Luis Fariñas del Cerro
Andreas Herzig   Jérôme Mengin (Eds.)

Logics in Artificial Intelligence

13th European Conference, JELIA 2012
Toulouse, France, September 26-28, 2012
Proceedings

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Luis Fariñas del Cerro
Université de Toulouse
Institut de Recherche en Informatique de Toulouse
118 route de Narbonne
31062 Toulouse Cedex 9, France
E-mail: luis.farinas@irit.fr

Andreas Herzig
Université de Toulouse
Institut de Recherche en Informatique de Toulouse
118 route de Narbonne
31062 Toulouse Cedex 9, France
E-mail: andreas.herzig@irit.fr

Jérôme Mengin
Université de Toulouse
Institut de Recherche en Informatique de Toulouse
118 route de Narbonne
31062 Toulouse Cedex 9, France
E-mail: jerome.mengin@irit.fr

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-33352-1 e-ISBN 978-3-642-33353-8
DOI 10.1007/978-3-642-33353-8
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012946470

CR Subject Classification (1998): I.2.2-4, I.2.8-9, F.4.1, F.3.1, D.1.6, H.3.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

JELIA is the European Conference on Logics in Artificial Intelligence. The acronym actually stands for its French translation *Journées Européennes sur la Logique en Intelligence Artificielle*: the conference series started back in 1988 as a small workshop that was held in Roscoff, France. The theme of the workshop was the use of logic as a formal basis for theoretical and practical studies in artificial intelligence. Since then, the number of applications and their importance have grown significantly, and theory and methods of logic for artificial intelligence have evolved a lot. Many fields like theorem proving or belief revision have matured, while new domains such as description logic or answer set programming have emerged. As from the second meeting, JELIA has adopted English and has published its proceedings in Springer's LNAI series.

Over the last three decades, JELIA has been organized biennially in many European countries: three times in Germany, twice in the UK and Portugal, and once in the Netherlands, Italy, Spain, and Finland. This year JELIA finally returned to France, taking place in Toulouse, "*la ville rose*", September 26–28, 2012.

This volume contains the papers selected for presentation at JELIA 2012. Competition was very high this year. We received 107 submissions from 31 countries (97 regular papers and 10 system descriptions). Only 36 regular papers and 5 system descriptions were selected for inclusion in the proceedings. The program included three invited talks whose abstracts can be found below:

– Leila Amgoud and Philippe Besnard "Logical Limits of Dung's Abstract Argumentation Framework"
– Ulrich Furbach "Extensions of Hyper Tableaux"
– Wiebe van der Hoek "On Two Results in Contemporary Modal Logic: Local Definability and Succinctness"

Many people contributed to making JELIA 2012 a success. We would like to thank the authors of the 107 submitted papers, which were of high quality and covered a broad range of topics. We also would like to thank the PC members for their hard work, as well as all the additional experts who made it possible to achieve a thorough reviewing process within a rather short time frame. Thanks are also due to IRIT (Institut de Recherche en Informatique de Toulouse), CNRS (Centre National de la Recherche Scientifique), UPS (Université Paul Sabatier), and LEA IREP (French Spanish Laboratory for Advanced Studies in Information, Representation and Processing) for their financial support. A final word of thanks goes to the JELIA 2012 organizing committee, in particular to Véronique Debats and Sabyne Lartigue for their precious support.

September 2012

Luis Fariñas del Cerro
Andreas Herzig
Jérôme Mengin

Organization

Program Chairs

Luis Fariñas del Cerro
Andreas Herzig
Jérôme Mengin

Program Committee

Thomas Ågotnes
Natasha Alechina
José Júlio Alferes
Franz Baader
Philippe Balbiani
Peter Baumgartner
Salem Benferhat
Philippe Besnard
Richard Booth
Gerhard Brewka
Pedro Cabalar
James Delgrande
Marc Denecker
Hans van Ditmarsch
Barbara Dunin-Kęplicz
Florence Dupin de Saint-Cyr
Ulle Endriss
Esra Erdem
Michael Fisher
Laura Giordano
Lluis Godo
Wiebe van der Hoek
Tomi Janhunen
Tommi Junttila
Jérôme Lang
Nicola Leone
Thomas Lukasiewicz
Carsten Lutz
Pierre Marquis
Luís Moniz Pereira
Angelo Montanari
David Pearce
Henri Prade
Jussi Rintanen
Francesca Rossi
Chiaki Sakama
Ulrike Sattler
Torsten Schaub
Renate A. Schmidt
Steven Schockaert
Leon van der Torre
Toby Walsh
Dirk Walther
Frank Wolter
Stefan Woltran

Additional Referees

Mario Alviano
Ringo Baumann
Jonathan Ben-Naim
Meghyn Bienvenu
Davide Bresolin
Andrea Cali
Martin Caminada
Broes De Cat
Pierangelo Dell'Acqua
Dario Della Monica
Agostino Dovier
Wolfgang Dvorak
Sjur Dyrkolbotn
Marcin Dziubinski
Patricia Everaere
Guillaume Feuillade
Martin Gebser
Adita Ghose
Valentina Gliozzi
Ricardo Gonçalves
Víctor Didier Gutiérrez Basulto
The Anh Han
Ullrich Hustadt
Mark Kaminiski
George Katsirelos
Piotr Kaźmierczak
Mohammad Khodadadi
Matthias Knorr
Sébastien Konieczny
Thomas Krennwallner
Temur Kutsia
Frédéric Lardeux
Brian Logan
Marco Manna
Marco Maratea
Thomas Meyer
Manuel Ojeda-Aciego
Madalena Ortiz
Max Ostrowski
Erik Parmann
Stef De Pooter
Gian Luca Pozzato
Bryan Renne
Francesco Ricca

Olivier Roussel	Peter Schueller	Levan Uridia
Pietro Sala	Nicolas Schwind	Pierfrancesco Veltri
Frédéric Saubion	Michael Thomazo	Hanne Vlaeminck
Marius Schneider	Dmitry Tishkovsky	Yì Nicholas Wáng
Thomas Schneider	Dmitry Tsarkov	Michal Zawidzki

Organizing Committee

Florence Dupin de Saint-Cyr	Sylvie Doutre	Sabyne Lartigue
Damien Bigot	Luis Fariñas del Cerro	Jérôme Mengin
Pierre Bisquert	Andreas Herzig	Frédéric Moisan
Claudette Cayrol	Seif-eddine Kramdi	
Véronique Debats	Marie-Christine Lagasquie	

Invited Talks

Leila Amgoud and Philippe Besnard (IRIT-CNRS, University of Toulouse, France), ***Logical Limits of Dung's Abstract Argumentation Framework***

A Dung's abstract argumentation framework takes as input a set of arguments and a binary relation encoding attacks between these arguments, and returns arguments gathered in some so-called extensions. General indications lack on how to instantiate this setting from a logical formalism, i.e., how to build arguments from a given *logical* knowledge base and how to choose an appropriate attack relation. This leads in some cases to undesirable results like inconsistent extensions (i.e., the set of logical formulas underlying an extension is inconsistent). This is due to the gap between the abstract setting and the knowledge base from which it is specified.

We first propose to fill in this gap by extending Dung's framework. The idea is to consider all the ingredients involved in an argumentation problem. We start with the notion of an abstract monotonic logic which consists of a language (defining the formulas) and a consequence operator. We show how to build, in a systematic way, arguments from a knowledge base formalised in such a logic.

When starting from a logical knowledge base, this takes care of defining *the* arguments. As evidenced by the literature, it often happens that people take a *syntax-based* subset of the arguments and a specific attack relation to form an argumentation framework that they claim to capture the argumentative information represented in the logical knowledge base. We show that such need not be the case, in particular with the mostly overrated undercut relation.

Ulrich Furbach (Department of Computer Science, University of Koblenz-Landau, Germany), ***Extensions of Hyper Tableaux***

At JELIA 1996 Hyper Tableaux were introduced as a first order calculus which combined ideas from hyper resolution and tableaux calculi. The first part of this talk reviews a number of extensions, which are implemented in the prover E-KRHyper. One of them incorporates efficient equality handling by the use of an adapted version of the well known superposition inference rule. Other extensions include a form of negation as failure, PROLOG-like data structures and arithmetic and a unique name assumption. By using a transformation from the description logic $\mathcal{SHIQ}$ to DL-clauses the prover E-KRHyper can also be used as a decision procedure for $\mathcal{SHIQ}$. The second part of the talk depicts the embedding of E-KRHyper within the natural language question answering system loganswer.de. We discuss the requirements which stem from such a time critical and knowledge intensive application, and we discuss how such a system can be evaluated.

Wiebe van der Hoek (Department of Computer Science, University of Liverpool, UK) ***On Two Results in Contemporary Modal Logic: Local Definability and Succinctness***

In this invited talk, I present two kinds of results and methods in modal logic. The first concerns *local definability*, and is joint work with Hans van Ditmarsch and Barteld Kooi. In modal logic, when adding a syntactic property to an axiomatisation, this property becomes true in all models, in all situations, under all circumstances. For instance, adding a property like $K_a p \to K_b p$ (agent b knows at least what agent a knows) to an axiomatisation of some epistemic logic has as an effect that such a property becomes *globally* true, i.e., it will hold in all states, at all time points (in a temporal setting), after every action (in a dynamic setting) and after any communication (in an update setting), and every agent will know that it holds, it will even be common knowledge. We propose a way to express that a property like the above only needs to hold *locally*: it may hold in the actual state, but not in all states. We achieve this by adding relational atoms to the language that represent (implicitly) quantification over all formulas, as in $\forall p(K_a p \to K_b p)$. We show how this can be done for a rich class of modal logics and a variety of syntactic properties.

The second theme concerns that of succinctness, and is joint work with Tim French, Petar Iliev and Barteld Kooi. One way of comparing knowledge representation formalisms is in terms of *representational succinctness,* i.e., we can ask whether one of the formalisms allows for a more 'economical' encoding of information than the other. Proving that one language is more succinct than another becomes harder when the underlying semantics is stronger. We propose to use *Formula Size Games* (as put forward by Adler and Immerman), games that are played on two *sets* of models, and that directly link the *length* of play with the *size* of the formula. Using Formula Size Games, we prove the following succinctness results for m-dimensional modal logic: (1) on general Kripke models , a notion of 'everybody knows' makes the resulting language exponentially more succinct for $m > 1$; (2) on epistemic models, the same language becomes more succinct for $m > 3$, (3) the results for the language with 'everybody knows' also hold of a language with 'somebody knows', and (4) on epistemic models, Public Announcement Logic is exponentially more succinct than epistemic logic, if $m > 3$. The latter settles an open problem raised by Lutz.

Table of Contents

Preferential Semantics for the Logic of Comparative Similarity over Triangular and Metric Models

Régis Alenda and Nicola Olivetti

Aix-Marseille Université, CNRS, LSIS UMR 7296, 13397, Marseille, France
regis.alenda@lsis.org, nicola.olivetti@univ-cezanne.fr

Abstract. The logic of Comparative Similarity CSL (introduced by Sheremet, Tishkovsky, Wolter and Zakharyaschev in 2005) allows one to reason about distance comparison and similarity comparison within a modal language. The logic can express assertions of the kind "A is closer/more similar to B than to C" and has a natural application to spatial reasoning, as well as to reasoning about concept similarity in ontologies. The semantics of CSL is defined in terms of models based on different classes of distance spaces. In this work we consider the cases where the distance satisfies the triangular inequality and the one where it is a metric. We show that in both cases the semantics can be equivalently specified in terms of preferential structures. Finally, we consider the relation of CSL with conditional logics and we provide semantics and axiomatizations of conditional logics over distance models with these properties.

1 Introduction

In a series of papers [1,2,3], Sheremet, Tishkovsky, Wolter and Zakharyaschev have presented several modal logics to reason about distance comparisons and topological properties. One of the most interesting is the logic $\mathcal{CSL}$, a modal logic of qualitative distance comparison, or a logic of comparative similarity between objects. This logic may be of interest and find an application in different areas of Knowledge representations, namely:

1. Spatial reasoning: this is the most obvious application, motivated by its distance semantics.
2. Reasoning about comparative similarity between concepts in ontologies (e.g. "Neoclassical Music is more similar to Baroque Music than to Romantic Music", see examples below).
3. Conditional reasoning: in this context *objects* to be compared become *possible worlds* and comparative similarity judgment become statements about *relative plausibility*: A is more plausible than /at least as plausible as B. It turns out that $\mathcal{CSL}$ is equivalent to some conditional logics.

To cope with the mentioned applications, the language of $\mathcal{CSL}$ must be suitably extended with further constructs specific to each context/application. However,

$\mathcal{CSL}$ encodes the core properties of qualitative similarity comparisons, so that it is worth to study it in its own.

The logic $\mathcal{CSL}$ corresponds to propositional logic extended with a modal binary operator $\leftleftarrows$ that is used to express distance comparisons between (set of) objects: a formula $A \leftleftarrows B$ denotes the set of objects w that are closer to A than to B, that is to say such that $d(w, A) < d(w, B)$, where d is the distance function. It covers in itself a wide range of logics of distance spaces; some of these systems or fragments thereof have been used to provide suitable logics for spatial reasoning in AI (see [3]). Topological notions of interior and closure of a set A can be represented,, as well as the **S5** existential modality. In case the underlying distance space satisfies some properties (namely *triangular inequality*) we obtain a precise embedding of the system $\mathbf{S4}_u$ into $\mathcal{CSL}$. We recall that $\mathbf{S4}_u$ (whence $\mathcal{CSL}$) strictly contains *RCC-8* region calculus. Quite surprisingly, in [3] it is shown that the powerful logic $\mathcal{QML}$, a kind of 'super logic' to reason about quantified distance relations can be encoded (although with an exponential blow-up) in $\mathcal{CSL}$ extended with the so-called realization operator $\textcircled{r}$ (that will not be considered here).

The logic $\mathcal{CSL}$ has also been investigated as a logic of qualitative similarity comparisons with possible applications in ontologies [1,2]. It can express qualitative comparisons such as: "Neoclassical Music is more similar to Baroque Music than to Romantic Music" (which is arguable), as well as definitions of concepts such as "Reddish ": a color which is more similar to a prototypical "Red" than to any other color [2]. In $\mathcal{CSL}$ this last example could be expressed by: Reddish $\equiv$ $\{$Red$\}$ $\leftleftarrows$ $\{$Green, . . . , Black$\}$. To develop the example about music a bit more, suppose KB contains:

$$\begin{aligned}
&(1)\ \text{Neoclassical} \sqsubseteq (\text{Baroque} \leftleftarrows \text{Romantic})\\
&(2)\ \text{Neoclassical} \sqsubseteq (\text{Romantic} \leftleftarrows \text{Contemporary})\\
&(3)\ \text{Modern} \equiv (\text{Contemporary} \sqcup \text{Romantic})
\end{aligned}$$

From this KB we can derive Neoclassical $\sqsubseteq$ (Baroque $\leftleftarrows$ Contemporary) and Neoclassical $\sqsubseteq$ (Baroque $\leftleftarrows$ Modern). Of course an application to similarity reasoning in ontologies, would require an extension of the language of $\mathcal{CSL}$ with nominals, roles, and possibly multiple operators of $\leftleftarrows$ to qualify comparisons.

The logic $\mathcal{CSL}$ is also strongly related to conditional logics based on Lewis' semantics [4,5,6]: a conditional $A{>}B$ is verified by a world x if B holds in all A-worlds closest/most similar to x. The conditional operator $>$ and $\leftleftarrows$ were observed to be interdefinable in minspace models [2,3] (see Sec. 5).

The properties of $\mathcal{CSL}$ depend on the properties of the distance function. In the weakest case the distance function is assumed to satisfy only the identity law: $d(x, y) = 0$ iff $x = y$. Further obvious properties are symmetry and triangular inequality. Another distinction is whether the distance function satisfies or not the so-called *minspace* principle: $d(x, A) = min\{d(x, y) \mid y \in A\}$, that is that $d(x, A)$ is always realized by an element in A. The authors in [3] have obtained axiomatizations of $\mathcal{CSL}$ over arbitrary, triangular and metric distance models; an axiomatization of $\mathcal{CSL}$ over minspace models is provided in [7], whereas an infinite axiomatization of $\mathcal{CSL}$ over symmetric minspace is provided in [8].

In [7] it has been shown that the semantics of $\mathcal{CSL}$ over minspaces can be equivalently restated in terms or preferential structures, the kind of structures used to define the semantics of conditional logics. A preferential structure is a set equipped by a family of binary relations indexed by the objects of the set (or equivalently a ternary relation). The relation $x \leq_y z$ can be interpreted intuitively as "y is at least as similar/close to x than z". Such a semantics was originally proposed by Lewis [4] to give a formal account of counterfactual conditionals and related notions. In [9], the correspondence between distance and preferential semantics has been extended to $\mathcal{CSL}$ interpreted over *symmetric* minspace models. Finally in [10] the correspondence has been extended to arbitrary non-minspace distance models. The semantics provided by preferential structures is close to ordinary modal semantics and it is better suited to study logical properties such axiomatization, finite model property; moreover it can be the base to develop proof systems (see [10,9,7]).

In this paper we aim to complete the picture by giving in Sec. 3 a preferential semantics counterpart for the remaining cases of $\mathcal{CSL}$ over distance models with triangular inequality as well as over metric models. We show by a filtration technique the finite model property with respect to triangular preferential models, a property that generally fails for distance models (except for minspace models). The finite model property is also conjectured for metric preferential models, yet it is still an open problem at this time. Unlike in previous work, the correspondence between distance and preferential models is not shown directly. Instead it is proven in Sec. 4 by a modular soundness and completeness result with respect to the preferential semantics of an axiomatization, which was already proven to be sound and complete with respect to distance models in [3].

In Sec. 5 we investigate the connection with conditional logics. In [2] the conditional and the comparative similarity connectives are shown to be interdefinable on minspace models. We extend this correspondence to all cases; it turns out that distance models provide a natural semantics for conditionals. More precisely, we characterize conditional logic corresponding to arbitrary, triangular and metric distance/preferential models. In all cases we provide a sound and complete axiomatization: whereas for the general case we (re)-discover well-known logic **VWU**, for the triangular and metric models we obtain new systems to the best of our knowledge.

Proof details are omitted due to space limitations and can be found in [11].

2 The Logic of Comparative Similarity $\mathcal{CSL}$

Let $\mathcal{V}_p = \{p_0, p_1, \ldots, p_i, \ldots\}$ be an enumerable set of *propositional variables.* We consider a propositional language $\mathcal{L}$, extended with the binary operator $\leftleftarrows$, and generated by: $A, B ::= \top \mid \bot \mid p_i \mid \neg A \mid A \sqcap B \mid A \sqcup B \mid A \leftleftarrows B$.

The original semantics of $\mathcal{CSL}$ is based on distance spaces [2]. A distance space is a pair (Δ, d) where Δ is a non-empty set, called the *domain* and whose elements are called *points*, and where $d : \Delta \times \Delta \mapsto \mathbb{R}^{\geq 0}$ is the *distance function* satisfying the following identity:

$$d(x,y) = 0 \text{ iff } x = y \ . \tag{ID}$$

We define the distance from a point $w \in \Delta$ to a set $X \subseteq \Delta$ by $d(w,X) = \inf \{d(w,x) \mid x \in X\}$. If X is empty, we let $d(w,X) = \infty$. We say that the distance $d(w,X)$ is *realized* iff there is a point $x \in X$ such that $d(w,X) = d(w,x)$. In this case the distance can be defined as a *minimum* instead of an *infimum.*

Additional properties can be assumed on d, such as the well known *triangular inequality* and *symmetry*, and the less well-known *minspace property*:

$$d(x,y) \leq d(x,z) + d(z,y) \ , \tag{TRI}$$

$$d(x,y) = d(y,x) \ , \tag{SYM}$$

$$X \neq \varnothing \text{ implies } d(w,X) \text{ is realized} \ . \tag{MIN}$$

When d satisfies (TRI) or (SYM), we call the distance space *triangular* or *symmetric* respectively. If it satisfies both, we call it a *metric* (distance) space. If it satisfies (MIN), we call it a *minspace.* Distance models are then defined as a kind of Kripke models based on a distance space.

Definition 1. *A distance model is a triple $\mathcal{M} = \langle \Delta, d, \pi \rangle$ where (Δ, d) is a distance space and $\pi \colon \mathcal{V}_p \mapsto 2^{\Delta}$ is the valuation function which maps every propositional variable $p_i \in \mathcal{V}_p$ to a* region *(subset) $\pi(p_i) \subseteq \Delta$.*

The interpretation $C^{\mathcal{M}}$ of a formula $C \in \mathcal{L}$ is as usual: $\top^{\mathcal{M}} = \Delta$, $\bot^{\mathcal{M}} = \varnothing$ and $p_i^{\mathcal{M}} = \pi(p_i)$; boolean operators denote set-theoretic boolean operations; the interpretation of $\leftleftarrows$ is shown below:

$$(A \leftleftarrows B)^{\mathcal{M}} = \left\{ w \in \Delta \;\middle|\; d(w, A^{\mathcal{M}}) < d(w, B^{\mathcal{M}}) \right\} \ .$$

Satisfiability and validity of formulas are defined as usual.

A distance model is a triangular (resp. metric, minspace) distance model iff it is based on a triangular (resp. metric, minspace) distance space.

Despite its apparent simplicity, the logic $\mathcal{CSL}$ turns out to be quite expressive [2,3]. The closure $\Diamond A = \{w \mid d(w,A) = 0\}$ of a set A, and its dual the interior $\Box A = \{w \mid d(w, \Delta - A) > 0\}$, can be expressed using the $\leftleftarrows$ operator. An **S5**-like existential modality $\exists$ (with the intended meaning that $(\exists A)^{\mathcal{M}} = \Delta$ iff $A^{\mathcal{M}} \neq \varnothing$), and its dual the universal modality $\forall$, can be defined as well:

$$\Diamond A \equiv \neg(\top \leftleftarrows A), \quad \Box A \equiv \top \leftleftarrows \neg A, \quad \exists A \equiv (A \leftleftarrows \bot), \quad \forall A \equiv \neg(A \leftleftarrows \bot). \tag{1}$$

Note that $\Diamond$ and $\Box$ do not behave like the **S4** modalities when interpreted over *arbitrary* distance spaces. Indeed, the topology induced by an arbitrary distance space is not a *topological space* – in particular Kuratowski's axiom $\Diamond\Diamond A \subseteq \Diamond A$ fails. On the contrary, triangular and metric distance models satisfy the aforementioned axiom and as consequence $\mathcal{CSL}$ interpreted over those two classes of models contains the full logic $\mathbf{S4}_u$.

The behavior of $\mathcal{CSL}$ over various classes of distance spaces is investigated in [2,3]. When the minspace (MIN) property is not assumed, $\mathcal{CSL}$ is indifferent to (SYM) (i.e., satisfiability wrt. arbitrary and symmetric distance models are equivalent). It is, on the contrary, sensible to triangular distances and to metrics.

