects of actions. With our algorithms and the correctness results we have established the link between the semantics and the syntax for theories with supra-models. (Due to page limits, proofs have been omitted here.)

For the sake of presentation, here we have abstracted from the frame and ramification problems. However our definitions could have been stated in a formalism with a suitable solution to them, like e.g. Castilho *et al.*'s approach [1999]. With regards to the qualification problem, this is not ignored here:

[1]We can emulate that in our approach with two modifications of $\mathcal{T}$: first adding the effect law and then an executability law.

revising wrong executability laws is an approach towards its solution. Indeed, given the difficulty of stating all sufficient conditions for executability of an action, the knowledge engineer writes down some of them and lets the theory ‘evolve’ via subsequent revisions.

The reason why we have chosen such a simple notion of distance between models is that with other distances one may not always get the intended result. This is better illustrated with the *contraction counterpart* of our operators [Varzinczak, 2008]. Suppose one wants to remove an executability law $\varphi \rightarrow \langle a \rangle \top$. Then we do that by removing *a*-arrows from φ-worlds. Suppose we have a model with two φ-worlds, w_1 with one leaving *a*-arrow and w_2 with two *a*-arrows. Then with e.g. Dalal’s distance [1988], the associated contraction operator would always exclude the resulting model in which w_2 loses its two arrows, simply because deleting 1 arrow is Dalal-better than deleting 2. This problem doesn’t happen with our distance, which gives us a version of maxichoice [Hansson, 1999].

One criticism to the approach here developed concerns the precedence of static laws in the revision process, which could make the revision operators to be interpreted as incoherent. As agreed in the literature, however, given that static laws are much easier to state, they are more likely to be correct, and then it makes sense to give them precedence. Supporting this is the fact that most of the attention in the reasoning about actions area has been paid to effect laws and executability laws, which are much more difficult to completely specify. Our approach is in line with that.

Our next step is to analyze the behavior of our operators w.r.t. AGM-like postulates [Alchourrón *et al.*, 1985] for modal theories and the relationship between our revision method and contraction. What is known is that Levi identity [Levi, 1977], $\mathcal{T}^{\star}_{\Phi} = \mathcal{T}^{-}_{\neg\Phi} \cup \{\Phi\}$, in general does not hold for action laws. The reason is that up to now there is no contraction operator for $\neg\Phi$ where Φ is an action law. Indeed this is the general contraction problem for action theories: contraction of a theory $\mathcal{T}$ by a general formula (like $\neg\Phi$ above) is still an open problem in the area. The definition of a general method will mostly benefit from the semantic modifications we studied here (addition/removal of arrows and worlds).

Given the relationship between modal logics and description logics, a revision method for DL TBoxes [Baader *et al.*, 2003] would also benefit from the constructions that we have defined here.

References

[Alchourrón *et al.*, 1985] C. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *J. of Symbolic Logic*, 50:510–530, 1985.

[Baader *et al.*, 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *Description Logic Handbook*. Cambridge University Press, 2003.

[Baral and Lobo, 1997] C. Baral and J. Lobo. Defeasible specifications in action theories. In *Proc. IJCAI*, pages 1441–1446, 1997.

[Burger and Heidema, 2002] I.C. Burger and J. Heidema. Merging inference and conjecture by information. *Synthese*, 131(2):223–258, 2002.

[Castilho *et al.*, 1999] M. 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.

[Dalal, 1988] M. Dalal. Investigations into a theory of knowledge base revision: preliminary report. In *Proc. AAAI*, pages 475–479, 1988.

[Eiter *et al.*, 2005] T. Eiter, E. Erdem, M. Fink, and J. Senko. Updating action domain descriptions. In *Proc. IJCAI*, pages 418–423, 2005.

[Eiter *et al.*, 2006] T. Eiter, E. Erdem, M. Fink, and J. Senko. Resolving conflicts in action descriptions. In *Proc. ECAI*, pages 367–371, 2006.

[Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, 1988.

[Hansson, 1999] S. Hansson. *A Textbook of Belief Dynamics: Theory Change and Database Updating*. Kluwer, 1999.

[Herzig and Rifi, 1999] A. Herzig and O. Rifi. Propositional belief base update and minimal change. *Artificial Intelligence*, 115(1):107–138, 1999.

[Herzig and Varzinczak, 2007] A. Herzig and I. Varzinczak. Metatheory of actions: beyond consistency. *Artificial Intelligence*, 171:951–984, 2007.

[Herzig *et al.*, 2006] A. Herzig, L. Perrussel, and I. Varzinczak. Elaborating domain descriptions. In *Proc. ECAI*, pages 397–401, 2006.

[Katsuno and Mendelzon, 1992] H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In *Belief revision*, pages 183–203. Cambridge, 1992.

[Levi, 1977] I. Levi. Subjunctives, dispositions and chances. *Synthese*, 34:423–455, 1977.

[Marquis, 2000] P. Marquis. Consequence finding algorithms. In *Alg. for Defensible and Uncertain Reasoning*, pages 41–145. 2000.

[Parikh, 1999] R. Parikh. Beliefs, belief revision, and splitting languages. In *Logic, Language and Computation*, pages 266–278, 1999.

[Popkorn, 1994] S. Popkorn. *First Steps in Modal Logic*. Cambridge University Press, 1994.

[Quine, 1952] W. V. O. Quine. The problem of simplifying truth functions. *American Mathematical Monthly*, 59:521–531, 1952.

[Varzinczak, 2008] I. Varzinczak. Action theory contraction and minimal change. In *Proc. KR*, pages 651–661, 2008.

[Winslett, 1988] M.-A. Winslett. Reasoning about action using a possible models approach. In *Proc. AAAI*, pages 89–93, 1988.