When the minspace property is assumed, the situation drastically changes: $\mathcal{CSL}$ is unable to distinguish between arbitrary minspaces, symmetric minspaces, triangular minspaces and metric minspaces. However when only *finite* minspaces are considered, $\mathcal{CSL}$ is able to distinguish between symmetric and non-symmetric ones. Furthermore in the general case $\mathcal{CSL}$ does not have the finite model property, yet it does when the minspace property is assumed.

The satisfiability problem is shown to be decidable and ExpTime-complete for all these classes of models. However, for models based on subspaces of $\mathbb{R}^n$ with the usual euclidean distance, it turns out that the logic is undecidable [2,3].

3 A Preferential Semantics

$\mathcal{CSL}$ is a logic of pure qualitative comparisons. This motivates the study of an alternative relational, modal-like, semantics which abstracts away from the numerical values by encoding the distance comparisons by means of a family of total preorders $(\leq_w)_{w\in\Delta}$. This semantics is similar to the one that some conditional logic systems enjoy [6,4,5], and where the relations $\leq_w$ are called *preferential relations* whence its name.

In [10] it shown that the semantics of $\mathcal{CSL}$ over arbitrary distance spaces can be equivalently defined using preferential structures. Points of the preferential structure intuitively corresponds to regions (subsets) of the distance space, and comparisons of the form $x \leq_w y$ means "all the points in region w are as least as close to region x than to region y".

Definition 2. *An (arbitrary) preferential model is a triple $\mathcal{M} = \langle \Delta, (\leq_w)_w, \pi\rangle$ where Δ is a non empty-set and $(\leq_w)_w$ is a family of total preorders indexed by the elements of Δ, each one satisfying the* weak centering *property:*

$$\text{For all } x \in \Delta, \quad w \leq_w x \ . \tag{WKCT}$$

The valuation function π and the interpretation of propositional and boolean formulas are defined as in Definition 1. The interpretation of $\leftleftarrows$ is shown below[1]*:*

$$(A \leftleftarrows B)^{\mathcal{M}} = \left\{ w \in \Delta \;\middle|\; \text{there is } x \in A^{\mathcal{M}} \text{ such that } x <_w y \text{ for all } y \in B^{\mathcal{M}} \right\} \ .$$

We furthermore require that $\mathcal{M}$ satisfies the limit assumption*: if $A^{\mathcal{M}} \neq \varnothing$ then $A^{\mathcal{M}}$ has a $\leq_w$-minimal element.*

The weak centering states that each region w is minimal with respect to its own preferential relation. Given a formula A satisfiable in $\mathcal{M}$, the limit assumption assure the existence of a minimal region satisfying A with respect to w.

[1] We recall that $x <_w y$ iff $x \leq_w y$ and $y \not\leq_w x$.

In light of (1), the closure and interior of a set preferentially correspond to:

$$\begin{aligned} \Diamond A &= \{w \mid \text{there exists } x \in A \text{ such that } x \leq_w w\} \ , \\ \Box A &= \{w \mid \text{for all } x \in \Delta,\ \ x \leq_w w \text{ implies } x \notin A\} \ . \end{aligned} \tag{2}$$

In [10] arbitrary distance and preferential models have been shown to be equivalent for formulas, in the sense that that the two semantics define the same set of valid formulas. It is also shown that the finite model property holds for arbitrary preferential models, whereas it obviously fails for distance ones[2].

In previous work, we have investigated how the additional properties of the distance function translate into the preferential semantics. In [7,9], we have shown that minspace models can be captured by the *strict centering* property:

$$w = x \text{ or } w <_w x. \tag{SCT}$$

The symmetric minspace case, not considered here, can also be captured [9,8].

We now extend those results to the remaining cases of triangular distance spaces and metric spaces. The correspondence with distance models will be given by Theorem 4 in the next section, through the axiomatization.

3.1 Triangular Case

Suppose that a point x is on the closure of a set A, that is $d(x, A) = 0$. When d satisfies the triangular inequality, we have that $d(w, A) \leq d(w, x) + d(x, A)$, that is $d(w, A) \leq d(w, x)$, for any point w. Intuitively, this means that we cannot be strictly closer to the closure of a set than to the set itself [3].

This property can be nicely translated in the preferential semantics. We say that a preferential model is a triangular preferential model iff it satisfies the following property for all $x, y, w \in \Delta$:

$$x \leq_y y \text{ implies } x \leq_w y. \tag{P-TR}$$

As in the general case [10], we can show by a filtration technique that $\mathcal{CSL}$ has the finite model property with respect to triangular preferential model[3].

Theorem 1. *If a formula C is satisfiable in a triangular preferential model, then it is satisfiable in a finite preferential model (of exponential size).*

3.2 Metric Case

Let (Δ, d) be a metric space, and consider a subset $X \subseteq \Delta$ and a point $w \in \Delta$ such that $d(w, X) = 0$ (that is w is in the closure of the set X). Let $A, B \subseteq \Delta$ and suppose that for all $x \in X$, $d(x, A) \leq d(x, B)$. That is $d(x, B) - d(x, A) \geq 0$.

For any $x \in X$, we have $d(w, A) \leq d(w, x) + d(x, A)$ and $d(w, B) \geq d(x, B) - d(x, w)$ by the triangular inequality. We then obtain $d(w, B) - d(w, A) \geq d(x, B) - d(x, A) - 2d(w, x)$ since d is symmetric. Now let x range over X and take the

[2] e.g., The formula $A \sqcap \Diamond\neg A$ is not satisfiable in any finite distance model.

[3] The finite model property does not hold for triangular *distance* models.

infimum: we obtain $d(X, A) \leq d(X, B)$. As $d(w, X) = 0$, we get $d(w, B) - d(w, A) \geq d(x, B) - d(x, A) - 0 \geq 0$, that is $d(w, A) \leq d(w, B)$.

This shows that in a metric space, every point have to satisfy all the non-strict inequalities that holds in its 'immediate neighborhood' [3]. This property translates nicely to preferential models: a *triangular* preferential model is a metric preferential model iff it satisfies, for all $x, y \in \Delta$:

$$x \leq_y y \text{ implies } \leq_x \subseteq \leq_y . \tag{P-MT}$$

Observe that (P-TR) and (P-MT) are independent, and that we have to take both to obtain a metric preferential model[4] – a metric model being a particular case of a triangular model. Whether the finite model property holds for $\mathcal{CSL}$ over metric preferential models is still open – we strongly conjecture that it is the case.

4 Axiomatization

We present here a reformulation of the axiomatization of $\mathcal{CSL}$ over arbitrary distance spaces that can be found in [3][5]. The axioms schemes, in addition to the ones of propositional logic, are shown below.

$$\begin{array}{lll}
\neg(A \leftleftarrows C) \sqcap \neg(C \leftleftarrows B) \to \neg(A \leftleftarrows B) & \textsf{(AX3)} & \top \leftleftarrows \bot \qquad \textsf{(AX1)} \\
(A \leftleftarrows B) \sqcap (A \leftleftarrows C) \to (A \leftleftarrows (B \sqcup C)) & \textsf{(AX5)} & \neg(A \leftleftarrows B) \sqcup \neg(B \leftleftarrows A) \qquad \textsf{(AX2)} \\
(A \leftleftarrows \bot) \to \neg(\neg(A \leftleftarrows \bot) \leftleftarrows \bot) & \textsf{(AX6)} & A \to \neg(\top \leftleftarrows A) \qquad \textsf{(AX4)} \\
\dfrac{\vdash A \to B}{\vdash (A \leftleftarrows C) \to (B \leftleftarrows C)} & \textsf{(Mon)} &
\end{array}$$

Intuitively, (AX1) states that we are strictly closer to the whole set than to the empty set. (AX2) says that two formulas can always be (non-strictly) compared. (AX3) states the transitivity of non-strict comparisons. (AX4) encodes the weak centering property. (AX5) states that if we are strictly closer to one set than to two others, then we are strictly closer to this set than to the union of the two. (AX6) is the axiom **5** of modal logic $\exists A \to \forall\exists A$. Finally, the rule (Mon) states the monotonicity of the comparative similarity operator in its first argument.

To capture the minspace, triangular and the metric case, we consider the following additional axioms (see [7,3]).

$$\begin{array}{ll}
A \to (\top \leftleftarrows \neg A) \quad \textsf{(AXMS)} & \neg(\neg(\top \leftleftarrows A) \leftleftarrows A) \quad \textsf{(AXTR)} \\
 & (A \leftleftarrows B) \to (\top \leftleftarrows \neg(A \leftleftarrows B)) \quad \textsf{(AXMT)}
\end{array}$$

[4] Note that (P-MT) does not correspond to symmetry alone: it is a consequence of an interaction between both the triangular inequality and the symmetry.

[5] The axiomatization in [3] makes use of the additional ⓡ modality. However, when the axioms involving ⓡ are ignored, it is equivalent to the axiomatization we present. Soundness and completeness results with respect to distance semantics still hold.

Axiom (AXMS), which can be rewritten as $A \to \Box A$, encodes the strict centering property (SCT). Axiom (AXTR) can be rewritten as $\neg(\Diamond A \Leftarrow A)$ and corresponds to the triangular property (P-TR). Axiom (AXMT) states $(A \Leftarrow B) \to \Box(A \Leftarrow B)$ and encodes the metric property (P-MT).

We define the systems HCSLMS=HCSL+(AXMS), HCSLTR=HCSL+(AXTR), and HCSLMT=HCSL+(AXTR)+(AXMT). Soundness and completeness of the axiomatizations with respect to the distance semantics were proven in [3] for the general, triangular and metric cases; and in [7] for the minspace case.

Theorem 2 ([3,7]). *A formula is derivable in* HCSL*,* HCSLMS*,* HCSLTR *or* HCSLMT *respectively, iff it is valid in every arbitrary, minspace, triangular or metric distance model respectively.*

We now show that those systems are sound and complete with respect to the preferential semantics as well[6].

Theorem 3. *A formula is derivable in* HCSL*,* HCSLMS*,* HCSLTR *or* HCSLMT *respectively, iff it is valid in every arbitrary, minspace, triangular or metric preferential model respectively.*

Proof. ($\Rightarrow$) One can check that every axiom of HCSL is valid in every preferential model, that the rule (Mon) preserve validity, and that (AXMS), (AXTR) and (AXMT) are valid in their respective classes of models.

($\Leftarrow$) We give a modular completeness proof of $\mathsf{HCSL} + X$, where X is either empty (general case), (AXMS) (minspace case), (AXTR) (triangular case), (AXTR)+(AXMT) (metric case). As usual, we prove completeness by exhibiting for every formula C that is consistent with respect to $\mathsf{HCSL} + X$, a *canonical model* (built upon *maximal consistent sets*) in which C is satisfiable.

Maximal consistent sets are defined as usual. We recall that any consistent set T can be extended to a maximal consistent set $S \subseteq T$.

Definition 3. *Let* $S, T \in \text{MaxCons}$ *and* $A, B \in \mathcal{L}$*. We define:*
1. $A \preceq_S B$ *iff* $\neg(B \Leftarrow A) \in S$*,*
2. $S\,\mathsf{R}\,T$ *iff* $(A \Leftarrow \bot) \in S$ *for all* $A \in \mathcal{L}$*,*
3. $S^A = \{\neg B \mid (A \Leftarrow B) \in S\}$*.*

The relation $\preceq_S$ is an order induced by (negative) comparative similarity formulas, comparing the relative closeness of *formulas* with respect to S. R is an *accessibility relation* that connect sets that are 'compatible' with each other, in the sense that both sets agree upon existential formulas of the form $\exists A$. The set S^A is introduced for technical reason. Intuitively, if $S^A \subseteq T$ then T should be a 'minimal' element of A for S.

Lemma 1. *Let* $S \in \text{MaxCons}$ *and* $A \in \mathcal{L}$*. We have the following:*
1. $\preceq_S$ *is a total preorder over* $\mathcal{L}$*.*
2. R *is an equivalence relation over* MaxCons*.*

[6] In the minspace case, this was already proven in [7].

3. *If A is consistent and $(A \leftleftarrows \bot) \in S$, then there exists $T \in \mathrm{MaxCons}$ such that $S^A \cup \{A\} \subseteq T$ and $S \,\mathsf{R}\, T$.*

Let C be a consistent formula; there exists a maximal consistent set $S^* \in \mathrm{MaxCons}$ such that $C \in S^*$. To build our canonical model, we pick the maximal consistent sets that are 'compatible' with S^* (i.e., in relation wrt. R).

Definition 4. *Let $S^* \in \mathrm{MaxCons}$. The* canonical model generated by S^* *(wrt. $\mathsf{HCSL} + X$) is a triple $\mathcal{M} = \langle \Delta, (\leq_S)_{S \in \Delta}, \pi \rangle$ defined as follows:*
- *$\Delta = \{S \mid S^* \,\mathsf{R}\, S\}$,*
- *for all $S, T, U \in \Delta$, we let $T \leq_S U$ iff for all $A \in T$, there exists $B \in U$ such that $A \preceq_S B$.*
- *$\pi(p_i) = \{S \in \Delta \mid p_i \in S\}$, for all propositional variable $p_i \in \mathcal{V}_p$.*

One can check that $\mathcal{M}$ is a preferential model (transitivity, totality and weak centering of each $\leq_S$ follow from (AX3), (AX2) and (AX4) respectively; the *limit assumption* will be shown later). In [7], (AXMS) was shown to force (SCT). We now show that (AXTR) and (AXMT) force (P-TR) and (P-MT) respectively.

Lemma 2. *If axioms (AXMS), or (AXTR), or (AXMT) are in X, then the canonical model with respect to $\mathsf{HCSL} + X$ satisfies the strict centering, or the triangular property, or the metric property respectively.*

Proof. **Axiom (AXTR):** Let $T, U \in \Delta'$ be such that $T \leq_U U$. Due to (AX4), observe that each formula in U is minimal for $\preceq_U$. As $\top \in U$ and by definition of $\leq_U$, we conclude that $A \preceq_U \top$, i.e., $\Diamond A \in U$, for all $A \in T$. Now take any $S \in \Delta'$. By (AXTR) we get that $A \preceq_S \Diamond A$ for all $A \in T$. We conclude by the definition of $\leq_S$ and the fact that $\Diamond A \in U$.

Axiom (AXMT): Let $T, U \in \Delta$ such that $T \leq_U U$. Then $\neg(\top \leftleftarrows A) \in U$ for every $A \in T$ by the definition of the preferential relation. In particular, we have that $\neg(\top \leftleftarrows \neg(A \leftleftarrows B)) \in U$ for every $\neg(A \leftleftarrows B) \in T$. By the contrapositive of (AXMT), we then obtain that $\neg(A \leftleftarrows B) \in U$ for every $\neg(A \leftleftarrows B) \in T$, and the inclusion $\leq_T \subseteq \leq_U$ follows. □

Lemma 3. *For all formula $A \in \mathcal{L}$, let $||A|| = \{S \in \Delta \mid A \in S\}$.*
1. *If $S^A \subseteq T$ then T is a $\leq_S$-minimal element of $||A||$.*
2. *For all formulas $F \in \mathcal{L}$, $F^{\mathcal{M}} = ||F||$.*

By the previous Lemma, we finally obtain that $\mathcal{M}$ satisfies the limit assumption. As $C \in S^*$, we conclude that C is satisfiable in $\mathcal{M}$. □

From Theorems 2 and 3, we obtain the equivalence between the preferential and the distance semantics, in particular for the new triangular and metric cases.

Theorem 4. *A formula is satisfiable in an arbitrary, minspace, triangular or metric distance model, iff it is satisfiable in an arbitrary, minspace, triangular or metric preferential model respectively.*

5 Link with Conditional Logics

Conditional logics have a long history [6,4,5]. First proposed to formalize some kind of hypothetical (counterfactual) reasoning, they have found various applications, from non-monotonic reasoning to belief update and revision.

The semantics of a conditional $A{>}B$ is defined in terms of possible-world models, where the intuition is that $A{>}B$ is true at a world x, if B is true at all A-worlds that are 'most similar' or 'closest' to x. In particular Lewis, in his pioneering work [4] has proposed to capture the semantics of counterfactual conditionals in two related ways: (i) by means of sphere-models, (ii) by a ternary accessibility relation expressing comparative similarity, the same as in our preferential semantics. In the latter case, the interpretation of $>$ in a preferential model $\mathcal{M} = \langle \Delta, (\leq_w)_{w\in\Delta}, \pi\rangle$ (with the limit assumption) is given by:

$$(A{>}B)^{\mathcal{M}} = \big\{w \mid \min_{\leq_w}(A^{\mathcal{M}}) \subseteq B^{\mathcal{M}}\big\}$$

As for $\mathcal{CSL}$, different properties of the preferential relations $(\leq_w)_{w\in\Delta}$ give rise to different conditional logic's systems.

The link between the comparative similarity and the conditional operators was first noted in [2,3] for minspaces models, where the authors remark that when the preferential relations are induced by a distance (that is $x \leq_w y$ iff $d(w,x) \leq d(w,y)$)[7], then the comparative similarity operator and the conditional operator are *interdefinable* in the following way:

$$\begin{aligned}(A \leftleftarrows B) &\equiv \neg(A{>}\bot) \sqcap (A \sqcup B{>}\neg B)\\ (A{>}B) &\equiv \neg(A \leftleftarrows \bot) \sqcup (A \leftleftarrows A \sqcap \neg B)\end{aligned} \tag{3}$$

It follows that when interpreted over minspaces, $\mathcal{CSL}$ is equivalent to a conditional logic (namely the logic **VCU** [6,4,5]).

By means of the preferential semantics and Theorem 4, we can extend this result to other classes of distance spaces: indeed one can check that (3) holds in all the preferential models of Sec. 3. Therefore $\mathcal{CSL}$ interpreted over any class of distance space considered here is a conditional logic "in disguise". We now characterize the conditional systems generated by each class of distance models.

Arbitrary preferential models of Definition 2 (corresponding to arbitrary distance models) characterize the conditional logic **VWU**; minspace preferential models, obtained by adding (SCT), define the logic **VCU** [4,5,12,6]. Triangular and metric preferential models do not seem to have been considered in the literature, hence the conditional logics they define, to which we refer as **VWUTR** and **VWUMT** respectively, are new to the best our knowledge.

[7] This direct translation between distance and preferential models only works in the minspace case, see [10].

From the axiomatic point of view, we show below an axiomatization of **VWU** [6,4,5] (in addition to axioms and rules of classical propositional calculus):

$$(A{>}A) \qquad \text{(ID)}$$

$$(A{>}B) \to (A \to B) \qquad \text{(MP)} \qquad (A{>}B) \wedge (A{>}C) \to ((A \wedge B){>}C) \qquad \text{(CM)}$$

$$\neg(A{>}\bot) \to ((A{>}\bot){>}\bot) \qquad \text{(U1)} \qquad (A{>}C) \wedge (B{>}C) \to (A \vee B{>}C) \qquad \text{(CA)}$$

$$(A{>}\bot) \to (\neg(A{>}\bot){>}\bot) \qquad \text{(U2)} \qquad (A{>}B) \wedge \neg(A{>}C) \to (A \wedge C{>}B) \qquad \text{(CV)}$$

$$\frac{\vdash A \leftrightarrow B}{\vdash (A{>}C) \leftrightarrow (B{>}C)} \ \text{(RCEA)} \qquad \frac{\vdash (B_1 \wedge \ldots \wedge B_n) \to C}{\vdash ((A{>}B_1) \wedge \ldots \wedge (A{>}B_n)) \to (A{>}C)} \ \text{(RCK)}$$

Additional properties of the preferential relations are captured by adding further axioms. In particular, we consider the following three[8]:

$$(A \wedge B) \to (A{>}B) \qquad \text{(CS)} \qquad (\neg(\top{>}\neg A){>}\neg A) \to (A{>}\bot) \qquad \text{(TR)}$$

$$(A{>}B) \to (\top{>}(A{>}B)) \qquad \text{(MT)}$$

Axiom (CS) is well-known: it encodes the strict centering and is part of **VCU**. Axiom (TR) and (MT) respectively correspond to the triangular and to the metric preferential properties. Whereas axiom (TR) seems to be new to the best of our knowledge, it is worth noting that axiom (MT) is part of the axiomatization of the logic BCR in [13] for representing AGM belief revision.

We let **VCU** = **VWU**+(CS), **VWUTR** = **VWU**+(TR), and **VWUMT** = **VWU**+(TR)+(MT). By means of the equivalence with $\mathcal{CSL}$, or by a direct canonical model construction similar to the one in Sec. 4 (such as in [14]), one can show that each of these axiomatizations is sound and complete with respect to their respective class of preferential models.

Theorem 5. *A formula is derivable in* **VWU**, **VCU**, **VWUTR** *or* **VWUMT** *respectively, iff it is valid in every arbitrary, minspace, triangular or metric preferential model respectively.*

In light of the correspondence result of Theorem 4 between the preferential semantics and the distance one (wrt. $\mathcal{CSL}$), one can wonder whether we can directly give a semantics of the conditional operator in terms of distances spaces.

Given a distance model $\mathcal{M} = \langle \Delta, d, \pi \rangle$ (see Def. 1), we define:

$$w \models (A{>}B) \text{ iff } A^{\mathcal{M}} = \varnothing \text{ or } d(w, A^{\mathcal{M}}) < d(w, (A \sqcap \neg B)^{\mathcal{M}}) \ . \qquad (4)$$

One can check that with this semantics, (3) now also holds in distance models. Moreover, as a consequence of Theorems 2 and 3 and interdefinability of $\leftleftarrows$ and $>$, we obtain the soundness and completeness of **VWU** and its variants with respect to their respective classes of distance models:

[8] In light of (2), the conditional formulas corresponding to the topological and existential modalities are: $\Diamond A \equiv \neg(\top{>}\neg A)$, $\Box A \equiv (\top{>}A)$, $\exists A \equiv \neg(A{>}\bot)$, $\forall A \equiv (\neg A{>}\bot)$.

Theorem 6. *A formula is derivable in* **VWU**, **VCU**, **VWUTR** *or* **VWUMT** *respectively, iff it is valid in every arbitrary, minspace, triangular or metric* distance *model respectively.*

This semantics seems similar to the semantics for conditionals *without the limit assumption* (see, e.g., [12,4,15]) which can be transposed to distance models.

$$w \models (A{>}B) \text{ iff } A^{\mathcal{M}} = \varnothing \text{ or there exists } x \in A^{\mathcal{M}} \text{ such that} \\ \text{for all } y \in A^{\mathcal{M}}, d(w,y) \leq d(w,x) \text{ implies } y \in B^{\mathcal{M}}. \tag{5}$$

However, those two semantics are different. Consider the following model:

$$\Delta = \{w, x, y_i \mid i \in \mathbb{N}\}, \qquad p^{\mathcal{M}} = \{x, y_i \mid i \in \mathbb{N}\}, \qquad q^{\mathcal{M}} = \{x\}, \\ d(w,x) = 1, \qquad d(w,y_i) = 1 + \frac{1}{1+i}.$$

Using (5) yields to $w \in (p{>}q)^{\mathcal{M}}$ (as x is a minimal element of $p^{\mathcal{M}}$ and it satisfies q). However when we consider (4), we have $d(w, p^{\mathcal{M}}) = d(w, (p \sqcap \neg q)^{\mathcal{M}}) = 1$ and hence $w \notin (p{>}q)^{\mathcal{M}}$. Intuitively, (4) takes 'limit points' (in this example the one of all of the y_i) into accounts when defining the truth value of a conditional, whereas (5) is 'blind' to them.

Hence (5) does not preserve the interdefinability of $\leftleftarrows$ and $>$ over distance models. Moreover, (5) always gives **VCU** whatever properties of the distance are assumed (due to the property (ID) that in this case works like the strict centering). When the minspace property is assumed, (4) and (5) become equivalent.

Preferential relations induced by a distance have been previously considered in the literature (e.g., [4,16,17]). However to our knowledge, this idea was not pushed to an end to define a semantics of conditional in terms of distance models.

Whether the above mentioned variants of **VCU** matter (in particular for modeling belief change) is open and might be further investigated.

6 Conclusion and Further Work

In this paper, we have extended the work of [8,10,11] by giving a preferential semantics for $\mathcal{CSL}$ over triangular and metric models. This complete the picture and each kind of distance semantics for $\mathcal{CSL}$ has now its own preferential counterpart. We have proved the equivalence between the preferential and the distance semantics by means of a modular completeness result of the known axiomatizations of $\mathcal{CSL}$ with respect to our preferential semantics. We also obtain the finite model property holds for triangular preferential model, whereas it does not hold for triangular distance models. Whether the finite model property holds for metric preferential models is still an open issue, but we strongly conjecture that it is the case. Finally we have shown that the comparative similarity operator corresponds to a conditional operator and that $\mathcal{CSL}$ over the considered classes of models do in fact corresponds to extensions of the well-known conditional logic **VWU** ; the systems corresponding to triangular and metric are new. We have provided sound and complete axiomatizations for all of them.

In future work, we plan to use the preferential semantics to give decision procedures in the form of labeled tableau calculi for $\mathcal{CSL}$ over triangular and metric models, in the line of of [10,9] (getting terminating calculi is namely the crucial issue). We also intend to investigate possible connections between $\mathcal{CSL}$ and belief change theory, as belief change operator are often defined in terms of distances. Finally, for ontology reasoning about comparative similarity, we intend to study the extension of $\mathcal{CSL}$ with nominals and other constructs of various families of description logics.

References

1. Sheremet, M., Tishkovsky, D., Wolter, F., Zakharyaschev, M.: A logic for concepts and similarity. J. Log. Comput. 17(3), 415–452 (2007)
2. Sheremet, M., Tishkovsky, D., Wolter, F., Zakharyaschev, M.: Comparative Similarity, Tree Automata, and Diophantine Equations. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS (LNAI), vol. 3835, pp. 651–665. Springer, Heidelberg (2005)
3. Sheremet, M., Wolter, F., Zakharyaschev, M.: A modal logic framework for reasoning about comparative distances and topology. APAL 161(4), 534–559 (2010)
4. Lewis, D.: Counterfactuals. Basil Blackwell Ltd. (1973)
5. Nute, D.: Topics in Conditional Logic. Reidel Publishing Company (1980)
6. Grahne, G.: Updates and counterfactuals. J. Log. Comput. 8(1), 87 (1998)
7. Alenda, R., Olivetti, N., Schwind, C.: Comparative Concept Similarity over Minspaces: Axiomatisation and Tableaux Calculus. In: Giese, M., Waaler, A. (eds.) TABLEAUX 2009. LNCS, vol. 5607, pp. 17–31. Springer, Heidelberg (2009)
8. Alenda, R., Olivetti, N., Schwind, C., Tishkovsky, D.: Preferential semantics for the logic of comparative concepts similarity. In: Proc. TACL-5 (2010)
9. Alenda, R., Olivetti, N., Schwind, C., Tishkovsky, D.: Tableau Calculi for $\mathcal{CSL}$ over minspaces. In: Dawar, A., Veith, H. (eds.) CSL 2010. LNCS, vol. 6247, pp. 52–66. Springer, Heidelberg (2010)
10. Alenda, R., Olivetti, N.: Tableau Calculus for the Logic of Comparative Similarity over Arbitrary Distance Spaces. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR-17. LNCS, vol. 6397, pp. 52–66. Springer, Heidelberg (2010)
11. Alenda, R., Olivetti, N.: Preferential semantics for the logic of comparative similarity over triangular and metric models. Technical report, Aix-Marseille Université, CNRS, LSIS UMR 7296, 13397, Marseille, France (2012), `http://www.lsis.org/squelettes/publication/upload/3355/alendaolivetti_jelia2012_techreport.pdf`
12. Friedman, N., Halpern, J.Y.: On the complexity of conditional logics. In: Doyle, J., Sandewall, E., Torasso, P. (eds.) KR, pp. 202–213. Morgan Kaufmann (1994)
13. Giordano, L., Gliozzi, V., Olivetti, N.: Weak AGM postulates and strong ramsey test: A logical formalization. Artif. Intell. 168(1-2), 1–37 (2005)
14. Giordano, L., Gliozzi, V., Olivetti, N., Schwind, C.: Tableau calculus for preference-based conditional logics: PCL and its extensions. TOCL 10(3) (2009)
15. Veltman, F.: Logics for Conditionals. Ph.D. dissertation, U. of Amsterdam (1985)
16. Schlechta, K.: Coherent systems, vol 2. Elsevier Science (2004)
17. Williamson, T.: First-order logics for comparative similarity. Notre Dame Journal of Formal Logic 29(4), 457–481 (1988)

Nested Sequent Calculi for Conditional Logics

Régis Alenda[1], Nicola Olivetti[1], and Gian Luca Pozzato[2]

[1] Aix-Marseille Université, CNRS, LSIS UMR 7296, 13397, Marseille, France
{regis.alenda,nicola.olivetti}@univ-cezanne.fr
[2] Dipartimento di Informatica, Università degli Studi di Torino, Italy
pozzato@di.unito.it

Abstract. Nested sequent calculi are a useful generalization of ordinary sequent calculi, where sequents are allowed to occur within sequents. Nested sequent calculi have been profitably employed in the area of (multi)-modal logic to obtain analytic and modular proof systems for these logics. In this work, we extend the realm of nested sequents by providing nested sequent calculi for the basic conditional logic CK and some of its significant extensions. The calculi are internal (a sequent can be directly translated into a formula), cut-free and analytic. Moreover, they can be used to design (sometimes optimal) decision procedures for the respective logics, and to obtain complexity upper bounds. Our calculi are an argument in favour of nested sequent calculi for modal logics and alike, showing their versatility and power.

1 Introduction

The recent history of the conditional logics starts with the work by Lewis [16,17], who proposed them in order to formalize a kind of hypothetical reasoning (if A were the case then B), that cannot be captured by classical logic with material implication. One original motivation was to formalize *counterfactual sentences*, i.e. conditionals of the form "if A were the case then B would be the case", where A is false. Conditional logics have found an interest in several fields of artificial intelligence and knowledge representation. They have been used to reason about prototypical properties [10] and to model belief change [14,12]. Moreover, conditional logics can provide an axiomatic foundation of nonmonotonic reasoning [5,15], here a conditional $A \Rightarrow B$ is read as "in normal circumstances if A then B". Recently, a kind of (multi)-conditional logics [3,4] have been used to formalize epistemic change in a multi-agent setting.

Semantically, all conditional logics enjoy a possible world semantics, with the intuition that a conditional $A \Rightarrow B$ is true in a world x, if B is true in the set of worlds where A is true and that are most similar/closest/"as normal as" x. Since there are different ways of formalizing "the set of worlds similar/closest/..." to a given world, there are expectedly rather different semantics for conditional logics, from the most general selection function semantics to the stronger sphere semantics.

From the point of view of proof-theory and automated deduction, conditional logics do not have however a state of the art comparable with, say, the one of modal logics, where there are well-established alternative calculi, whose proof-theoretical and computational properties are well-understood. This is partially due to the mentioned lack of a unifying semantics; as a matter of fact the most general semantics, the *selection* *function* one, is of little help for proof-theory, and the preferential/sphere semantics only captures a subset of (actually rather strong) systems. Similarly to modal logics and other extensions/alternative to classical logics two types of calculi have been studied: *external* calculi which make use of labels and relations on them to import the semantics into the syntax, and *internal* calculi which stay within the language, so that a "configuration" (sequent, tableaux node...) can be directly interpreted as a formula of the language. Just to mention some work, to the first stream belongs [2] proposing a calculus for (unnested) cumulative logic **C** (see below). More recently, [18] presents modular labeled calculi (of optimal complexity) for CK and some of its extensions, basing on the selection function semantics, and [13] presents modular labeled calculi for preferential logic PCL and its extensions. The latter calculi take advantage of a sort of hybrid modal translation. To the second stream belong the calculi by Gent [9] and by de Swart [21] for Lewis' logic VC and neighbours. These calculi manipulate set of formulas and provide a decision procedure, although they comprise an infinite set of rules. Very recently, some internal calculi for CK and some extensions (with any combination of MP, ID, CEM) have been proposed by Pattinson and Schröder [19]. The calculi are obtained by a general method for closing a set of rules (corresponding to Hilbert axioms) with respect to the cut rule. These calculi have optimal complexity; notice that some of the rules do not have a fixed number of premises. These calculi have been extended to preferential conditional logics [20], i.e. including cumulativity (CM) and or-axiom (CA), although the resulting systems are fairly complicated.

In this paper we begin to investigate *nested sequents* calculi for conditional logics. Nested sequents are a natural generalization of ordinary sequents where sequents are allowed to occur within sequents. However a nested sequent always corresponds to a formula of the language, so that we can think of the rules as operating "inside a formula", combining subformulas rather than just combining outer occurrences of formulas as in ordinary sequents[1]. Limiting to modal logics, nested calculi have been provided, among others, for modal logics by Brünnler [7,6] and by Fitting [8].

In this paper we treat the basic normal conditional logic CK (its role is the same as K in modal logic) and its extensions with ID and CEM. We also consider the *flat* fragment (i.e., without nested conditionals) of CK+CSO+ID, coinciding with the logic of cumulativity **C** introduced in [15]. The calculi are rather natural, all rules have a fixed number of premises. The completeness is established by cut-elimination, whose peculiarity is that it must take into account the substitution of equivalent antecedents of conditionals (a condition corresponding to normality). The calculi can be used to obtain a decision procedure for the respective logics by imposing some restrictions preventing redundant applications of rules. In all cases, we get a PSPACE upper bound, a bound that for CK and CK+ID is optimal (but not for CK+CEM that is known to be coNP). For flat CK+CSO+ID = cumulative logic **C** we also get a PSPACE bound, we are not aware of better upper bound for this logic (although we may suspect that it is not optimal). We can see the present work as a further argument in favor of nested sequents as a useful tool to provide natural, yet computationally adequate, calculi for modal extensions of classical logics.

Technical details and proofs can be found in the accompanying report [1].

[1] In this sense, they are a special kind of "deep inference" calculi by Guglielmi and colleagues.

2 Conditional Logics

A propositional conditional language $\mathcal{L}$ contains: - a set of propositional variables ATM; - the symbol of *false* $\bot$; - a set of connectives $\top$, $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\Rightarrow$. We define formulas of $\mathcal{L}$ as follows: - $\bot$ and the propositional variables of ATM are *atomic formulas*; - if A and B are formulas, then $\neg A$ and $A \otimes B$ are *complex formulas*, where $\otimes \in \{\wedge, \vee, \rightarrow, \Rightarrow\}$. We adopt the *selection function semantics*, that we briefly recall here. We consider a non-empty set of possible worlds $\mathcal{W}$. Intuitively, the selection function f selects, for a world w and a formula A, the set of worlds of $\mathcal{W}$ which are *closer* to w given the information A. A conditional formula $A \Rightarrow B$ holds in a world w if the formula B holds in *all the worlds selected by f for w and A*.

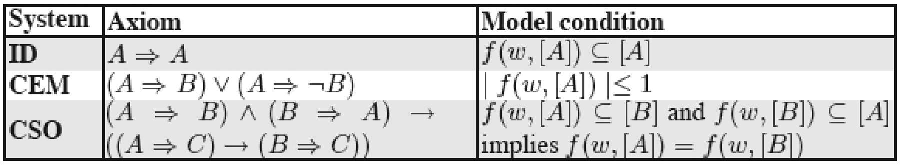
Definition 1 (Selection function semantics). *A model is a triple $\mathcal{M} = \langle \mathcal{W}, f, [\ ] \rangle$ where: - $\mathcal{W}$ is a non empty set of* worlds*; - f is the* selection function $f : \mathcal{W} \times 2^{\mathcal{W}} \longrightarrow 2^{\mathcal{W}}$*; - $[\ ]$ is the* evaluation function*, which assigns to an atom $P \in ATM$ the set of worlds where P is true, and is extended to boolean formulas as usual, whereas for conditional formulas $[A \Rightarrow B] = \{w \in \mathcal{W} \mid f(w, [A]) \subseteq [B]\}$.*

We have defined f taking $[A]$ rather than A (i.e. $f(w, [A])$ rather than $f(w, A)$) as an argument; this is equivalent to define f on formulas, i.e. $f(w, A)$ but imposing that if $[A] = [A']$ in the model, then $f(w, A) = f(w, A')$. This condition is called *normality*.

The semantics above characterizes the *basic conditional system*, called CK [17]. An axiomatization of CK is given by ($\vdash$ denotes provability in the axiom system):

- any axiomatization of the classical propositional calculus;
- If $\vdash A$ and $\vdash A \rightarrow B$, then $\vdash B$ (Modus Ponens)
- If $\vdash A \leftrightarrow B$ then $\vdash (A \Rightarrow C) \leftrightarrow (B \Rightarrow C)$ (RCEA)
- If $\vdash (A_1 \wedge \cdots \wedge A_n) \rightarrow B$ then $\vdash (C \Rightarrow A_1 \wedge \cdots \wedge C \Rightarrow A_n) \rightarrow (C \Rightarrow B)$ (RCK)

Other conditional systems are obtained by assuming further properties on the selection function; we consider the following standard extensions of the basic system CK:



System	Axiom	Model condition
ID	$A \Rightarrow A$	$f(w, [A]) \subseteq [A]$
CEM	$(A \Rightarrow B) \vee (A \Rightarrow \neg B)$	$\mid f(w, [A]) \mid \leq 1$
CSO	$(A \Rightarrow B) \wedge (B \Rightarrow A) \rightarrow ((A \Rightarrow C) \rightarrow (B \Rightarrow C))$	$f(w, [A]) \subseteq [B]$ and $f(w, [B]) \subseteq [A]$ implies $f(w, [A]) = f(w, [B])$

The above axiomatization is complete with respect to the semantics [17].

3 Nested Sequent Calculi $\mathcal{N}S$ for Conditional Logics

In this section we present nested sequent calculi $\mathcal{N}S$, where S is an abbreviation for CK+X, and X=$\{$CEM, ID, CEM+ID$\}$. As usual, completeness is an easy consequence of the admissibility of cut. We are also able to turn $\mathcal{N}S$ into a terminating calculus, which gives us a decision procedure for the respective conditional logics.

Definition 2. *A nested sequent Γ is defined inductively as follows: - A finite multiset of formulas is a nested sequent. - If A is a formula and Γ is a nested sequent, then $[A : \Gamma]$ is a nested sequent. - A finite multiset of nested sequents is a nested sequent.*

A nested sequent can be displayed as

$$A_1, \dots, A_m, [B_1 : \Gamma_1], \dots, [B_n : \Gamma_n],$$

where $n, m \geq 0$, $A_1, \dots, A_m, B_1, \dots, B_n$ are formulas and $\Gamma_1, \dots, \Gamma_n$ are nested sequents. The depth $d(\Gamma)$ of a nested sequent Γ is defined as follows: - if $\Gamma = A_1 \dots, A_n$, then $d(\Gamma) = 0$; - if $\Gamma = [A : \Delta]$, then $d(\Gamma) = 1 + d(\Delta)$ - if $\Gamma = \Gamma_1, \dots, \Gamma_n$, then $d(\Gamma) = max(\Gamma_i)$. A nested sequent can be directly interpreted as a formula, just replace "," by $\vee$ and ":" by $\Rightarrow$. More explicitly, the interpretation of a nested sequent $A_1, \dots, A_m, [B_1 : \Gamma_1], \dots, [B_n : \Gamma_n]$ is inductively defined by the formula $\mathcal{F}(\Gamma) = A_1 \vee \dots \vee A_m \vee (B_1 \Rightarrow \mathcal{F}(\Gamma_1)) \vee \dots \vee (B_n \Rightarrow \mathcal{F}(\Gamma_n))$. For example, the nested sequent $A, B, [A : C, [B : E, F]], [A : D]$ denotes the formula $A \vee B \vee (A \Rightarrow C \vee (B \Rightarrow (E \vee F))) \vee (A \Rightarrow D)$.

The specificity of nested sequent calculi is to allow inferences that apply within formulas. In order to introduce the rules of the calculus, we need the notion of context. Intuitively a context denotes a "hole", a *unique* empty position, within a sequent that can be filled by a formula/sequent. We use the symbol ( ) to denote the empty context. A context is defined inductively as follows:

Definition 3. *If Δ is a nested sequent, $\Gamma(\ ) = \Delta, (\ )$ is a context with depth $d(\Gamma(\ )) = 0$; if Δ is a nested sequent and $\Sigma(\ )$ is a context, $\Gamma(\ ) = \Delta, [A : \Sigma(\ )]$ is a context with depth $d(\Gamma(\ )) = 1 + d(\Sigma(\ ))$.*

Finally we define the result of filling "the hole" of a context by a sequent:

Definition 4. *Let $\Gamma(\ )$ be a context and Δ be a sequent, then the sequent obtained by filling the context by Δ, denoted by $\Gamma(\Delta)$ is defined as follows: - if $\Gamma(\ ) = \Lambda, (\ )$, then $\Gamma(\Delta) = \Lambda, \Delta$; - if $\Gamma(\ ) = \Lambda, [A : \Sigma(\ )]$, then $\Gamma(\Delta) = \Lambda, [A : \Sigma(\Delta)]$.*

The calculi $\mathcal{N}S$ are shown in Figure 1. As usual, we say that a nested sequent Γ is *derivable* in $\mathcal{N}S$ if it admits a *derivation*. A derivation is a tree whose nodes are nested sequents. A branch is a sequence of nodes $\Gamma_1, \Gamma_2, \dots, \Gamma_n, \dots$ Each node Γ_i is obtained from its immediate successor Γ_{i-1} by applying *backward* a rule of $\mathcal{N}S$, having Γ_{i-1} as the conclusion and Γ_i as one of its premises. A branch is closed if one of its nodes is an instance of axioms (AX) and $(AX_\top)$, otherwise it is open. We say that a tree is

$$\underset{P \in ATM}{\Gamma(P, \neg P) \quad (AX)} \qquad \Gamma(\top) \quad (AX_\top) \qquad \frac{\Gamma(A) \qquad \Gamma(B)}{\Gamma(A \wedge B)}(\wedge^+) \qquad \frac{\Gamma(\neg A, \neg B)}{\Gamma(\neg(A \wedge B))}(\wedge^-)$$

$$\frac{\Gamma(A, B)}{\Gamma(A \vee B)}(\vee^+) \qquad \frac{\Gamma(\neg A) \qquad \Gamma(\neg B)}{\Gamma(\neg(A \vee B))}(\vee^-) \qquad \frac{\Gamma(\neg A, B)}{\Gamma(A \rightarrow B)}(\rightarrow^+) \qquad \frac{\Gamma(A) \qquad \Gamma(\neg B)}{\Gamma(\neg(A \rightarrow B))}(\rightarrow^-)$$

$$\frac{\Gamma(A)}{\Gamma(\neg\neg A)}(\neg) \qquad \frac{\Gamma([A : B])}{\Gamma(A \Rightarrow B)}(\Rightarrow^+) \qquad \frac{\Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B]) \qquad A, \neg A' \qquad A', \neg A}{\Gamma(\neg(A \Rightarrow B), [A' : \Delta])}(\Rightarrow^-)$$

$$\frac{\Gamma([A : \Delta, \neg A])}{\Gamma([A : \Delta])}(ID) \qquad \frac{\Gamma([A : \Delta, \Sigma], [B : \Sigma]) \qquad A, \neg B \qquad B, \neg A}{\Gamma([A : \Delta], [B : \Sigma])}(CEM)$$

Fig. 1. The nested sequent calculi $\mathcal{N}S$

$$\dfrac{\dfrac{\dfrac{}{[P : P, \neg P]}\,(AX)}{[P : P]}\,(ID)}{P \Rightarrow P}\,(\Rightarrow^{+})$$

Fig. 2. A derivation of the axiom ID

closed if all its branches are closed. A nested sequent Γ has a derivation in $\mathcal{NS}$ if there is a closed tree having Γ as a root. As an example, Figure 2 shows a derivation of (an instance of) the axiom ID.

The following lemma shows that axioms can be generalized to any formula F:

Lemma 1. *Given any formula F, the sequent $\Gamma(F, \neg F)$ is derivable in $\mathcal{NS}$.*

The easy proof is by induction on the complexity of F.

In [19] the authors propose optimal sequent calculi for CK and its extensions by any combination of ID, MP and CEM. It is not difficult to see that the rules CK_g, $CKID_g$, $CKCEM_g$, $CKCEMID_g$ of their calculi are derivable in our calculi.

3.1 Basic Structural Properties of $\mathcal{NS}$

First of all, we show that weakening and contraction are height-preserving admissible in the calculi $\mathcal{NS}$. Furthermore, we show that all the rules of the calculi, with the exceptions of $(\Rightarrow^{-})$ and (CEM), are height-preserving invertible. As usual, we define the height of a derivation as the height of the tree corresponding to the derivation itself.

Lemma 2 (Admissibility of weakening). *Weakening is height-preserving admissible in $\mathcal{NS}$: if $\Gamma(\Delta)$ (resp. $\Gamma([A : \Delta])$) is derivable in $\mathcal{NS}$ with a derivation of height h, then also $\Gamma(\Delta, F)$ (resp. $\Gamma([A : \Delta, F])$) is derivable in $\mathcal{NS}$ with a proof of height $h' \leq h$, where F is either a formula or a nested sequent $[B : \Sigma]$.*

The easy proof is by induction on the height of the derivation of $\Gamma(\Delta)$.

Lemma 3 (Invertibility). *All the rules of $\mathcal{NS}$, with the exceptions of $(\Rightarrow^{-})$ and (CEM), are height-preserving invertible: if Γ has a derivation of height h and it is an instance of a conclusion of a rule $(\mathbf{R})$, then also Γ_i, $i = 1, 2$, are derivable in $\mathcal{NS}$ with derivations of heights $h_i \leq h$, where Γ_i are instances of the premises of $(\mathbf{R})$.*

Proof. Let us first consider the rule (ID). In this case, we can immediately conclude because the premise $\Gamma([A : \Delta, \neg A])$ is obtained by weakening, which is height-preserving admissible (Lemma 2), from $\Gamma([A : \Delta])$. For the other rules, we proceed by induction on the height of the derivation of Γ. We only show the most interesting case of $(\Rightarrow^{+})$. For the base case, consider $(*)$ $\Gamma(A \Rightarrow B, \Delta)$ where either (i) $P \in \Delta$ and $\neg P \in \Delta$, i.e. $(*)$ is an instance of (AX), or (ii) $\top \in \Delta$, i.e. $(*)$ is an instance of $(AX_{\top})$; we immediately conclude that also $\Gamma([A : B], \Delta)$ is an instance of either (AX) in case

(i) or $(AX_\top)$ in case (ii). For the inductive step, we consider each rule ending (looking forward) the derivation of $\Gamma(A \Rightarrow B)$. If the derivation is ended by an application of $(\Rightarrow^+)$ to $\Gamma([A : B])$, we are done. Otherwise, we apply the inductive hypothesis to the premise(s) and then we conclude by applying the same rule. □

It can be observed that a "weak" version of invertibility also holds for the rules $(\Rightarrow^-)$ and (CEM). Roughly speaking, if $\Gamma(\neg(A \Rightarrow B), [A' : \Delta])$, which is an instance of the conclusion of $(\Rightarrow^-)$, is derivable, then also the sequent $\Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B])$, namely the left-most premise in the rule $(\Rightarrow^-)$, is derivable too. Similarly for (CEM).

Since the rules are invertible, it follows that *contraction* is admissible, that is to say:

Lemma 4 (Admissibility of contraction). *Contraction is height-preserving admissible in $\mathcal{N}S$: if $\Gamma(F, F)$ has a derivation of height h, then also $\Gamma(F)$ has a derivation of height $h' \leq h$, where F is either a formula or a nested sequent $[A : \Sigma]$.*

3.2 Soundness of the Calculi $\mathcal{N}S$

To improve readability, we slightly abuse the notation identifying a sequent Γ with its interpreting formula $\mathcal{F}(\Gamma)$, thus we shall write $A \Rightarrow \Delta$, $\Gamma \wedge \Delta$, etc. instead of $A \Rightarrow \mathcal{F}(\Gamma), \mathcal{F}(\Gamma) \wedge \mathcal{F}(\Delta)$. First of all we prove that nested inference is sound (similarly to Brünnler [7], Lemma 2.8).

Lemma 5. *Let $\Gamma(\ )$ be any context. If the formula $A_1 \wedge \ldots \wedge A_n \rightarrow B$, with $n \geq 0$, is (CK+X)-valid, then also $\Gamma(A_1) \wedge \ldots \wedge \Gamma(A_n) \rightarrow \Gamma(B)$ is (CK+X) valid.*

Proof. By induction on the depth of a context $\Gamma(\ )$. Let $d(\Gamma(\ )) = 0$, then $\Gamma = \Lambda, (\ )$. Since $A_1 \wedge \ldots \wedge A_n \rightarrow B$ is valid, by propositional reasoning, we have that also $(\Lambda \vee A_1) \wedge \ldots (\Lambda \vee A_n) \rightarrow (\Lambda \vee B)$ is valid, that is $\Gamma(A_1) \wedge \ldots \wedge \Gamma(A_n) \rightarrow \Gamma(B)$ is valid. Let $d(\Gamma(\ )) > 0$, then $\Gamma(\ ) = \Delta, [C : \Sigma(\ )]$. By inductive hypothesis, we have that $\Sigma(A_1) \wedge \ldots \wedge \Sigma(A_n) \rightarrow \Sigma(B)$ is valid. By (RCK), we obtain that also $(C \Rightarrow \Sigma(A_1)) \wedge \ldots \wedge (C \Rightarrow \Sigma(A_n)) \rightarrow (C \Rightarrow \Sigma(B))$ is valid. Then, we get that $(\Lambda \vee (C \Rightarrow \Sigma(A_1))) \wedge \ldots \wedge (\Lambda \vee (C \Rightarrow \Sigma(A_n))) \rightarrow (\Lambda \vee (C \Rightarrow \Sigma(B)))$ is also valid, that is $\Gamma(A_1) \wedge \ldots \wedge \Gamma(A_n) \rightarrow \Gamma(B)$ is valid. □

Theorem 1. *If Γ is derivable in $\mathcal{N}S$, then Γ is valid.*

Proof. By induction on the height of the derivation of Γ. If Γ is an axiom, that is $\Gamma = \Gamma(P, \neg P)$, then trivially $P \vee \neg P$ is valid; by Lemma 5 (case $n = 0$), we get $\Gamma(P, \neg P)$ is valid. Similarly for $\Gamma(\top)$. Otherwise Γ is obtained by a rule (**R**):
- (**R**) is a propositional rule, say $\Gamma_1, \Gamma_2 / \Delta$, we first prove that $\Gamma_1 \wedge \Gamma_2 \rightarrow \Delta$ is valid. All rules are easy, since for the empty context they are nothing else than trivial propositional tautologies. We can then use Lemma 5 to propagate them to any context. For instance, let the rule (**R**) be $(\vee^-)$. Then $(\neg A \wedge \neg B) \rightarrow \neg(A \vee B)$ and, by the previous lemma, we get that $\Gamma(\neg A) \wedge \Gamma(\neg B) \rightarrow \Gamma(\neg(A \vee B))$. Thus if Γ is derived by (**R**) from Γ_1, Γ_2, we use the inductive hypothesis that Γ_1 and Γ_2 are valid and the above fact to conclude.
- (**R**) is $(\Rightarrow^+)$: trivial by inductive hypothesis.
- (**R**) is $(\Rightarrow^-)$ then $\Gamma = \Gamma(\neg(A \Rightarrow B), [A' : \Delta])$ is derived from (i) $\Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B])$, (ii) $\neg A, A'$, (iii) $\neg A', A$. By inductive hypothesis we have that $A \leftrightarrow A'$ is

valid. We show that also $(*)$ $[\neg(A \Rightarrow B) \vee (A' \Rightarrow (\Delta \vee \neg B))] \rightarrow [\neg(A \Rightarrow B) \vee (A' \Rightarrow \Delta)]$ is valid, then we apply Lemma 5 and the inductive hypothesis to conclude. To prove (*), by (RCK) we have that the following is valid: $[(A' \Rightarrow B) \wedge (A' \Rightarrow (\Delta \vee \neg B))] \rightarrow (A' \Rightarrow \Delta)$. Since $A \leftrightarrow A'$ is valid, by (RCEA) we get that $(A \Rightarrow B) \rightarrow (A' \Rightarrow B)$ is valid, so that also $(A \Rightarrow B) \rightarrow ((A' \Rightarrow (\Delta \vee \neg B)) \rightarrow (A' \Rightarrow \Delta))$ is valid, then we conclude by propositional reasoning.

- (**R**) is (ID), then $\Gamma = \Gamma([A : \Delta])$ is derived from $\Gamma([A : \Delta, \neg A])$. We show that $(A \Rightarrow (\Delta \vee \neg A)) \rightarrow (A \Rightarrow \Delta)$ is valid in CK+ID, then we conclude by Lemma 5 and by the inductive hypothesis. The mentioned formula is derivable: by (RCK) we obtain $(A \Rightarrow A) \rightarrow ((A \Rightarrow (\Delta \vee \neg A)) \rightarrow (A \Rightarrow \Delta))$ so that we conclude by (ID).
- (**R**) is (CEM), thus $\Gamma = \Gamma([A : \Delta], [A', \Sigma])$ and it is derived from (i) $\Gamma([A : \Delta, \Sigma], [A' : \Sigma])$, (ii) $\neg A, A'$, (iii) $\neg A', A$. By inductive hypothesis $A \leftrightarrow A'$ is valid. We first show that $(**)$ $(A \Rightarrow (\Delta \vee \Sigma)) \rightarrow ((A \Rightarrow \Delta) \vee (A' \Rightarrow \Sigma))$. Then we conclude as before by Lemma 5 and inductive hypothesis. To prove (**), we notice that the following is derivable by (RCK): $(A \Rightarrow (\Delta \vee \Sigma)) \rightarrow [(A \Rightarrow \neg\Delta) \rightarrow (A \Rightarrow \Sigma)]$. By (CEM), the following is valid: $(A \Rightarrow \Delta) \vee (A \Rightarrow \neg\Delta)$. Thus we get that $(A \Rightarrow (\Delta \vee \Sigma)) \rightarrow [(A \Rightarrow \Delta) \vee (A \Rightarrow \Sigma)]$ is valid. Since $A \leftrightarrow A'$ is valid, (by RCEA) we have that also $(A \Rightarrow \Sigma) \rightarrow (A' \Rightarrow \Sigma)$ is valid, obtaining (**). □

3.3 Completeness of the Calculi $\mathcal{N}S$

Completeness is an easy consequence of the admissibility of the following rule *cut*:

$$\frac{\Gamma(F) \qquad \Gamma(\neg F)}{\Gamma(\emptyset)}\ (cut)$$

where F is a formula. The standard proof of admissibility of cut proceeds by a double induction over the complexity of F and the sum of the heights of the derivations of the two premises of (cut), in the sense that we replace one cut by one or several cuts on formulas of smaller complexity, or on sequents derived by shorter derivations. However, in $\mathcal{N}S$ the standard proof does not work in the following case, in which the cut formula F is a conditional formula $A \Rightarrow B$:

$$\dfrac{\dfrac{(1)\ \Gamma([A : B], [A' : \Delta])}{(3)\ \Gamma(A \Rightarrow B, [A' : \Delta])}\,(\Rightarrow^+) \qquad \dfrac{(2)\ \Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B]) \qquad A, \neg A' \qquad A', \neg A}{\Gamma(\neg(A \Rightarrow B), [A' : \Delta])}\,(\Rightarrow^+)}{\Gamma([A' : \Delta])}\,(cut)$$

Indeed, even if we apply the inductive hypothesis on the heights of the derivations of the premises to cut (2) and (3), obtaining (modulo weakening, which is admissible by Lemma 2) a derivation of $(2')$ $\Gamma([A' : \Delta, \neg B], [A' : \Delta])$, we cannot apply the inductive hypothesis on the complexity of the cut formula to $(2')$ and $(1')$ $\Gamma([A : \Delta, B], [A' : \Delta])$ (obtained from (1) again by weakening). Such an application would be needed in order to obtain a proof of $\Gamma([A' : \Delta], [A' : \Delta])$ and then to conclude $\Gamma([A' : \Delta])$ since contraction is admissible (Lemma 4).

In order to prove the admissibility of cut for $\mathcal{N}S$, we proceed as follows. First, we show that if $A, \neg A'$ and $A', \neg A$ are derivable, then if $\Gamma([A : \Delta])$ is derivable, then $\Gamma([A' : \Delta])$, obtained by replacing $[A : \Delta]$ with $[A' : \Delta]$, is also derivable. We prove that cut is admissible by "splitting" the notion of cut in two propositions:

Theorem 2. *In $\mathcal{NS}$, the following propositions hold:* (A) *If $\Gamma(F)$ and $\Gamma(\neg F)$ are derivable, so is $\Gamma(\emptyset)$, i.e. (cut) is admissible in $\mathcal{NS}$;* (B) *if (I) $\Gamma([A : \Delta])$, (II) $A, \neg A'$ and (III) $A', \neg A$ are derivable, then $\Gamma([A' : \Delta])$ is derivable.*

Proof. The proof of both is by mutual induction. To make the structure of the induction clear call: $Cut(c, h)$ the property (A) for any Γ and any formula F of complexity c and such that the sum of the heights of derivation of the premises is h. Similarly call $Sub(c)$ the assertion that (B) holds for any Γ and any formula A of complexity c. Then we show the following facts:

(i) $\forall h\ Cut(0, h)$
(ii) $\forall c\ Cut(c, 0)$
(iii) $\forall c' < c\ Sub(c') \rightarrow (\forall c' < c\ \forall h'\ Cut(c', h') \wedge \forall h' < h\ Cut(c, h') \rightarrow Cut(c, h))$
(iv) $\forall h\ Cut(c, h) \rightarrow Sub(c)$

This will prove that $\forall c\ \forall h Cut(c, h)$ and $\forall c\ Sub(c)$, that is (A) and (B) hold. The proof of (iv) (that is that $Sub(c)$ holds) in itself is by induction on the height h of the derivation of the premise (I) of (B). To save space, we only present the most interesting cases.

Inductive step for (A): we distinguish the following two cases:

• (case 1) the last step of *one* of the two premises is obtained by a rule (**R**) in which F is *not* the principal formula. This case is standard, we can permute (**R**) over the cut, i.e. we cut the premise(s) of (**R**) and then we apply (**R**) to the result of cut.
• (case 2) F is the principal formula in the last step of *both* derivations of the premises of the cut inference. There are seven subcases: F is introduced a) by $(\wedge^-)$ - $(\wedge^+)$, b) by $(\vee^-)$ - $(\vee^+)$, c) by $(\rightarrow^-)$ - $(\rightarrow^+)$, d) by $(\Rightarrow^-)$ - $(\Rightarrow^+)$, e) by $(\Rightarrow^-)$ - (ID), f) by $(\Rightarrow^-)$ - (CEM), g) by (CEM) - (ID). We only show the most interesting case d), where the derivation is as follows:

$$\dfrac{\dfrac{(1)\ \Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B]) \quad A, \neg A' \quad A', \neg A}{\Gamma(\neg(A \Rightarrow B), [A' : \Delta])}\,(\Rightarrow^-) \qquad \dfrac{(2)\ \Gamma([A : B], [A' : \Delta])}{(3)\ \Gamma(A \Rightarrow B, [A' : \Delta])}\,(\Rightarrow^+)}{\Gamma([A' : \Delta])}\,(cut)$$

First of all, since we have proofs for $A, \neg A'$ and for $A', \neg A$ and $cp(A) < cp(A \Rightarrow B)$, we can apply the inductive hypothesis for (B) to (2), obtaining a proof of $(2')$ $\Gamma([A' : B], [A' : \Delta])$. By Lemma 2, from (3) we obtain a proof of at most the same height of $(3')$ $\Gamma(A \Rightarrow B, [A' : \Delta, \neg B])$. We can then conclude as follows: we first apply the inductive hypothesis on the height for (A) to cut (1) and $(3')$, obtaining a derivation of (4) $\Gamma([A' : \Delta, \neg B])$. By Lemma 2, we have also a derivation of $(4')$ $\Gamma([A' : \Delta, \neg B], [A' : \Delta])$. Again by Lemma 2, from $(2')$ we obtain a derivation of $(2'')$ $\Gamma([A' : \Delta, B], [A' : \Delta])$. We then apply the inductive hypothesis on the complexity of the cut formula to cut $(2'')$ and $(4')$, obtaining a proof of $\Gamma([A' : \Delta], [A' : \Delta])$, from which we conclude since contraction is admissible (Lemma 4).

Inductive step for (B) (that is statement (iv) of the induction): we have to consider all possible rules ending (looking forward) the derivation of $\Gamma([A : \Delta])$. We only show the most interesting case, when $(\Rightarrow^-)$ is applied by using $[A : \Delta]$ as principal formula. The derivation ends as follows:

$$\dfrac{(1)\ \Gamma(\neg(C \Rightarrow D), [A : \Delta, \neg D]) \qquad (2)\ C, \neg A \qquad (3)\ A, \neg C}{\Gamma(\neg(C \Rightarrow D), [A : \Delta])}\ (\Rightarrow^-)$$

We can apply the inductive hypothesis to (1) to obtain a derivation of $(1')$ $\Gamma(\neg(C \Rightarrow D), [A' : \Delta, \neg D])$. Since weakening is admissible (Lemma 2), from (II) we obtain a derivation of (II') $C, A, \neg A'$, from (III) we obtain a derivation of (III') $A', \neg A, \neg C$. Again by weakening, from (2) and (3) we obtain derivations of $(2')$ $C, \neg A, \neg A'$ and $(3')$ $A', A, \neg C$, respectively. We apply the inductive hypothesis of (A) that is that cut holds for the formula A (of a given complexity c) and conclude as follows:

$$\dfrac{(1')\ \Gamma(\neg(C \Rightarrow D), [A' : \Delta, \neg D]) \quad \dfrac{(II')\ C, A, \neg A' \quad (2')\ C, \neg A, \neg A'}{C, \neg A'}\ (cut) \quad \dfrac{(III')\ A', \neg A, \neg C \quad (3')\ A', A, \neg C}{A', \neg C}\ (cut)}{\Gamma(\neg(C \Rightarrow D), [A' : \Delta])}\ (\Rightarrow^-)$$

□

Theorem 3 (Completeness of $\mathcal{NS}$). *If Γ is valid, then it is derivable in $\mathcal{NS}$.*

Proof. We prove that the axioms are derivable and that the set of derivable formulas is closed under (Modus Ponens), (RCEA), and (RCK). A derivation of an instance of ID has been shown in Figure 2. Here is a derivation of an instance of CEM:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{[A : B, \neg B]}\ (AX) \quad \dfrac{}{A, \neg A}\ (AX) \quad \dfrac{}{\neg A, A}\ (AX)}{[A : B], [A : \neg B]}\ (CEM)}{[A : B], A \Rightarrow \neg B}\ (\Rightarrow^+)}{A \Rightarrow B, A \Rightarrow \neg B}\ (\Rightarrow^+)}{(A \Rightarrow B) \vee (A \Rightarrow \neg B)}\ (\vee^+)$$

For (Modus Ponens), the proof is standard and is omitted to save space. For (RCEA), we have to show that if $A \leftrightarrow B$ is derivable, then also $(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)$ is derivable. As usual, $A \leftrightarrow B$ is an abbreviation for $(A \rightarrow B) \wedge (B \rightarrow A)$. Since $A \leftrightarrow B$ is derivable, and since $(\wedge^+)$ and $(\rightarrow^+)$ are invertible (Lemma 3), we have a derivation for $A \rightarrow B$, then for (1) $\neg A, B$, and for $B \rightarrow A$, then for (2) $A, \neg B$. We derive $(A \Rightarrow C) \rightarrow (B \Rightarrow C)$ (the other half is symmetric) as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(A \Rightarrow C), [B : C, \neg C]}\ (AX) \qquad (1)\ \neg A, B \qquad (2)\ A, \neg B}{\neg(A \Rightarrow C), [B : C]}\ (\Rightarrow^-)}{\neg(A \Rightarrow C), B \Rightarrow C}\ (\Rightarrow^+)}{(A \Rightarrow C) \rightarrow (B \Rightarrow C)}\ (\rightarrow^+)$$

For (RCK), suppose that we have a derivation in $\mathcal{NS}$ of $(A_1 \wedge \ldots \wedge A_n) \rightarrow B$. Since $(\rightarrow^+)$ is invertible (Lemma 3), we have also a derivation of $B, \neg(A_1 \wedge \ldots \wedge A_n)$. Since $(\wedge^-)$ is also invertible, then we have a derivation of $B, \neg A_1, \ldots, \neg A_n$ and, by weakening (Lemma 2), of (1) $\neg(C \Rightarrow A_1), \ldots, \neg(C \Rightarrow A_n), [C : B, \neg A_1, \neg A_2, \ldots, \neg A_n]$, from which we conclude as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}(1)\ \neg(C \Rightarrow A_1), \dots, \neg(C \Rightarrow A_n), [C : B, \neg A_1, \neg A_2, \dots, \neg A_n] \\ \cdots\end{matrix}}{\neg(C \Rightarrow A_1), \dots, \neg(C \Rightarrow A_n), [C : B, \neg A_1, \neg A_2]}(\Rightarrow^-) \quad \dfrac{}{C, \neg C}(AX) \quad \dfrac{}{\neg C, C}(AX)}{\neg(C \Rightarrow A_1), \dots, \neg(C \Rightarrow A_n), [C : B, \neg A_1]}(\Rightarrow^-) \quad \dfrac{}{C, \neg C}(AX) \quad \dfrac{}{\neg C, C}(AX)}{\neg(C \Rightarrow A_1), \dots, \neg(C \Rightarrow A_n), [C : B]}(\Rightarrow^-)}{\neg(C \Rightarrow A_1 \wedge \dots \wedge C \Rightarrow A_n), [C : B]}(\wedge^-)}{\neg(C \Rightarrow A_1 \wedge \dots \wedge C \Rightarrow A_n), C \Rightarrow B}(\Rightarrow^+)}{(C \Rightarrow A_1 \wedge \dots \wedge C \Rightarrow A_n) \rightarrow (C \Rightarrow B)}(\rightarrow^+)$$

□

3.4 Termination and Complexity of $\mathcal{N}S$

The rules $(\Rightarrow^-)$, (CEM), and (ID) may be applied infinitely often. In order to obtain a terminating calculus, we have to put some restrictions on the application of these rules. Let us first consider the systems without CEM. We put the following restrictions:

- apply $(\Rightarrow^-)$ to $\Gamma(\neg(A \Rightarrow B), [A' : \Delta])$ only if $\neg B \notin \Delta$;
- apply (ID) to $\Gamma([A : \Delta])$ only if $\neg A \notin \Delta$.

These restrictions impose that $(\Rightarrow^-)$ is applied only once to each formula $\neg(A \Rightarrow B)$ with a context $[A' : \Delta]$ in each branch, and that (ID) is applied only once to each context $[A : \Delta]$ in each branch.

Theorem 4. *The calculi $\mathcal{N}S$ with the termination restrictions is sound and complete for their respective logics.*

Proof. We show that it is useless to apply the rules $(\Rightarrow^-)$ and (ID) without the restrictions. We only present the case of $(\Rightarrow^-)$. Suppose it is applied twice on $\Gamma([A : \Delta], [B : \Sigma])$ in a branch. Since $(\Rightarrow^-)$ is "weakly" invertible, we can assume, without loss of generality, that the two applications of $(\Rightarrow^-)$ are consecutive, starting from $\Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B, \neg B])$. By Lemma 4 (contraction), we have a derivation of $\Gamma([A : \Delta, \Sigma], [B : \Sigma])$, and we can conclude with a (single) application of $(\Rightarrow^-)$. □

The above restrictions ensure a terminating proof search for the nested sequents for CK and CK+ID, in particular:

Theorem 5. *The calculi $\mathcal{N}CK$ and $\mathcal{N}CK + ID$ with the termination restrictions give a* PSPACE *decision procedure for their respective logics.*

For the systems allowing CEM, we need a more sophisticated machinery[2], that allows us also to conclude that:

Theorem 6. *The calculi $\mathcal{N}CK{+}CEM$ and $\mathcal{N}CK{+}CEM{+}ID$ with the termination restrictions give a* PSPACE *decision procedure for their respective logics.*

It is worth noticing that our calculi match the PSPACE lower-bound of the logics CK and CK+ID, and are thus optimal with respect to these logics. On the contrary the calculi for CK+CEM(+ID) are not optimal, since validity in these logics is known to be decidable in coNP. In future work we shall try to devise an optimal decision procedure by adopting a suitable strategy.

[2] The termination of the calculi with (CEM) can be found in [1].

4 A Calculus for the Flat Fragment of CK+CSO+ID

In this section we show another application of nested sequents to give an analytic calculus for the flat fragment, i.e. without nested conditionals $\Rightarrow$, of CK+CSO+ID. This logic is well-known and it corresponds to logic **C**, the logic of *cumulativity*, the weakest system in the family of KLM logics [15]. Formulas are restricted to boolean combinations of propositional formulas and conditionals $A \Rightarrow B$ where A and B are propositionals. A sequent has then the form:

$$A_1, \dots, A_m, [B_1 : \Delta_1], \dots, [B_m : \Delta_m]$$

where B_i and Δ_i are propositional. The logic has also an alternative semantics in terms of *weak preferential models*. The rules of $\mathcal{N}CK + CSO + ID$ are those ones of $\mathcal{N}CK + ID$ (restricted to the flat fragment) where the rule $(\Rightarrow^-)$ is replaced by the rule (CSO):

$$\dfrac{\Gamma, \neg(C \Rightarrow D), [A : \Delta, \neg D] \qquad \Gamma, \neg(C \Rightarrow D), [A : C] \qquad \Gamma, \neg(C \Rightarrow D), [C : A]}{\Gamma, \neg(C \Rightarrow D), [A : \Delta]}\ (CSO)$$

A derivation of an instance of CSO is easy and left to the reader. More interestingly, in Figure 3 we give an example of derivation of the cumulative axiom $((A \Rightarrow B) \wedge (A \Rightarrow C)) \rightarrow (A \wedge B) \Rightarrow C$.

$$\dfrac{\dfrac{\neg(A \Rightarrow B), \neg(A \Rightarrow C), [A \wedge B : C, \neg C] \qquad \dfrac{\dfrac{\Sigma, [A \wedge B : A, \neg A, \neg B)]}{\Sigma, [A \wedge B : A, \neg(A \wedge B))]}\ (\wedge^-)}{\neg(A \Rightarrow B), \neg(A \Rightarrow C), [A \wedge B : A]}\ (ID) \qquad \overset{\Pi_1}{\neg(A \Rightarrow B), \neg(A \Rightarrow C), [A : A \wedge B]}}{\neg(A \Rightarrow B), \neg(A \Rightarrow C), [A \wedge B : C]}\ (CSO)}{\neg(A \Rightarrow B), \neg(A \Rightarrow C), (A \wedge B) \Rightarrow C}\ (\Rightarrow^+)$$

where Π_1 is the following derivation:

$$\dfrac{\dfrac{\Sigma, [A : A, \neg A]}{\neg(A \Rightarrow B), \neg(A \Rightarrow C), [A : A]}\ (ID) \qquad \dfrac{\Sigma, [A : B, \neg B] \qquad \dfrac{\Sigma, [A : A, \neg A]}{\Sigma, [A : A]}\ (ID) \qquad \dfrac{\Sigma, [A : A, \neg A]}{\Sigma, [A : A]}\ (ID)}{\neg(A \Rightarrow B), \neg(A \Rightarrow C), [A : B]}\ (CSO)}{\neg(A \Rightarrow B), \neg(A \Rightarrow C), [A : A \wedge B]}\ (\wedge^+)$$

Fig. 3. A derivation of the cumulative axiom $((A \Rightarrow B) \wedge (A \Rightarrow C)) \rightarrow (A \wedge B) \Rightarrow C$. We omit the first propositional steps, and we let $\Sigma = \neg(A \Rightarrow B), \neg(A \Rightarrow C)$.

Definition 5. *A sequent Γ is* reduced *if it has the form $\Gamma = \Lambda, \Pi, [B_1 : \Delta_1], \dots, [B_m : \Delta_m]$, where Λ is a multiset of literals and Π is a multiset of* negative *conditionals.*

The following proposition is a kind of *disjunctive property* for reduced sequents.

Proposition 1. *Let $\Gamma = \Lambda, \Pi, [B_1 : \Delta_1], \dots, [B_m : \Delta_m]$ be reduced, if Γ is derivable then for some i, the sequent $\Lambda, \Pi, [B_i : \Delta_i]$ is (height-preserving) derivable.*

Proposition 2. *Let $\Gamma = \Sigma, [B_1 : \Delta_1], \ldots, [B_m : \Delta_m]$ be any sequent, if Γ is derivable then it can be (height-preserving) derived from some* reduced *sequents $\Gamma_i = \Sigma_j, [B_1 : \Delta_1], \ldots, [B_m : \Delta_m]$. Moreover all rules applied to derive Γ from Γ_i are either propositional rules or the rule $(\Rightarrow^+)$.*

Proposition 2 can be proved by permuting (downwards) all the applications of propositional and $(\Rightarrow^+)$ rules.

Proposition 3. *Let $\Gamma = \Sigma, [A : \Delta], [A : \Delta]$ be derivable, then $\Gamma = \Sigma, [A : \Delta]$ is (height-preserving) derivable.*

Proof. By Proposition 2, Γ is height-preserving derivable from a set of reduced sequents $\Sigma_i, [A : \Delta], [A : \Delta]$. By Proposition 1, each $\Sigma_i, [A : \Delta]$ is derivable; we then obtain $\Sigma, [A : \Delta]$ by applying the same sequence of rules. □

Theorem 7. *Contraction is admissible in $\mathcal{N}CK+CSO+ID$: if $\Gamma(F, F)$ is derivable, then $\Gamma(F)$ is (height-preserving) derivable, where F is either a formula or a nested sequent $[A : \Sigma]$.*

Proof. (Sketch) If F is a formula the proof is essentially the same as the one for Lemma 4 in $\mathcal{N}S$. If $F = [A : \Sigma]$ we apply Proposition 3. □

Observe that the standard inductive proof of contraction does not work in the case $F = [A : \Delta]$, that is why we have obtained it by Proposition 3 which in turn is based on Proposition 1, a kind of disjunctive property. The same argument *does not extend* immediatly to the full language with nested conditionals.

As usual, we obtain completeness by cut-elimnation. As in case of $\mathcal{N}S$, the proof is by mutual induction together with a substitution property and is left to the reader due to space limitations.

Theorem 8. *In $\mathcal{N}CK + CSO + ID$, the following propositions hold:* (A) *If $\Gamma(F)$ and $\Gamma(\neg F)$ are derivable, then so is Γ;* (B) *if (I) $\Gamma([A : \Delta])$, (II) $\Gamma([A : A'])$, and (III) $\Gamma([A' : A])$ are derivable, then so is $\Gamma([A' : \Delta])$.*

Theorem 9. *The calculus $\mathcal{N}CK + CSO + ID$ is sound and complete for the flat fragment of* CK+CSO+ID.

Proof. For soundness just check the validity of the (CSO) rule. For completeness, one can derive all instances of CSO axioms. Moreover the rules RCK and RCEA are derivable too (by using the rules (ID) and (CSO)). For closure under modus ponens, as usual we use the previous Theorem 8. Details are left to the reader. □

Termination of this calculus can be proved similarly to Theorem 5, details will be given in a full version of the paper.

Theorem 10. *The calculus $\mathcal{N}CK + CSO + ID$ with the termination restrictions give a* PSPACE *decision procedure for the flat fragment of* CK+CSO+ID.

We do not know whether this bound is optimal. The study of the optimal complexity for CK+CSO+ID is still open. A NEXP tableau calculus for cumulative logic **C** has been proposed in [11]. In [20] the authors provide calculi for full (i.e. with nested conditionals) CK+ID+CM and CK+ID+CM+CA: these logics are related to CSO, but they do not concide with it, even for the flat fragment as CSO = CM+RT (restricted transitivity). Their calculi are internal, but rather complex as the make use of ingenious but highly combinatorial rules. They obtain a PSPACE bound in all cases.

5 Conclusions and Future Works

In this work we have provided nested sequent calculi for the basic normal conditional logic and a few extensions of it. The calculi are analytic and their completeness is established via cut-elimination. The calculi can be used to obtain a decision procedure, in some cases of optimal complexity. We have also provided a nested sequent calculus for the flat fragment of CK+CSO+ID, corresponding to the cumulative logic **C** of the KLM framework. Even if for some of the logics considered in this paper there exist other proof systems, we think that nested calculi are particularly natural internal calculi. Obviously, it is our goal to extent them to a wider spectrum of conditional logics, in particular preferential conditional logics, which still lack "natural" and internal calculi. We also intend to study improvements of the calculi towards efficiency, based on a better control of duplication. Finally, we wish to take advantage of the calculi to study logical properties of the corresponding systems (disjunction property, interpolation) in a constructive way.

References

1. Alenda, R., Olivetti, N., Pozzato, G.L.: Nested sequents for Conditional Logics: preliminary results, Technical Report (2012), `http://www.di.unito.it/~pozzato/termcso.pdf`
2. Artosi, A., Governatori, G., Rotolo, A.: Labelled tableaux for non-monotonic reasoning: Cumulative consequence relations. Journal of Logic and Computation 12(6), 1027–1060 (2002)
3. Baltag, A., Smets, S.: The logic of conditional doxastic actions. Texts in Logic and Games, Special Issue on New Perspectives on Games and Interaction 4, 9–31 (2008)
4. Board, O.: Dynamic interactive epistemology. Games and Economic Behavior 49(1), 49–80 (2004)
5. Boutilier, C.: Conditional logics of normality: a modal approach. Artificial Intelligence 68(1), 87–154 (1994)
6. Brünnler, K.: Deep sequent systems for modal logic. Archive for Mathematical Logic 48(6), 551–577 (2009), `http://www.iam.unibe.ch/~kai/Papers/2009dssml.pdf`
7. Brünnler, K.: Nested sequents (2010), habilitation Thesis, `http://arxiv.org/pdf/1004.1845`
8. Fitting, M.: Prefixed tableaus and nested sequents. Annals of Pure Applied Logic 163(3), 291–313 (2012)
9. Gent, I.P.: A sequent or tableaux-style system for lewis's counterfactual logic vc. Notre Dame Journal of Formal Logic 33(3), 369–382 (1992)
10. Ginsberg, M.L.: Counterfactuals. Artificial Intelligence 30(1), 35–79 (1986)

11. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Analytic Tableaux Calculi for KLM Logics of Nonmonotonic Reasoning. ACM Trans. Comput. Logic 10(3) (2009)
12. Giordano, L., Gliozzi, V., Olivetti, N.: Weak AGM postulates and strong ramsey test: A logical formalization. Artificial Intelligence 168(1-2), 1–37 (2005)
13. Giordano, L., Gliozzi, V., Olivetti, N., Schwind, C.: Tableau calculus for preference-based conditional logics: Pcl and its extensions. ACM Trans. Comput. Logic 10(3) (2009)
14. Grahne, G.: Updates and counterfactuals. Journal of Logic and Computation 8(1), 87–117 (1998)
15. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
16. Lewis, D.: Counterfactuals. Basil Blackwell Ltd. (1973)
17. Nute, D.: Topics in conditional logic. Reidel, Dordrecht (1980)
18. Olivetti, N., Pozzato, G.L., Schwind, C.B.: A Sequent Calculus and a Theorem Prover for Standard Conditional Logics. ACM Trans. Comput. Logic 8(4) (2007)
19. Pattinson, D., Schröder, L.: Generic modal cut elimination applied to conditional logics. Logical Methods in Computer Science 7(1) (2011)
20. Schröder, L., Pattinson, D., Hausmann, D.: Optimal tableaux for conditional logics with cautious monotonicity. In: ECAI. pp. 707–712 (2010)
21. de Swart, H.C.M.: A gentzen- or beth-type system, a practical decision procedure and a constructive completeness proof for the counterfactual logics vc and vcs. Journal of Symbolic Logic 48(1), 1–20 (1983)

Conflict-Tolerant Semantics for Argumentation Frameworks

Ofer Arieli

School of Computer Science, The Academic College of Tel-Aviv, Israel
`oarieli@mta.ac.il`

Abstract. We introduce new kinds of semantics for abstract argumentation frameworks, in which, while all the accepted arguments are justified (in the sense that each one of them must be defended), they may still attack each other. The rationality behind such semantics is that in reality there are situations in which contradictory arguments coexist in the same theory, yet the collective set of accepted arguments is not trivialized, in the sense that other arguments may still be rejected.

To provide conflict-tolerant semantics for argumentation frameworks we extend the two standard approaches for defining coherent (conflict-free) semantics for argumentation frameworks: the extension-based approach and the labeling-based approach. We show that the one-to-one relationship between extensions and labelings of conflict-free semantics is carried on to a similar correspondence between the extended approaches for providing conflict-tolerant semantics. Thus, in our setting as well, these are essentially two points of views for the same thing.

1 Introduction and Motivation

An abstract argumentation framework consists of a set of (abstract) arguments and a binary relation that intuitively represents attacks between arguments. A semantics for such a structure is an indication which arguments can be collectively accepted. A starting point of all the existing semantics for abstract argumentation frameworks is that their set(s) of acceptable arguments must be conflict-free, that is: an accepted argument should not attack another accepted argument. This means, in particular, a dismissal of any self-referring argument and a rejection of any contradictory fragment of the chosen arguments. However, in everyday life it is not always the case that theories are completely coherent although each of their arguments provides a solid assertion, and so contradictory sets of arguments should sometimes be accepted and tolerated. Moreover, a removal of contradictory indications in such theories may imply a loss of information and may lead to erroneous conclusions.

In this paper, we consider a more liberal approach for argumentation semantics, adhering conflicting indications (and so inconsistencies). For this, we extend the two most standard approaches for defining semantics to abstract argumentation frameworks as follows:

- *Extension-Based Semantics.* Existing semantics that are defined by this method share two primary principles: admissibility and conflict-freeness (see, e.g., [2, 3]). The former principle, guaranteeing that an extension Ext 'defends' all of its elements (i.e., Ext 'counterattacks' each argument that attacks some $e \in Ext$), is preserved also in our framework, since otherwise acceptance of arguments would be an arbitrary choice. However, the other principle is lifted, since – as indicated above – we would like to permit, in some cases, conflicting arguments.
- *Labeling-Based Semantics.* We extend the traditional three-state labelings of arguments (accepted, rejected, undecided – see [5, 7]) by a fourth state, so now apart of accepting or rejecting an argument, we have *two* additional states, representing two opposite reasons for avoiding a definite opinion about the argument as hand: One ('none'), indicating that there is too little evidence for reaching a precise conclusion about the argument's validity, and the other ('both') indicating 'too much' (contradictory) evidence, i.e., the existence of both supportive and opposing arguments concerning the argument under consideration.

Both of these generalized approaches are a conservative extension of the standard approaches of giving semantics to abstract argumentation systems, in the sense that they do not exclude standard extensions or labelings, but rather offer additional points of views to the state of affairs as depicted by the argumentation framework. This allows us to introduce a brand new family of semantics that accommodate conflicts in the sense that internal attacks among accepted arguments are allowed, while the set of accepted arguments is not trivialized (i.e., it is not the case that every argument is necessarily accepted).

We introduce an extended set of criteria for selecting the most plausible four-valued labelings for an argumentation framework. These criteria are then justified by showing that the one-to-one relationship between extensions and labelings obtained for conflict-free semantics (see [7]) is carried on to a similar correspondence between the extended approaches for providing conflict-tolerant (paraconsistent) semantics. This also shows that in the case of conflict-tolerant semantics as well, extensions and labelings are each other's dual.

2 Preliminaries

Let us first recall some basic definitions and useful notions regarding abstract argumentation theory.

Definition 1. A (finite) *argumentation framework* [8] is a pair $\mathcal{AF} = \langle Args, att \rangle$, where $Args$ is a finite set, the elements of which are called *arguments*, and att is a binary relation on $Args \times Args$ whose instances are called *attacks*. When $(A, B) \in att$ we say that A *attacks* B (or that B is *attacked by* A).

Given an argumentation framework $\mathcal{AF} = \langle Args, att \rangle$, in the sequel we shall use the following notations for an argument $A \in Args$ and a set of arguments $\mathcal{S} \subseteq Args$:

- The set of arguments that are *attacked by* A is $A^+ = \{B \in \mathit{Args} \mid \mathit{att}(A, B)\}$.
- The set of arguments that *attack* A is $A^- = \{B \in \mathit{Args} \mid \mathit{att}(B, A)\}$.

Similarly, $\mathcal{S}^+ = \bigcup_{A \in \mathcal{S}} A^+$ and $\mathcal{S}^- = \bigcup_{A \in \mathcal{S}} A^-$ denote, respectively, the set of arguments that are attacked by some argument in $\mathcal{S}$ and the set of arguments that attack some argument in $\mathcal{S}$. Accordingly, we denote:

- The set of arguments that are *defended by* $\mathcal{S}$: $\mathrm{Def}(\mathcal{S}) = \{A \in \mathit{Args} \mid A^- \subseteq \mathcal{S}^+\}$.

Thus, an argument A is defended by $\mathcal{S}$ if each attacker of A is attacked by (an argument in) $\mathcal{S}$. The two primary principles of acceptable sets of arguments are now defined as follows:

Definition 2. Let $\mathcal{AF} = \langle \mathit{Args}, \mathit{att} \rangle$ be an argumentation framework.

- A set $\mathcal{S} \subseteq \mathit{Args}$ is *conflict-free* (with respect to $\mathcal{AF}$) iff $\mathcal{S} \cap \mathcal{S}^+ = \emptyset$.
- A conflict-free set $\mathcal{S} \subseteq \mathit{Args}$ is *admissible* for $\mathcal{AF}$, iff $\mathcal{S} \subseteq \mathrm{Def}(\mathcal{S})$.

Conflict-freeness assures that no argument in the set is attacked by another argument in the set, and admissibility guarantees, in addition, that the set is self defendant. A stronger notion is the following:

- A conflict-free set $\mathcal{S} \subseteq \mathit{Args}$ is *complete* for $\mathcal{AF}$, iff $\mathcal{S} = \mathrm{Def}(\mathcal{S})$.

The principles defined above are a cornerstone of a variety of extension-based semantics for an argumentation framework $\mathcal{AF}$, i.e., formalizations of sets of arguments that can collectively be accepted in $\mathcal{AF}$ (see, e.g., [8, 12]). In what follows, we shall usually denote an extension by Ext. This includes, among others, *grounded extensions* (the minimal set, with respect to set inclusion, that is complete for $\mathcal{AF}$), *preferred extensions* (the maximal subset of Args that is complete for $\mathcal{AF}$), *stable extensions* (any complete subset Ext of Args for which $\mathit{Ext}^+ = \mathit{Args} \setminus \mathit{Ext}$), and so forth.[1]

An alternative way to describe argumentation semantics is based on the concept of an *argument labeling* [5, 7]. The main definitions and the relevant results concerning this approach are surveyed below.

Definition 3. Let $\mathcal{AF} = \langle \mathit{Args}, \mathit{att} \rangle$ be an argumentation framework. An *argument labeling* is a complete function $\mathit{lab} : \mathit{Args} \to \{\mathsf{in}, \mathsf{out}, \mathsf{undec}\}$. We shall sometimes write $\mathsf{In}(\mathit{lab})$ for $\{A \in \mathit{Args} \mid \mathit{lab}(A) = \mathsf{in}\}$, $\mathsf{Out}(\mathit{lab})$ for $\{A \in \mathit{Args} \mid \mathit{lab}(A) = \mathsf{out}\}$ and $\mathsf{Undec}(\mathit{lab})$ for $\{A \in \mathit{Args} \mid \mathit{lab}(A) = \mathsf{undec}\}$.

In essence, an argument labeling expresses a position on which arguments one accepts (labeled in), which arguments one rejects (labeled out), and which arguments one abstains from having an explicit opinion about (labeled undec). Since a labeling lab of $\mathcal{AF} = \langle \mathit{Args}, \mathit{att} \rangle$ can be seen as a partition of Args, we shall sometimes write it as a triple $\langle \mathsf{In}(\mathit{lab}), \mathsf{Out}(\mathit{lab}), \mathsf{Undec}(\mathit{lab}) \rangle$.

[1] Common definitions of conflict-free extension-based semantics for argumentation frameworks, different methods for computing them, and computational complexity analysis appear, e.g., in [1, 6, 8, 9, 10, 11].

Definition 4. Consider the following conditions on a labeling *lab* and an argument A in a framework $\mathcal{AF} = \langle Args, att \rangle$:

Pos1	If $lab(A) = \mathsf{in}$, there is no $B \in A^-$ such that $lab(B) = \mathsf{in}$.
Pos2	If $lab(A) = \mathsf{in}$, for every $B \in A^-$ it holds that $lab(B) = \mathsf{out}$.
Neg	If $lab(A) = \mathsf{out}$, there exists some $B \in A^-$ such that $lab(B) = \mathsf{in}$.
Neither	If $lab(A) = \mathsf{undec}$, not for every $B \in A^-$ it holds that $lab(B) = \mathsf{out}$ and there does not exist a $B \in A^-$ such that $lab(B) = \mathsf{in}$.

Given a labeling *lab* of an argumentation framework $\langle Args, att \rangle$, we say that

- *lab* is *conflict-free* if for every $A \in Args$ it satisfies conditions **Pos1** and **Neg**,
- *lab* is *admissible* if for every $A \in Args$ it satisfies conditions **Pos2** and **Neg**,
- *lab* is *complete* if it is admissible and for every $A \in Args$ it satisfies **Neither**.

Again, the labelings considered above serve as a basis for a variety of labeling-based semantics that have been proposed for an argumentation framework $\mathcal{AF}$, each one of which is a counterpart of a corresponding extension-based semantics. This includes, for instance, the *grounded labeling* (a complete labeling for $\mathcal{AF}$ with a minimal set of in-assignments), the *preferred labeling* (a complete labeling for $\mathcal{AF}$ with a maximal set of in-assignments), *stable labelings* (any complete labeling of $\mathcal{AF}$ without undec-assignments), and so forth.

The next correspondence between extensions and labelings is shown in [7]:

Proposition 1. *Let $\mathcal{AF} = \langle Args, att \rangle$ be an argumentation framework, $\mathcal{CFE}$ the set of all conflict-free extensions of $\mathcal{AF}$, and $\mathcal{CFL}$ the set of all conflict-free labelings of $\mathcal{AF}$. Consider the function $\mathcal{LE}_{\mathcal{AF}} : \mathcal{CFL} \to \mathcal{CFE}$, defined by $\mathcal{LE}_{\mathcal{AF}}(lab) = \mathsf{In}(lab)$ and the function $\mathcal{EL}_{\mathcal{AF}} : \mathcal{CFE} \to \mathcal{CFL}$, defined by $\mathcal{EL}_{\mathcal{AF}}(Ext) = \langle Ext, Ext^+, Args \setminus (Ext \cup Ext^+) \rangle$. It holds that:*

1. *If Ext is an admissible (respectively, complete) extension, then $\mathcal{EL}_{\mathcal{AF}}(Ext)$ is an admissible (respectively, complete) labeling.*
2. *If lab is an admissible (respectively, complete) labeling, then $\mathcal{LE}_{\mathcal{AF}}(lab)$ is an admissible (respectively, complete) extension.*
3. *When the domain and range of $\mathcal{EL}_{\mathcal{AF}}$ and $\mathcal{LE}_{\mathcal{AF}}$ are restricted to complete extensions and complete labelings of $\mathcal{AF}$, then these functions become bijections and each other's inverses, making complete extensions and complete labelings one-to-one related.*

3 Conflicts Tolerance

In this section we extend the two approaches considered previously in order to define conflict-tolerant semantics for abstract argumentation frameworks. Recall that our purpose here is twofold:

1. Introducing self-referring argumentation and avoiding information loss that may be caused by the conflict-freeness requirement (thus, for instance, it may be better to accept extensions with a small fragment of conflicting arguments than, say, sticking to the empty extension).

2. Refining the undec-indication in standard labeling systems, which reflects (at least) two totally different situations: One case is that the reasoner abstains from having an opinion about an argument because there are no indications whether this argument should be accepted or rejected. Another case that may cause a neutral opinion is that there are simultaneous considerations for and against accepting a certain argument. These two cases should be distinguishable, since their outcomes may be different.

3.1 Four-Valued Paraconsistent Labelings

Item 2 above may serve as a motivation for the following definition:

Definition 5. Let $\mathcal{AF} = \langle Args, att\rangle$ be an argumentation framework. A *four-valued labeling* for $\mathcal{AF}$ is a complete function $lab : Args \to \{\mathsf{in}, \mathsf{out}, \mathsf{none}, \mathsf{both}\}$. We shall sometimes write $\mathsf{None}(lab)$ for $\{A \in Args \mid lab(A) = \mathsf{none}\}$ and $\mathsf{Both}(lab)$ for $\{A \in Args \mid lab(A) = \mathsf{both}\}$.

As before, the labeling function reflects the state of mind of the reasoner regarding each argument in $\mathcal{AF}$. The difference is, of-course, that four-valued labelings are a refinement of 'standard' labelings (in the sense of Definition 3), so that *four* states are allowed. Thus, we continue to denote by $\mathsf{In}(lab)$ the set of arguments that one accepts and by $\mathsf{Out}(lab)$ the set of arguments that one rejects, but now the set $\mathsf{Undec}(lab)$ is splitted to two new sets: $\mathsf{None}(lab)$, consisting of arguments that may neither be accepted nor rejected, and $\mathsf{Both}(lab)$, consisting of arguments who have both supportive and rejective evidences. Since a four-valued labeling lab is a partition of $Args$, we sometimes write it as a quadruple $\langle \mathsf{In}(lab), \mathsf{Out}(lab), \mathsf{None}(lab), \mathsf{Both}(lab)\rangle$.

Definition 6. Let $\mathcal{AF} = \langle Args, att\rangle$ be an argumentation framework.

– Given a set $Ext \subseteq Args$ of arguments, the function that is *induced by* (or, is *associated with*) Ext is the four-valued labeling $p\mathcal{EL}_{\mathcal{AF}}(Ext)$ of $\mathcal{AF}$,[2] defined for every $A \in Args$ as follows:

$$p\mathcal{EL}_{\mathcal{AF}}(Ext)(A) = \begin{cases} \mathsf{in} & \text{if } A \in Ext \text{ and } A \notin Ext^+, \\ \mathsf{both} & \text{if } A \in Ext \text{ and } A \in Ext^+, \\ \mathsf{out} & \text{if } A \notin Ext \text{ and } A \in Ext^+, \\ \mathsf{none} & \text{if } A \notin Ext \text{ and } A \notin Ext^+. \end{cases}$$

A four-valued labeling that is induced by some subset of $Args$ is called a *paraconsistent labeling* (or a *p-labeling*) of $\mathcal{AF}$.

– Given a four-valued labeling lab of $\mathcal{AF}$, the set of arguments that is *induced by* (or, is *associated with*) lab is defined by

$$p\mathcal{LE}_{\mathcal{AF}}(lab) = \mathsf{In}(lab) \cup \mathsf{Both}(lab).$$

[2] Here, $p\mathcal{EL}$ stands for a **p**araconsistent-based conversion of **e**xtensions to **l**abelings.

The intuition behind the transformation from a labeling lab to its extension $p\mathcal{LE}_{\mathcal{AF}}(lab)$ is that any argument for which there is some supportive indication (i.e., it is labeled in or both) should be included in the extension (even if there are also opposing indications). The transformation from an extension Ext to its induced labeling function $p\mathcal{EL}_{\mathcal{AF}}(Ext)$ is motivated by the aspiration to accept the arguments in the extension by marking them as either in or both. Since Ext is not necessarily conflict-free, two labels are required to indicate whether the argument at hand is attacked by another argument in the extension, or not.

Definition 6 indicates a one-to-one correspondence between sets of arguments of an argumentation framework and the labelings that are induced by them. It follows that while there are $4^{|Args|}$ four-valued labelings for an argumentation framework $\mathcal{AF} = \langle Args, att \rangle$, the number of paraconsistent labelings (p-labelings) for $\mathcal{AF}$ is limited by the number of the subsets of $Args$, i.e., $2^{|Args|}$.

Example 1. Consider the argumentation framework $\mathcal{AF}_1$ of Figure 1.

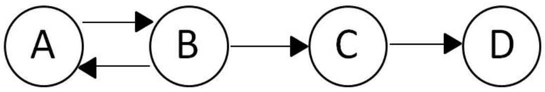


Fig. 1. The argumentation framework $\mathcal{AF}_1$

To compute the paraconsistent labelings of $\mathcal{AF}_1$, note for instance that if for some $Ext \subseteq Args$ it holds that $p\mathcal{EL}_{\mathcal{AF}}(Ext)(A) = \text{in}$, then $A \in Ext$ and $A \notin Ext^+$, which implies, respectively, that $B \in Ext^+$ and $B \notin Ext$, thus B must be labeled out. Similarly, if A is labeled out then B must be labeled in, if A is labeled both, B must be labeled both as well, and if A is labeled none, so B is labeled none. These labelings correspond to the four possible choices of either accepting exactly one of the mutually attacking arguments A and B, accepting both of them, or rejecting both of them. In turn, each such choice is augmented with four respective options for labeling C and D. Table 1 lists the corresponding sixteen p-labelings of $\mathcal{AF}_1$.

A p-labeling may be regarded as a description of the state of affairs for *any* chosen set of arguments in a framework. For instance, the second p-labeling in Table 1 (Example 1) indicates that if $\{A, C, D\}$ is the accepted set of arguments, then B is rejected (labeled out) since it is attacked by an accepted argument, and the status of D is ambiguous (so it is labeled both), since on one hand it is included in the set of accepted arguments, but on the other hand it is attacked by an accepted argument. Note, further, that choosing D as an accepted argument in this case is somewhat arguable, since D is not defended by the set $\{A, C, D\}$.

The discussion above implies that the role of a p-labeling is *indicative* rather than *justificatory*; A labeling that is induced by Ext describes the role of each argument in the framework according to Ext, but it does not *justify* the choice

Table 1. The p-labelings of $\mathcal{AF}_1$

	A	B	C	D	Induced set
1	in	out	in	out	$\{A, C\}$
2	in	out	in	both	$\{A, C, D\}$
3	in	out	none	in	$\{A, D\}$
4	in	out	none	none	$\{A\}$
5	out	in	out	in	$\{B, D\}$
6	out	in	out	none	$\{B\}$
7	out	in	both	out	$\{B, C\}$
8	out	in	both	both	$\{B, C, D\}$
9	none	none	in	out	$\{C\}$
10	none	none	in	both	$\{C, D\}$
11	none	none	none	in	$\{D\}$
12	none	none	none	none	$\{\}$
13	both	both	out	in	$\{A, B, D\}$
14	both	both	out	none	$\{A, B\}$
15	both	both	both	out	$\{A, B, C\}$
16	both	both	both	both	$\{A, B, C, D\}$

of *Ext* as a plausible extension for the framework. For the latter, we should pose further restrictions on the p-labelings. This is what we do next.

Definition 7. Given an argumentation framework $\mathcal{AF} = \langle Args, att \rangle$, a p-labeling *lab* for $\mathcal{AF}$ is called *p-admissible*, if it satisfies the following rules:

pIn If $lab(A) = \mathsf{in}$, for every $B \in A^-$ it holds that $lab(B) = \mathsf{out}$.
pOut If $lab(A) = \mathsf{out}$, there exists $B \in A^-$ such that $lab(B) \in \{\mathsf{in}, \mathsf{both}\}$.
pBoth If $lab(A) = \mathsf{both}$, for every $B \in A^-$ it holds that $lab(B) \in \{\mathsf{out}, \mathsf{both}\}$ and there exists some $B \in A^-$ such that $lab(B) = \mathsf{both}$.
pNone If $lab(A) = \mathsf{none}$, for every $B \in A^-$ it holds that $lab(B) \in \{\mathsf{out}, \mathsf{none}\}$.

The constraints in Definition 7 may be strengthen as follows:

Definition 8. Given an argumentation framework $\mathcal{AF} = \langle Args, att \rangle$, a p-labeling *lab* for $\mathcal{AF}$ is called *p-complete*, if it satisfies the following rules:

pIn$^+$ $lab(A) = \mathsf{in}$ iff for every $B \in A^-$ it holds that $lab(B) = \mathsf{out}$.
pOut$^+$ $lab(A) = \mathsf{out}$ iff there is $B \in A^-$ such that $lab(B) \in \{\mathsf{in}, \mathsf{both}\}$ and there is some $B \in A^-$ such that $lab(B) \in \{\mathsf{in}, \mathsf{none}\}$.
pBoth$^+$ $lab(A) = \mathsf{both}$ iff for every $B \in A^-$ it holds that $lab(B) \in \{\mathsf{out}, \mathsf{both}\}$ and there exists some $B \in A^-$ such that $lab(B) = \mathsf{both}$.
pNone$^+$ $lab(A) = \mathsf{none}$ iff for every $B \in A^-$ it holds that $lab(B) \in \{\mathsf{out}, \mathsf{none}\}$ and there exists some $B \in A^-$ such that $lab(B) = \mathsf{none}$.

Example 2. Consider again the p-labelings for $\mathcal{AF}_1$ (Example 1), listed in Table 1.

– The rule **pIn** is violated by labelings 3, 9, 10, 11, and the rule **pBoth** is violated by labelings 2, 7, 8, 10. Therefore, the p-admissible labelings in this case are 1, 4, 5, 6, 12–16.
– Among the p-admissible labelings in the previous item, labelings 4 and 6 violate **pNone$^+$**, and labelings 13–15 violate **pOut$^+$**. Thus, the p-complete labelings of $\mathcal{AF}_1$ are 1, 5, 12 and 16.[3]

[3] Intuitively, these labelings represent the most plausible states corresponding to the four possible choices of arguments among the mutually attacking A and B.

In Section 3.3 and Section 4 we shall justify the rules in Definitions 7 and 8 by showing the correspondence between p-admissible/p-complete labelings and related extensions.

3.2 Paraconsistent Extensions

Recall that Item 1 at the beginning of Section 3 suggests that the 'conflict-freeness' requirement in Definition 2 may be lifted. However, the other properties in the same definition, implying that an argument in an extension must be defended, are still necessary.

Definition 9. Let $\mathcal{AF} = \langle Args, att \rangle$ be an argumentation framework and let $Ext \subseteq Args$. We say that Ext is a *paraconsistently admissible* (or: *p-admissible*) extension for $\mathcal{AF}$ if $Ext \subseteq \mathrm{Def}(Ext)$. Ext is a *paraconsistently complete* (or: *p-complete*) extension for $\mathcal{AF}$ if $Ext = \mathrm{Def}(Ext)$.

Thus, every admissible (respectively, complete) extension for $\mathcal{AF}$ is also p-admissible (respectively, p-complete) extension for $\mathcal{AF}$, but not the other way around.

It is well-known that every argumentation framework has at least one complete extension. However, there are cases (see, e.g., the argumentation framework $\mathcal{AF}_2$ in Figure 2) that the only complete extension for a framework is the empty set. The next proposition shows that this is not the case regarding p-complete extensions.

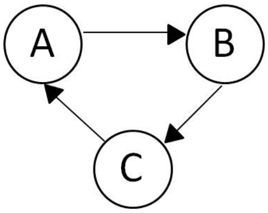


Fig. 2. The argumentation framework $\mathcal{AF}_2$

Proposition 2. *Any argumentation framework has a nonempty p-complete extension.*[4]

Example 3. The argumentation framework $\mathcal{AF}_2$ that is shown in Figure 2 has two p-complete extensions: $\emptyset$ (which is also the only complete extension of $\mathcal{AF}_2$), and $\{A, B, C\}$.

[4] Due to lack of space proofs are omitted.

3.3 Relating Paraconsistent Extensions and Labelings

We are now ready to consider the extension-based semantics induced by paraconsistent labelings. We show, in particular, that as in the case of (conflict-free) complete labelings and (conflict-free) complete extensions, there is a one-to-one correspondence between them, thus they represent two equivalent approaches for giving conflict-tolerant semantics to abstract argumentation frameworks.

Proposition 3. *If Ext is a p-admissible extension of $\mathcal{AF}$ then $p\mathcal{EL}_{\mathcal{AF}}(Ext)$ is a p-admissible labeling of $\mathcal{AF}$.*

Proposition 4. *If lab is a p-admissible labeling of $\mathcal{AF}$ then $p\mathcal{LE}_{\mathcal{AF}}(lab)$ is a p-admissible extension for $\mathcal{AF}$.*

Proposition 5. *Let $\mathcal{AF} = \langle Args, att\rangle$ be an argumentation framework.*

– *For every p-admissible labeling lab for $\mathcal{AF}$, $p\mathcal{EL}_{\mathcal{AF}}(p\mathcal{LE}_{\mathcal{AF}}(lab)) = lab$.*
– *For every p-admissible extension Ext of $\mathcal{AF}$, $p\mathcal{LE}_{\mathcal{AF}}(p\mathcal{EL}_{\mathcal{AF}}(Ext)) = Ext$.*

By Propositions 3, 4, and 5, we conclude the following:

Corollary 1. *The functions $p\mathcal{EL}_{\mathcal{AF}}$ and $p\mathcal{LE}_{\mathcal{AF}}$, restricted to the p-admissible labelings and the p-admissible extensions of $\mathcal{AF}$, are bijective, and are each other's inverse.*

It follows that p-admissible extensions and p-admissible labelings are, in essence, different ways of describing the same thing (see also Figure 3 below).

Note 1. In a way, the correspondence between p-admissible extensions and p-admissible labelings of an argumentation framework is tighter than the correspondence between (conflict-free) admissible extensions and (conflict-free) admissible labelings, as depicted in [7] (see Section 2). Indeed, as indicated in [7], admissible labelings and admissible extensions have a many-to-one relationship: each admissible labeling is associated with exactly one admissible extension, but an admissible extension may be associated with several admissible labelings. For instance, in the argumentation framework $\mathcal{AF}_1$ of Figure 1 (Example 1), $lab_1 = \langle\{B\}, \{A, C\}, \{D\}\rangle$ and $lab_2 = \langle\{B\}, \{A\}, \{C, D\}\rangle$ are different admissible labelings that are associated with the same admissible extension $\{B\}$. Note that, viewed as *four-valued* labelings into $\{\mathsf{in}, \mathsf{out}, \mathsf{none}\}$, only lab_1 is p-admissible, since lab_2 violates **pNone**. Indeed, the p-admissible extension $\{B\}$ is associated with exactly *one* p-admissible labeling (number 6 in Table 1), as guaranteed by the last corollary.

Let us now consider p-complete labelings.

Proposition 6. *If Ext is a p-complete extension of $\mathcal{AF}$ then $p\mathcal{EL}_{\mathcal{AF}}(Ext)$ is a p-complete labeling of $\mathcal{AF}$.*

Proposition 7. *If lab is a p-complete labeling of $\mathcal{AF}$ then $p\mathcal{LE}_{\mathcal{AF}}(lab)$ is a p-complete extension for $\mathcal{AF}$.*

Proposition 8. *Let $\mathcal{AF}$ be an argumentation framework.*

– *For every p-complete labeling lab for $\mathcal{AF}$, $p\mathcal{EL}_{\mathcal{AF}}(p\mathcal{LE}_{\mathcal{AF}}(lab)) = lab$.*
– *For every p-complete extension Ext of $\mathcal{AF}$, $p\mathcal{LE}_{\mathcal{AF}}(p\mathcal{EL}_{\mathcal{AF}}(Ext)) = Ext$.*

By Propositions 6, 7, and 8, we conclude the following:

Corollary 2. *The functions $p\mathcal{EL}_{\mathcal{AF}}$ and $p\mathcal{LE}_{\mathcal{AF}}$, restricted to the p-complete labelings and the p-complete extensions of $\mathcal{AF}$, are bijective, and are each other's inverse.*

It follows that p-complete extensions and p-complete labelings are different ways of describing the same thing (see also Figure 3). This is in correlation with the results for conflict-free semantics, according to which there is a one-to-one relationship between complete extensions and complete labelings (Proposition 1).

Example 4. Consider again the framework $\mathcal{AF}_1$ of Example 1.

1. By Example 2 and Propositions 4, the p-admissible extensions of $\mathcal{AF}_1$ are induced by labelings 1, 4, 5, 6, 12–16 in Table 1, i.e., $\{A, C\}$, $\{A\}$, $\{B, D\}$, $\{B\}$, $\emptyset$, $\{A, B, D\}$, $\{A, B\}$, $\{A, B, C\}$, and $\{A, B, C, D\}$ (respectively).
2. By Example 2 and Propositions 7, the p-complete extensions of $\mathcal{AF}_1$ are those that are induced by labelings 1, 5, 12 and 16 in Table 1, namely $\{A, C\}$, $\{B, D\}$, $\emptyset$, and $\{A, B, C, D\}$ (respectively).

4 From Conflict-Tolerant to Conflict-Free Semantics

In this section we show that the variety of 'standard' semantics for argumentation frameworks, based on conflict-free extensions and conflict-free labeling functions, are still available in our conflict-tolerant setting. First, we consider admissible extensions (Definition 2) and admissible labelings (Definition 4).

Proposition 9. *Let lab be a p-admissible labeling for an argumentation framework $\mathcal{AF}$. If lab is into $\{\mathsf{in}, \mathsf{out}, \mathsf{none}\}$,*[5] *then it is admissible.*

As Note 1 shows, the converse of the last proposition does not hold. Indeed, as indicated by Caminada and Gabbay [7], there is a many-to-one relationship between admissible labelings and admissible extensions. On the other hand, by the next proposition (together with Corollary 2), there is a one-to-one relationship between both-free p-admissible labelings and admissible extensions.

Proposition 10. *Let $\mathcal{AF} = \langle Args, att \rangle$ be an argumentation framework. Then*

1. *If lab is a* both*-free p-admissible labeling for $\mathcal{AF}$, then $p\mathcal{LE}_{\mathcal{AF}}(lab)$ is an admissible extension of $\mathcal{AF}$.*
2. *If Ext is an admissible extension of $\mathcal{AF}$ then $p\mathcal{EL}_{\mathcal{AF}}(Ext)$ is a* both*-free p-admissible labeling for $\mathcal{AF}$.*

[5] In which case *lab* is called 'both-free'.

Let us now consider complete extensions and complete labelings. The next two propositions are the analogue, for complete labelings and complete extensions, of Propositions 9 and 10:

Proposition 11. *Let lab be a p-complete labeling for an argumentation framework $\mathcal{AF}$. If lab is into* $\{\mathsf{in}, \mathsf{out}, \mathsf{none}\}$*, then it is complete.*

Proposition 12. *Let $\mathcal{AF} = \langle Args, att \rangle$ be an argumentation framework. Then*

1. *If lab is a* both*-free p-complete labeling for $\mathcal{AF}$, then $p\mathcal{LE}_{\mathcal{AF}}(lab)$ is a complete extension of $\mathcal{AF}$.*
2. *If Ext is a complete extension of $\mathcal{AF}$ then $p\mathcal{EL}_{\mathcal{AF}}(Ext)$ is a* both*-free p-complete labeling for $\mathcal{AF}$.*

Proposition 13. *Let $\mathcal{AF}$ be an argumentation framework. Then lab is a complete labeling for $\mathcal{AF}$ iff it is a* both*-free p-complete labeling for $\mathcal{AF}$.*

Figure 3 summarizes the relations between the conflict-free semantics and the conflict-tolerant semantics considered so far (the arrows in the figure denote "is-a" relationships, and the double-arrows denote one-to-one relationships).

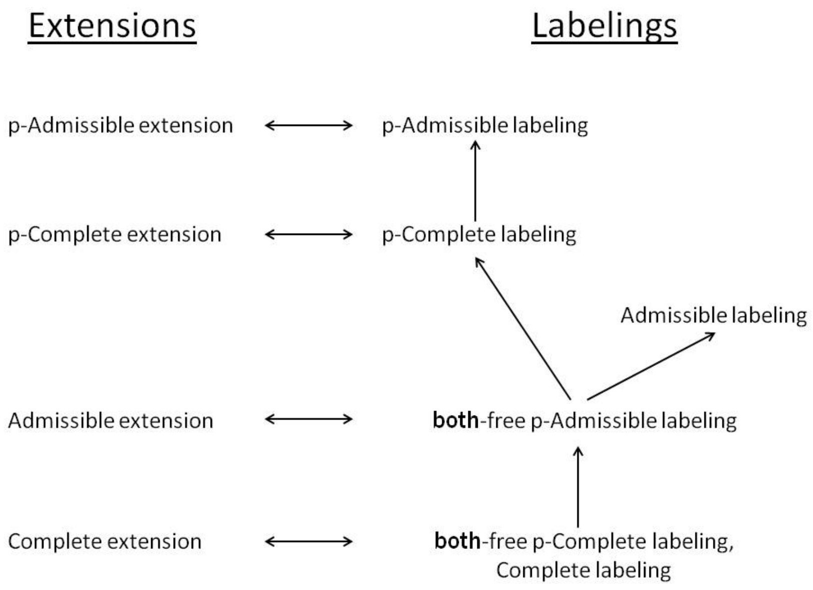


Fig. 3. Conflict-free and conflict-tolerant semantics

By Proposition 13, a variety of conflict-free, extension-based (Dung-style) semantics for abstract argumentation frameworks may be defined in terms of both-free p-complete labelings. For instance,

- Ext is a grounded extension of $\mathcal{AF}$ iff it is induced by a both-free p-complete labeling lab of $\mathcal{AF}$ such that there is no both-free p-complete labeling lab'' of $\mathcal{AF}$ with $\mathsf{In}(lab'') \subset \mathsf{In}(lab)$.
- Ext is a preferred extension of $\mathcal{AF}$ iff it is induced by a both-free p-complete labeling lab of $\mathcal{AF}$ such that there is no both-free p-complete labeling lab'' of $\mathcal{AF}$ with $\mathsf{In}(lab) \subset \mathsf{In}(lab'')$.
- Ext is a semi-stable extension of $\mathcal{AF}$ iff it is induced by a both-free p-complete labeling lab of $\mathcal{AF}$ such that there is no both-free p-complete labeling lab'' of $\mathcal{AF}$ with $\mathsf{None}(lab'') \subset \mathsf{None}(lab)$.
- Ext is a stable extension of $\mathcal{AF}$ iff it is induced by a both-free p-complete labeling lab of $\mathcal{AF}$ such that $\mathsf{None}(lab) = \emptyset$.

By the last item, stable extensions correspond to $\{\mathsf{both}, \mathsf{none}\}$-free p-complete labelings:

Corollary 3. *Let $\mathcal{AF} = \langle Args, att \rangle$ be an argumentation framework.*

1. *If lab is a $\{\mathsf{both}, \mathsf{none}\}$-free p-complete labeling for $\mathcal{AF}$, then $p\mathcal{LE}_{\mathcal{AF}}(lab)$ is a stable extension of $\mathcal{AF}$.*
2. *If Ext is a stable extension of $\mathcal{AF}$, then $p\mathcal{EL}_{\mathcal{AF}}(Ext)$ is a $\{\mathsf{both}, \mathsf{none}\}$-free p-complete labeling for $\mathcal{AF}$.*

Example 5. Consider again the framework $\mathcal{AF}_1$ of Example 1.

1. By Proposition 12, the complete extensions of $\mathcal{AF}_1$ are induced by the both-free p-complete labelings, i.e., $\{A, C\}$, $\{B, D\}$ and $\emptyset$ (which are the both-free labelings among those mentioned in Items 2 of Examples 2 and 4).
2. By Corollary 3, the stable extensions of $\mathcal{AF}_1$ are those induced by the none-free labelings among the labeling in the previous item, i.e., $\{A, C\}$ and $\{B, D\}$.

5 Conclusion

We have introduced a four-valued approach to provide conflict-tolerant semantics for abstract argumentation frameworks. Such an approach may be beneficial for several reasons:

- From a purely theoretical point of view, we have shown that the correlation between the labeling-based approach and the extension-based approach to argumentation theory is preserved also when conflict-freeness is abandon. Interestingly, as indicated in Note 1, in our framework this correlation holds also between admissibility-based labelings and admissibility-based extensions, which is *not* the case in the conflict-free setting of [7].
- From a more pragmatic point of view, new types of semantics are introduced, which accommodate conflicts, yet they are not trivialized by inconsistency. It is shown that this setting is not a substitute of standard (conflict-free) semantics, but rather a generalized framework, offering an option for inter-attacks when such attacks make sense or are unavoidable.

– Conflicts handling in argumentation systems turns out to be more evasive than what it looks like at first sight. In fact, conflicts may implicitly arise even in conflict-free semantics, because such semantics simulate binary attacks and not collective conflicts (this is demonstrated in the last example of [2]). In this respect, the possibility of having conflicts is not completely ruled out even in some conflict-free semantics (such as CF2 and stage semantics; see [2]), and our approach may be viewed as an explication of this possibility.

Our setting may be related to other settings that to the best of our knowledge have not been connected so far to argumentation theory. For instance, the resemblance to Belnap's well-known four-valued framework for computerized reasoning [4] is evident. Moreover, the use of four-valued labelings suggests that the four-valued signed systems, used in [1] for representing conflict-free semantics of argumentation frameworks, may be incorporated for representing the conflict-tolerant semantics in this paper. We leave this for a future work.

Acknowledgement. Martin Caminada is thanked for useful discussions on topics related to this paper.

References

[1] Arieli, O., Caminada, M.W.A.: A general QBF-based framework for formalizing argumentation. In: Proc. COMMA 2012. IOS Press (in press, 2012)
[2] Baroni, P., Caminada, M., Giacomin, M.: An introduction to argumentation semantics. The Knowledge Engineering Review 26(4), 365–410 (2011)
[3] Baroni, P., Giacomin, M.: Semantics for abstract argumentation systems. In: Rahwan and Simary [13], pp. 25–44
[4] Belnap, N.D.: A useful four-valued logic. In: Dunn, J.M., Epstein, G. (eds.) Modern Uses of Multiple-Valued Logics, pp. 7–37. Reidel Publishing (1977)
[5] Caminada, M.: On the Issue of Reinstatement in Argumentation. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) JELIA 2006. LNCS (LNAI), vol. 4160, pp. 111–123. Springer, Heidelberg (2006)
[6] Caminada, M., Carnielli, W.A., Dunne, P.: Semi-stable semantics. Journal of Logic and Computation (in print, 2012)
[7] Caminada, M., Gabbay, D.M.: A logical account of formal argumentation. Studia Logica. 93(2-3), 109–145 (2009)
[8] Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77, 321–357 (1995)
[9] Dung, P.M., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artificial Intelligence 171(10-15), 642–674 (2007)
[10] Dvořák, W.: On the Complexity of Computing the Justification Status of an Argument. In: Modgil, S., Oren, N., Toni, F. (eds.) TAFA 2011. LNCS (LNAI), vol. 7132, pp. 32–49. Springer, Heidelberg (2012)
[11] Egly, U., Gaggl, S.A., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Argument and Computation 1(2), 144–177 (2010)
[12] Rahwan, I., Simari, G.R.: Argumentation in Artificial Intelligence. Springer (2009)

Knowledge Means '*All*', Belief Means '*Most*'

Dimitris Askounis[1], Costas D. Koutras[2], and Yorgos Zikos[3]

[1] Decision Support Systems Lab
School of Electrical and Comp. Engineering
National Technical University of Athens
9, Iroon Polytechniou Street, 15773 Athens, Greece
askous@epu.ntua.gr

[2] Department of Computer Science and Technology
University of Peloponnese
End of Karaiskaki Street, 22 100 Tripolis, Greece
ckoutras@uop.gr

[3] Graduate Programme in Logic, Algorithms and Computation (MPLA)
Department of Mathematics, University of Athens
Panepistimioupolis, 157 84 Ilissia, Greece
zikos@sch.gr

Abstract. We introduce a bimodal epistemic logic intended to capture *knowledge* as truth in *all* epistemically alternative states and *belief* as a generalized '*majority*' quantifier, interpreted as truth in *many* (a '*majority*' of the) epistemically alternative states. This doxastic interpretation is of interest in KR applications and it also has an independent philosophical and technical interest. The logic **KBM** comprises an **S4** epistemic modal operator, a doxastic modal operator of consistent and complete belief and 'bridge' axioms which relate knowledge to belief. To capture the notion of a '*majority*' we use the '*large sets*' introduced independently by K. Schlechta and V. Jauregui, augmented with a requirement of completeness, which furnishes a '*weak ultrafilter*' concept. We provide semantics in the form of possible-worlds frames, properly blending relational semantics with a version of general Scott-Montague (neighborhood) frames and we obtain soundness and completeness results. We examine the validity of certain epistemic principles discussed in the literature, in particular some of the 'bridge' axioms discussed by W. Lenzen and R. Stalnaker, as well as the '*paradox of the perfect believer*', which is not a theorem of **KBM**.

Keywords: modal epistemic logic, majorities, large sets.

1 Introduction

Epistemic Logic is a very important research area, with interesting applications in Computer Science and Artificial Intelligence, along with deeply interesting philosophical issues. Revolving around the fundamental notions of **knowledge** and **belief**, it is of prime importance in *Knowledge Representation* as it is a valuable tool for modeling the epistemic state and the dynamics of a rational agent acting in a complex environment. The *logic of knowledge* and *belief* (**epistemic** and **doxastic logic**) has been successfully developed within the realm of *Modal Logic*. Its modern phase starts in 1962 with

J. Hintikka's book [17]. One of Hintikka's major contributions has been the treatment of knowledge as a modal operator, devising thus a semantics for **epistemically possible worlds**, similar to the relational semantics Kripke introduced for necessity; this work has been a real breakthrough, critically contributing to the advent of *relational semantics*, which made Modal Logic a flexible, useful tool [14].

When working with relational possible-worlds models, the modal operator is treated as a *universal quantifier*: something is *necessarily true* inside a state s iff it is true in *every* state considered as an alternative to s. In particular, when working with possible-worlds models for epistemic logic, the key concept is **indistinguishability**: the epistemic alternatives to state s are the ones the agent cannot distinguish from s by his/her knowledge. *Given the different nature of knowledge and belief, what happens if one tries to capture the distinction between the two propositional attitudes*? In a very interesting direction, a separate modal *doxastic* operator is introduced, interpreted over a different binary accessibility relation between states. Thus, both modal operators for knowledge and belief are treated as *universal quantifiers*, ranging over typically different sets of states. Here, we take a different approach which is novel and interesting from many perspectives: the KR perspective, the philosophical and the technical perspective. We introduce a **bimodal logic** for *knowledge* and *belief*, in which *knowledge* is a *universal quantifier* but *belief* is a *'majority' quantifier*. There exists a **single indistinguishability relation** between states, knowledge is classically defined as '*truth in every epistemically alternative state*' but belief is captured as '*truth in most of the epistemically alternative states*'.

Why might such an approach to knowledge and belief be interesting at the first place? Because, we claim it successfully models situations of interest in Knowledge Representation; independently, it has also a separate technical interest, as it grafts a 'majority' notion from default reasoning into epistemic logic. Assume a rational agent whose epistemic state comprises items s/he *knows* and items s/he *believes*. As it is often the case in *planning*, something is taken to be known if it holds in *every* scenario the agent can consider while something is believed if it holds in *most* of these scenarios. It is interesting to pin down the principles relating knowledge to belief in this kind of complex epistemic state. On the other hand, there exist real-life situations involving this kind of interaction between knowledge and belief. Let us consider the case of a stockbroker, reviewing the situation before selecting a stock. S/he *knows* that the P/E (Price/Earnings) ratio is a critical factor: every acceptable theory of stock selection s/he has in mind tells that a high P/E value means the stock is overvalued. On the other hand, s/he *believes* that the stock under consideration is a good value, because in most of her scenarios a stock is a good value if it is trading very close to its 52-week-low. S/he also *knows* that liquidity is low because the volume - the number of stocks bought and sold in a day of trading, is low; s/he can infer from this knowledge that this implies a lot of volatility, something s/he *believes* s/he should avoid as, in most cases, this is annoying. And, of course, s/he *believes* that this stock will offer option contracts for buying and selling in the future; most stocks do so. *Which epistemic logic can give us a formal account of this kind of situation*?

In this paper, we introduce the bimodal logic $\mathbf{KBM}$, appropriate for capturing this intuition: **knowledge as '*all*', belief as '*most*'**. Knowledge is a normal **S4** operator, belief corresponds to a non-normal one, interpreted over '*majorities*' of epistemically alternative states. To express the notion of '**majority**', we use its '**large sets**' incarnation, introduced independently by K. Schlechta [25,26] and V. Jauregui [18] in the field of nonmonotonic reasoning, in order to capture a notion of *normality-based* default entailment. *Commonsense Reasoning* has dealt with various forms of *default entailment.* A natural approach is through '*normality*': something is '*normally*' the case, if '*it holds by default*'. A candidate route to a formal definition of '*normality*' is the notion of a '*majority*': something is considered to be '*normally the case*' iff it holds in a '*majority*' (a '*significant proportion*') of the conceivably alternative situations. '*Majorities*' over a set can be conceived as '*overwhelming majorities*', '*simple majorities*', or in a more relaxed way, as its '*large subsets*'. Model Theory captures '*large subsets*' with *filters* over powerset algebras. A relaxed notion of '**weak filter**' has been independently introduced by K. Schlechta [25] and V. Jauregui [18] with the intention to capture the collections of '*large*' ('*important*') subsets of a given set, expressing thus a notion of '*majority*'. A more complex notion of '*majority space*' has been introduced and analyzed by E. Pacuit and S. Salame [23,24]. The definition of *large sets* we employ, naturally incurs a requirement for complete, consistent belief, something neither surprising nor undesirable, if one has KR applications in mind. We examine which of the 'bridge' axioms discussed in the literature, in particular with respect to the discussion in R. Stalnaker's article [28], J. Halpern's work in [16] and P. Gochet's and P. Gribomond's handbook article [13] remain valid in our framework.

In Section 2, we provide some background material, reviewing basic definitions and facts and establishing notation. Section 3 comprises the syntax, semantics, axiomatization of the logic $\mathbf{KBM}$, along with the frame completeness results and the discussion on the epistemic principles which hold (or do not hold) in this framework. Finally, we conclude in Section 4 with a short discussion on the nature of the attitude we capture here as a 'majoritarian' form of belief and some interesting questions for future research. Due to space limitations, some proofs are omitted and others are briefly sketched; the details can be found in the full report (draft version in [1]).

2 Background Material

Modal epistemic logic extends classical propositional logic. Its language $\mathcal{L}_{\mathsf{K}}$ is endowed with an epistemic operator K; if the attitude considered is *belief*, it is usually written as $\mathsf{B}\varphi$. The formula $\mathsf{K}\varphi$ reads as '*it is known that* φ' and $\mathsf{B}\varphi$ reads as '*it is believed that* φ'. For a complete treatment of modal logic, the reader can consult any of the books [2,4]. The handbook article [13] is a very good starting point for epistemic logic. The modal axiom schemata with a use in *epistemic* and/or *doxastic* logic comprise: $\mathbf{K}$. $\mathsf{K}\varphi \wedge \mathsf{K}(\varphi \supset \psi) \supset \mathsf{K}\psi$ (*Epistemic Distribution*), $\mathbf{T}$. $\mathsf{K}\varphi \supset \varphi$ (*Axiom of Knowledge*), $\mathbf{4}$. $\mathsf{K}\varphi \supset \mathsf{KK}\varphi$ (*Positive Introspection*), $\mathbf{5}$. $\neg\mathsf{K}\varphi \supset \mathsf{K}\neg\mathsf{K}\varphi$ (*Negative Introspection*), $\mathbf{D}$. $\mathsf{B}\varphi \supset \neg\mathsf{B}\neg\varphi$ (*Consistent Belief*). The axiom $\mathbf{G}$. $\neg\mathsf{K}\neg\mathsf{K}\varphi \supset \mathsf{K}\neg\mathsf{K}\neg\varphi$ does not have an obvious epistemic interpretation; however, if belief is defined as $\neg\mathsf{K}\neg\mathsf{K}\varphi$ in a monomodal epistemic language, then $\mathbf{G}$ encodes a principle of consistent belief [21]. The relationship between *knowledge* and *belief* is usually captured with bimodal languages, employing an *epistemic* (K) and a doxastic (B) modal operator. The relationship between the two propositional attitudes is pinned down by 'bridge' axioms relating the two modal operators. Some of these axioms will be discussed below in this paper.

Normal Epistemic Logics are defined as sets of $\mathcal{L}_{\mathsf{K}}$-formulas that contain classical propositional logic, all the instances of schema $\mathbf{K}$, and are closed under the rules *Uniform Substitution* and $\mathbf{MP}$. $\frac{\varphi,\varphi\supset\psi}{\psi}$ and $\mathbf{RN}$. $\frac{\varphi}{\mathsf{K}\varphi}$. Customarily, by $\mathbf{KA_1}\dots\mathbf{A_n}$ is denoted the smallest normal modal logic containing the axiom schemata $\mathbf{A_1}$ to $\mathbf{A_n}$. Some important normal epistemic logics have acquired names from the historical path of Modal Logic in the 20th Century : $\mathbf{S4}$ is $\mathbf{KT4}$, $\mathbf{S5}$ is $\mathbf{KT5}$ ($=\mathbf{KT45}$) and $\mathbf{S4.2}$ stands for $\mathbf{KT4G}$. $\mathbf{S5}$ is a strong logic of knowledge, with many important applications in Computer Science and Knowledge Representation, and logic $\mathbf{KD45}$ is considered a good candidate for the logic of consistent belief. Normal epistemic logics are interpreted over **Kripke** (**relational**) **possible-worlds models**: a *Kripke model* $\mathfrak{M}=\langle W,\mathcal{R},V\rangle$ consists of a set W of *possible worlds* (*states, situations*) and a *binary accessibility relation* between them: $\mathcal{R}\subseteq W\times W$. The very idea in possible-worlds semantics in epistemic and doxastic logic is that epistemically alternative situations are the ones which are consistent with the agent's knowledge (or belief). In that respect, $w\mathcal{R}v$ means that, in terms of the agent's knowledge, the states w and v are **indistinguishable**. In common modal logic parlance, whenever $w\mathcal{R}v$, we say that world w '*sees*' world v, or that v is an epistemic alternative state to w. Abusing notation, we denote by $\mathcal{R}(w)$ **the set of epistemically alternative states to** w: $\mathcal{R}(w)=\{v\in W\mid w\mathcal{R}v\}$. The valuation V determines which states satisfy a propositional variable. Within a state w, the propositional connectives ($\neg$, $\supset$, $\wedge$, $\vee$) are interpreted classically, while $\mathsf{K}\varphi$ is true at w iff it is true in every epistemic alternative to w (notation: $\mathfrak{M},w\Vdash\mathsf{K}\varphi$). The pair $\mathfrak{F}=\langle W,\mathcal{R}\rangle$ is called the **frame** underlying $\mathfrak{M}$. A logic Λ is *determined* by a class of frames iff it is *sound* and *complete* with respect to this class.

There exist more general semantics for modal epistemic logic: the so-called **Scott-Montague semantics** or **neighborhood semantics**. The reader can find more details in the book of B. Chellas [6] (where they are called *minimal models*), in K. Segerberg's essay [27] and the mini-course notes [22]. In *Scott-Montague models*, each state is not associated to its epistemically compatible (indistinguishable) states, but it is associated to '*neighborhoods*' (subsets) of states (or possible worlds). A **neighborhood model** is a triple $\mathfrak{N}=\langle W,\mathcal{N},V\rangle$, where W is a set of possible worlds (states), $\mathcal{N}:W\to\mathcal{P}(\mathcal{P}(W))$ is a **neighborhood function** assigning to a state the set of its '*neighborhoods*' and V is again a valuation. Inside a state, formulas of the form $\mathsf{K}\varphi$ become true at w iff the set of states $\overline{V}(\varphi)$ where φ holds (called the *truth set* of φ, $\overline{V}(\varphi)=\{v\in W\mid\mathfrak{N},v\Vdash\varphi\}$), belongs to the set of neighborhoods of w: $\overline{V}(\varphi)\in\mathcal{N}(w)$. The pair $\mathfrak{F}=\langle W,\mathcal{N}\rangle$ is called a **Scott-Montague** (neighborhood) **frame**. It is known that the smallest normal modal logic $\mathbf{K}$ is determined by the class of Scott-Montague frames, in which the neighborhood of each state is a *filter* over W [6]. Given a neighborhood model $\mathfrak{N}=\langle W,\mathcal{N},V\rangle$ and a subset X of W, we define a map $m_{\mathcal{N}}:\mathcal{P}(W)\to\mathcal{P}(W)$, where $m_{\mathcal{N}}(X)=\{w\in W\mid X\in\mathcal{N}(w)\}$. A **general Scott-Montague frame** is a triple $\mathfrak{F}=\langle W,A,\mathcal{N}\rangle$, where A is a set of *admissible* subsets of W (A contains $\varnothing$, it is closed under union, complementation in W, and the operator $m_{\mathcal{N}}$). General neighborhood frames provide a base for models, in which the propositional valuations are restricted only to admissible sets. We will be based on a stronger version of general Scott-Montague frames, used also in [9,19]: it requires that all neighborhoods should be admissible (which means that $\mathcal{N} : W \to \mathcal{P}(A)$) and, moreover, in a model, based on a general Scott-Montague frame, that the truth sets of atomic formulas also correspond to admissible sets. Then, it is easy to see that the *truth set* of every formula is an admissible set. The notion we employ for the semantics of our modal logic are based on this, but they are appropriately modified to host a notion of 'large sets'.

Notation. Given a consistent logic Λ, Λ-maximal consistent sets ($mc\Lambda$-sets) do exist, by a standard Lindenbaum argument. We denote by $[\varphi]_\Lambda$ (or by $[\varphi]$, when Λ is understood) the set of all $mc\Lambda$-sets, which contain φ. We use small-caps Greek capital characters (Γ, Δ etc.) for denoting maximal consistent sets. We use the notion of *strong completeness*: given a consistent logic Λ and a class of frames S, we say that Λ *is strongly complete with respect to* S iff for every consistent with Λ theory I and formula φ, if I entails semantically φ, then φ is deducible in Λ from I. To prove strong completeness with respect to S, it suffices to show that every consistent with Λ theory I is satisfiable in a frame from S [2].

Majorities and Large Sets. Assume a set W; it is our universe of discourse. With a view to its use below, we will call its members 'states' or 'worlds'. *Which subsets of W would we accept to represent a 'majority' of worlds*? Of course, the answer depends on whether we wish to capture an '*overwhelming majority*' (which needs more clarification), or a '*simple majority*' with a view to Social Choice or Voting Theory applications (simple majority might simply mean any subset with cardinality strictly more than $|W|/2$ for finite W, but it is absolutely non-trivial to define for infinite sets). It might be easier to define '*majority*' in terms of '*large subsets*' or '*significant fragments*' of W. In *Model Theory* [5], this is is captured by the notion of filters over W: a **filter over** (a nonempty set) W is a collection F of subsets of W, such that **(i)** $W \in F$ and $\emptyset \notin F$, **(ii)** $X \in F$ and $X \subseteq Y \subseteq W$ implies $Y \in F$ (*filters are upwards closed*), **(iii)** $X \in F$ and $Y \in F$ implies $(X \cap Y) \in F$ (*filters are closed under intersection*). This definition disallows *the* improper filter over W (which is just the whole powerset algebra). **Ultrafilters** over W are the *maximal* (proper) *filters*, or equivalently, the filters satisfying the following completeness requirement: for every $X \subseteq W$, either $X \in F$ or $(W \setminus X) \in F$.

In [18], a different definition of the '*large subsets*' of a set is introduced: a nonempty collection F of subsets of W is a collection of large subsets iff: **(i)** $X \in F$ and $X \subseteq Y \subseteq W$ implies $Y \in F$ (*F is upwards closed*), **(ii)** $X \notin F$ or $(W \setminus X) \notin F$ (*it cannot be the case that both a set and its complement are large*). In [25] a different, but provably equivalent, notion of large sets had been given: it is essentially the same with the previous one, replacing the second condition for the following one: $X \in F$ and $Y \in F$ implies that $X \cap Y \neq \emptyset$ (*There cannot be disjoint large sets*). Perhaps the more fine-grained definition of a collection F of 'majorities' has been given by E. Pacuit and S. Salame [23,24]. The motivation of this definition has to do with applications of graded modal logic and the intended interpretation of a '*majority space*' has been that of a '*simple majority*'.

3 The Logic KBM

3.1 The Language and the Axioms

We assume a bimodal propositional language $\mathcal{L}_{KB}$, built upon the usual propositional connectives ($\neg$, $\supset$, $\wedge$, $\vee$, $\equiv$) and endowed with the following modal operators: $\mathsf{K}\varphi$, read as '*the agent knows* φ', $\mathsf{B}\varphi$, read as '*the agent believes that* φ *is the case*'. The axiomatizations of **KBM** comprises the following items:

K. $\mathsf{K}\varphi \wedge \mathsf{K}(\varphi \supset \psi) \supset \mathsf{K}\varphi$ (*Consequences of knowledge constitute knowledge.*)

T. $\mathsf{K}\varphi \supset \varphi$ (*Only true things are known.*)

4. $\mathsf{K}\varphi \supset \mathsf{K}\mathsf{K}\varphi$ (*Positive introspection regarding knowledge.*)

KB. $\mathsf{K}\varphi \supset \mathsf{B}\varphi$ (*Knowledge implies belief.*)

CB. $\neg\mathsf{B}\varphi \equiv \mathsf{B}\neg\varphi$ (*Belief is consistent and complete.*)

BK. $\mathsf{B}\varphi \wedge \mathsf{K}(\varphi \supset \psi) \supset \mathsf{B}\psi$
If the agent believes φ*, and* <u>*knows*</u> *that it entails* ψ*, then she believes also* ψ*.*

PI. $\mathsf{B}\varphi \supset \mathsf{K}\mathsf{B}\varphi$ (*Positive introspection regarding belief.*)

Definition 1. **KBM** *is the modal logic axiomatized by* **K**, **T**, 4, **KB**, **CB**, **BK**,**PI** *and closed under* Uniform Substitution *and the rules* **MP**. $\frac{\varphi,\varphi\supset\psi}{\psi}$ *and* $\mathbf{RN}_{\mathsf{K}}$. $\frac{\varphi}{\mathsf{K}\varphi}$.

Some comments with respect to the axiomatization of **KBM**, are in order. We have axiomatized knowledge by **S4**, declared as a minimal acceptable epistemic basis by W. Lenzen, leaving out the strongly debated *negative introspection axiom for knowledge* and its weak variants. Regarding the 'bridge' axioms: we have adopted axiom **KB**, accepted also by W. Lenzen [20] and R. Stalnaker [28], also discussed extensively by J. Halpern with respect to the scope of its applicability [16]. The axiom **PI** is also accepted by R. Stalnaker (under the same name) and W. Lenzen. It will be shown later that the logic of the doxastic operator B is not normal, since the doxastic distribution axiom (**K** for B) is not a **KBM** theorem. In that respect, axiom **BK** is a careful application of this principle, combining both epistemic attitudes. Some more epistemic principles of this sort, are discussed below.

The axiomatization of **belief**, needs some explanation. Half of the axiom **CB** is just **D**, that is consistent belief $\mathsf{B}\neg\varphi \supset \neg\mathsf{B}\varphi$. The other half is a requirement for completeness of belief, something compatible with our interpretation of belief as '*truth in a large set of epistemically indistinguishable states*': we are committed to two-valued logic and we take into account that either a set or its complement should be regarded as majorities esp. in a finite universe[1].

We observe now that **KBM** is normal with respect to K. This means that **KBM** is also closed under the rule (cf. [6]) $\mathbf{RM}_{\mathsf{K}}$. $\frac{\varphi\supset\psi}{\mathsf{K}\varphi\supset\mathsf{K}\psi}$. On the other hand, not surprisingly, **KBM** is not normal with respect to B. The following remark asserts however that

[1] Note also that in the case of the **KD45** logic of consistent belief something weaker is valid, namely $\neg\mathsf{B}\varphi \equiv \mathsf{B}\neg\mathsf{B}\varphi$ (cf. [6, Sect. 4.4]).

$\mathbf{KBM}$ has the two normal rules mentioned, with respect to the doxastic operator B. In what follows, the notation $\mathbf{PC}$ in the (condensed) derivations, denotes one or more propositionally valid inference step(s).

Fact 1. $\mathbf{KBM}$ is closed under the rules $\mathbf{RN}_{\mathsf{B}}$. $\frac{\varphi}{\mathsf{B}\varphi}$ and $\mathbf{RM}_{\mathsf{B}}$. $\frac{\varphi \supset \psi}{\mathsf{B}\varphi \supset \mathsf{B}\psi}$.

To get a better feeling about the epistemic principles of $\mathbf{KBM}$, let us examine some 'bridge' axioms discussed in the literature of bimodal epistemic logic.

– $\mathsf{B}\varphi \supset \mathsf{BK}\varphi$

  This is Stalnaker's axiom **SB** for *strong belief* [28, p. 179], and axiom (A12) for *positive certainty* in [16]. It holds also in F. Voorbraak's OK&RIB system of *objective knowledge* and *rational introspective belief* [29].

– $\neg\mathsf{B}\varphi \supset \mathsf{K}\neg\mathsf{B}\varphi$

  This is Stalnaker's axiom **NI** for *negative introspection* [28, p. 179], discussed also in [16].

– $\neg\mathsf{B}\varphi \supset \mathsf{B}\neg\mathsf{K}\varphi$

  This is the axiom of *negative certainty* (A13) in [16]. It holds also in F. Voorbraak's OK&ORIB system [29].

The last two principles are theorems of $\mathbf{KBM}$, as the following two facts verify. The first one is not, but we will have to wait for a soundness result for our logic.

Fact 2. $\vdash_{\mathbf{KBM}} \neg\mathsf{B}\varphi \supset \mathsf{K}\neg\mathsf{B}\varphi$

Fact 3. $\vdash_{\mathbf{KBM}} \neg\mathsf{B}\varphi \supset \mathsf{B}\neg\mathsf{K}\varphi$

3.2 Semantics

Our intention now, is to provide possible-worlds semantics for our modal language, with the aim to obtain an appropriate frame characterization result for $\mathbf{KBM}$. The idea is that a binary indistinguishability relation, interpreting the epistemic operator, will be the basis. Then, we wish to obtain belief as truth in a 'majority' of the epistemically alternative states; this creates a need for defining what will be considered as a 'large set'. We have also to combine this, with the 'constraint' of the admissible sets, as we are going to employ a modified version of general Scott-Montague semantics. So, firstly, let us define a collection of sets, which can be considered as 'large sets' in our framework.

Definition 2. *Consider a set $W \neq \varnothing$ and a collection $A \subseteq \mathcal{P}(W)$. Furthermore, consider a $Z \subseteq W$ and the set $U = \{Z \cap X \mid X \in A\}$. Then, the collection $\mathbf{C} \subseteq U$ will be called an **$\mathbf{A}$-collection of majorities of $\mathbf{Z}$** iff*

$\mathbf{(m1)}$ $Z \in C$

$\mathbf{(m2)}$ $(\forall X \in A)\ \big(Z \cap X \in C \iff Z \cap (W \setminus X) \notin C\big)$

$\mathbf{(m3)}$ $(\forall T \in C)(\forall S \in U)\ \big(T \subseteq S \Longrightarrow S \in C\big)$

In order to compare Def. 2 with the notions we have described in Section 2, let us examine the case $A = \mathcal{P}(W)$ and $Z = W$. Then, the properties $(\mathbf{m1})$ to $(\mathbf{m3})$ translate into: **(i)** $W \in C$, **(ii)** $(\forall X \subseteq W)\big(X \in C \iff (W \setminus X) \notin C\big)$ and **(iii)** $(\forall T \in C)(\forall S \subseteq W)\big(T \subseteq S \Longrightarrow S \in C\big)$. In this case, the collection C contains W and it is upwards closed. These requirements describe the members of C, which are 'large' subsets of W. C contains every superset of a large set being in C; if it contains a 'large' set, its complement can not be 'large'; and finally, either a subset of W or its complement should be considered as 'large'. It is now obvious that this notion is V. Jauregui's & K. Schlechta's notion of 'large sets', augmented with a requirement of completeness. If the basic notion can be considered as a '**weak filter**', then this is a '**weak ultrafilter**'.

Next, having defined our 'large sets' (the sets we will be considering as 'majorities'), we can proceed to define possible-worlds frames, which will be relational (Kripke) with respect to the operator K, and (modified) 'general Scott-Montague' with respect to the operator B. In the next definition, note that $\mathcal{N}(w)$ plays the role of $\mathbf{C}$ in Def. 2 and the states indistinguishable from w, namely $\mathcal{R}(w)$, play the role of Z, again with respect to Def. 2.

Definition 3. *Consider a quadruple* $\mathfrak{F} = \langle W, A, \mathcal{R}, \mathcal{N}\rangle$, *where*

– $W \neq \varnothing$
– $A \subseteq \mathcal{P}(W)$ *s.t.*
  $(\mathbf{a1})$ $\varnothing \in A$
  and $(\forall X, Y \in A)$
  $(\mathbf{a2})$ $W \setminus X \in A$
  $(\mathbf{a3})$ $X \cup Y \in A$
  $(\mathbf{a4})$ $\{w \in W \mid \mathcal{R}(w) \subseteq X\} \in A$
  $(\mathbf{a5})$ $\{w \in W \mid \mathcal{R}(w) \cap X \in \mathcal{N}(w)\} \in A$
– $\mathcal{R} \subseteq W \times W$ *s.t.* $\mathcal{R}$ *is reflexive and transitive.*[2]
– $\mathcal{N} : W \to \mathcal{P}(\mathcal{P}(W))$ *s.t.* $(\forall w \in W)$
  $(\mathbf{n1})$ $\mathcal{N}(w) \subseteq \{\mathcal{R}(w) \cap X \mid X \in A\}$
  $(\mathbf{n2})$ $\mathcal{N}(w)$ *is an* A*-collection of majorities of* $\mathcal{R}(w)$
  $(\mathbf{n3})$ $(\forall X \in A)\big(\mathcal{R}(w) \cap X \in \mathcal{N}(w) \Longrightarrow \mathcal{R}(w) \subseteq \{v \in W \mid \mathcal{R}(v) \cap X \in \mathcal{N}(v)\}\big)$

Then, $\mathfrak{F}$ *is called a* ***kbm-frame***. *A model* $\mathfrak{M} = \langle \mathfrak{F}, V\rangle$ *based on a kbm-frame* $\mathfrak{F} = \langle W, A, \mathcal{R}, \mathcal{N}\rangle$, *where* $V : \Phi \to A$ *is called a* ***kbm-model***.

Fact 4. The class of all kbm-frames is nonempty.

Finally, we extend the valuation function V to all formulas in the usual way, with slight differences as far as B is concerned.

[2] We remind that $\mathcal{R}(w)$ denotes the set $\{v \in W \mid w\mathcal{R}v\}$.

Definition 4. *Let* $\overline{V} : \mathcal{L}_{KB} \to A$ *be the extension of the valuation* $V : \Phi \to A$ *to all formulas, defined recursively as follows:*

- $\overline{V}(p) = V(p), \quad (\forall p \in \Phi)$
- $\overline{V}(\bot) = \varnothing$

and $(\forall \varphi, \psi \in \mathcal{L}_{KB})$

- $\overline{V}(\varphi \supset \psi) = (W \setminus \overline{V}(\varphi)) \cup \overline{V}(\psi)$
- $\overline{V}(\mathsf{K}\varphi) = \{w \in W \mid \mathcal{R}(w) \subseteq \overline{V}(\varphi)\}$
- $\overline{V}(\mathsf{B}\varphi) = \{w \in W \mid \mathcal{R}(w) \cap \overline{V}(\varphi) \in \mathcal{N}(w)\}$

Instead of $w \in \overline{V}(\varphi)$*, we usually write* $\mathfrak{M}, w \Vdash \varphi$*.*

Remark 1. The properties $(\mathbf{a1})$ to $(\mathbf{a5})$ of Def. 3 represent exactly what is required to ensure that $(\forall \varphi \in \mathcal{L}_{KB})\ \overline{V}(\varphi) \in A$, i.e. that such a function $\overline{V}$ in Def. 4 actually exists.

After a series of formal definitions, we should check whether our constructs accurately reflect our intuitions about belief in $\mathbf{KBM}$.

Note 1. According to Definitions 3 and 4, and in light of our comments after Definition 2, note that $\mathfrak{M}, w \Vdash \mathsf{B}\varphi$ implies $\mathcal{R}(w) \cap \overline{V}(\varphi) \in \mathcal{N}(w)$ and furthermore, every member of $\mathcal{N}(w)$ is a 'large' subset of $\mathcal{R}(w)$. Hence, φ holds in the 'majority' of the $\mathcal{R}$-neighbors of w.

3.3 Soundness and Completeness of KBM

We will now prove that $\mathbf{KBM}$ is sound and complete with respect to the class of all kbm-frames.

Proposition 1. (Soundness) *The logic* $\mathbf{KBM}$ *is sound with respect to the class of all kbm-frames.*

PROOF. [*sketch*] It suffices to show that all axioms of logic $\mathbf{KBM}$ are valid in every kbm-frame. So, let $\mathfrak{M} = \langle W, A, \mathcal{R}, \mathcal{N}, V\rangle$ be any kbm-model based on $\mathfrak{F}$. Since $\mathfrak{M}$ is a Kripke-model for $\mathcal{R}$, $\mathbf{K}$ is valid in $\mathfrak{M}$, and since $\mathcal{R}$ is reflexive and transitive, axioms $\mathbf{T}$ and 4 are also valid in $\mathfrak{M}$. From the remaining axioms, we show the case of **CB**; see [1] for details. Consider an arbitrary $w \in W$.

CB. Firstly, since $\mathcal{N}(w)$ is an A-collection of majorities of $\mathcal{R}(w)$, by $(\mathbf{m2})$, Rem.1, and $(\mathbf{a2})$, $\mathcal{R}(w) \cap \overline{V}(\varphi) \in \mathcal{N}(w) \iff \mathcal{R}(w) \cap (W \setminus \overline{V}(\varphi)) \notin \mathcal{N}(w)$, hence $\mathcal{R}(w) \cap \overline{V}(\varphi) \in \mathcal{N}(w) \iff \mathcal{R}(w) \cap \overline{V}(\neg\varphi)) \notin \mathcal{N}(w) \qquad (*)$

Hence, $\mathfrak{M}, w \Vdash \neg\mathsf{B}\varphi \iff \mathcal{R}(w) \cap \overline{V}(\varphi) \notin \mathcal{N}(w) \overset{(*)}{\iff} \mathfrak{M}, w \Vdash \mathsf{B}\neg\varphi$.

∎

Having proved the soundness theorem, we may now use it to invalidate certain epistemic principles which are not theorems of $\mathbf{KBM}$. We have already mentioned the axiom $\mathsf{B}\varphi \supset \mathsf{BK}\varphi$ and the belief distribution axiom $\mathsf{B}\varphi \wedge \mathsf{B}(\varphi \supset \psi) \supset \mathsf{B}\psi$. Another interesting principle is the '*paradox of the perfect believer*' $\mathsf{BK}\varphi \supset \mathsf{K}\varphi$. This principle is derived in some axiomatic systems by the axiom $\mathbf{KB}$, the axiom $\mathbf{D}$ for *consistent belief* ($\mathsf{B}\varphi \supset \neg\mathsf{B}\neg\varphi$) and the axiom $\mathbf{5}$ for *negative introspection* regarding knowledge ($\neg\mathsf{K}\varphi \supset \mathsf{K}\neg\mathsf{K}\varphi$) [13, Section 2.4]. This principle is difficult to accept: in the presence of $\mathbf{T}$ it declares that '*we cannot believe to know a false proposition*', modeling thus a *perfect believer*. Having abandoned axiom $\mathbf{5}$ in our framework, we expect that $\mathbf{KBM}$ does not fall prey to this '*infallibility argument*'. The following fact declares that, indeed, this is the case.

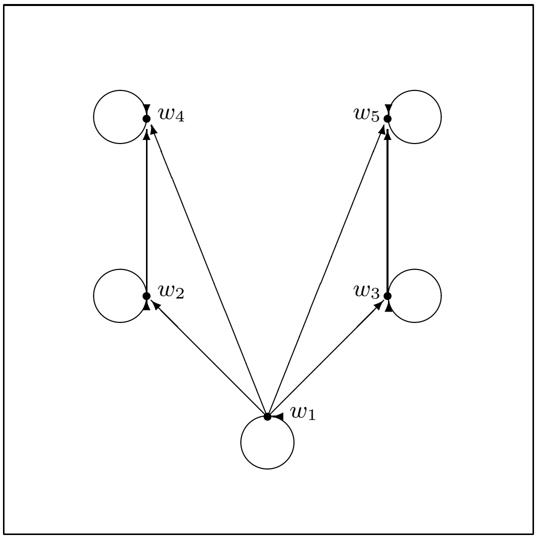


Fig. 1. The kbm-frame $\mathfrak{F}_1$

Fact 5.

i. $\nvdash_{\mathbf{KBM}} \mathsf{B}\varphi \supset \mathsf{BK}\varphi$
ii. $\nvdash_{\mathbf{KBM}} \mathsf{BK}\varphi \supset \mathsf{K}\varphi$
iii. $\nvdash_{\mathbf{KBM}} \mathsf{B}\varphi \wedge \mathsf{B}(\varphi \supset \psi) \supset \mathsf{B}\psi$

PROOF. Consider the frame $\mathfrak{F}_1 = \langle W, A, \mathcal{R}, \mathcal{N} \rangle$ of Figure 1, where $W = \{w_1, w_2, w_3, w_4, w_5\}$, $A = \big\{\varnothing, \{w_1\}, \{w_2, w_3\}, \{w_4, w_5\}, \{w_1, w_2, w_3\},, \{w_1, w_4, w_5\}, \{w_2, w_3, w_4, w_5\}, W\big\}$, $\mathcal{R}$ is as shown in Figure 1, $\mathcal{N}(w_1) = \big\{\{w_1, w_4, w_5\}, W\big\}$, $\mathcal{N}(w_2) = \big\{\{w_4\}, \{w_2, w_4\}\big\}$, $\mathcal{N}(w_3) = \big\{\{w_5\}, \{w_3, w_5\}\big\}$, $\mathcal{N}(w_4) = \big\{\{w_4\}\big\}$, $\mathcal{N}(w_5) = \big\{\{w_5\}\big\}$. With a little effort one may check that all properties of Def.3 do hold, hence $\mathfrak{F}_1$ is a kbm-frame. Consider now the model $\mathfrak{M}_1 = \langle \mathfrak{F}_1, V \rangle$ based on frame $\mathfrak{F}_1$ of Figure 1, where $V(p) = \{w_1, w_4, w_5\}$, $V(q) = \{w_1\}$, and $V(r) = \varnothing$, for all $r \in \Phi \setminus \{p, q\}$.

Then, $V : \Phi \to A$ and $\mathfrak{M}_1$ is a kbm-model. Check now that:$\mathfrak{M}_1, w_1 \Vdash \mathsf{B}p \wedge \neg\mathsf{BK}p$, $\mathfrak{M}_1, w_2 \Vdash \mathsf{BK}p \wedge \neg\mathsf{K}p$ and $\mathfrak{M}_1, w_1 \Vdash \mathsf{B}p \wedge \mathsf{B}(p \supset q) \wedge \neg\mathsf{B}q$. It follows, by Prop. 1, that the axioms above are not theorems of $\mathbf{KBM}$. ∎

We will now prove the completeness of $\mathbf{KBM}$, via canonicity. To relax our notation, we will denote $\mathbf{KBM}$ by Λ in the sequel. We proceed to define its canonical model.

Definition 5. *The* canonical model $\mathfrak{M}^\Lambda$ *for Λ is the tuple* $\langle W^\Lambda, A^\Lambda, \mathcal{R}^\Lambda, \mathcal{N}^\Lambda, V^\Lambda\rangle$,, *where*

(i) $W^\Lambda = \{\Gamma \subseteq \mathcal{L}_{KB} \mid \Gamma : mc\Lambda\}$
(the set of all maximal Λ-consistent sets)

(ii) $A^\Lambda = \{\Theta \subseteq W^\Lambda \mid (\exists\varphi \in \mathcal{L}_{KB})\ \Theta = [\varphi]\}$ [3]

(iii) $\mathcal{R}^\Lambda = \{(\Gamma, \Delta) \in W^\Lambda \times W^\Lambda \mid (\forall\varphi \in \mathcal{L}_{KB})(\mathsf{K}\varphi \in \Gamma \Rightarrow \varphi \in \Delta)\}$

(iv) $(\forall\Gamma \in W^\Lambda)$
$\mathcal{N}^\Lambda(\Gamma) = \{\mathcal{R}^\Lambda(\Gamma) \cap \Theta \subseteq W^\Lambda \mid (\exists\varphi \in \mathcal{L}_{KB})(\Theta = [\varphi]\ \&\ \mathsf{B}\varphi \in \Gamma)\}$

(v) $(\forall p \in \Phi)\ V^\Lambda(p) = [p]$

The frame $\mathfrak{F}^\Lambda$ underlying $\mathfrak{M}^\Lambda$ is called the *canonical frame* for Λ. The following lemmata are fairly standard.

Lemma 1. (Lindenbaum) *Let I be a $c\Lambda$-theory. Then, there is a $mc\Lambda$ theory Γ s.t. $I \subseteq \Gamma$.*

Lemma 2. (Existence Lemma)
$(\forall\varphi \in \mathcal{L}_{KB})(\forall\Gamma \in W^\Lambda)\ \neg\mathsf{K}\neg\varphi \in \Gamma \iff (\exists\Delta \in W^\Lambda)(\Delta \in \mathcal{R}^\Lambda(\Gamma)\ \&\ \varphi \in \Delta)$ *and equivalently* $\mathsf{K}\varphi \in \Gamma \iff (\forall\Delta \in W^\Lambda)(\Delta \in \mathcal{R}^\Lambda(\Gamma) \Rightarrow \varphi \in \Delta)$.

Lemma 3. $(\forall\Gamma \in W^\Lambda)(\forall\varphi, \psi \in \mathcal{L}_{KB})\ (\mathcal{R}^\Lambda(\Gamma) \cap [\varphi] \subseteq \mathcal{R}^\Lambda(\Gamma) \cap [\psi]\ \&\ \mathsf{B}\varphi \in \Gamma) \implies \mathsf{B}\psi \in \Gamma$

Lemma 4. (Truth Lemma) $(\forall\Gamma \in W^\Lambda)(\forall\varphi \in \mathcal{L}_{KB})\ \mathfrak{M}^\Lambda, \Gamma \Vdash \varphi \iff \varphi \in \Gamma$

Theorem 6. (Completeness) *The logic* $\mathbf{KBM}$ *is strongly complete with respect to the class of all kbm-frames.*

PROOF. [*sketch*] We prove that every consistent with Λ theory is satisfied in $\mathfrak{F}^\Lambda$. It suffices to show that $\mathfrak{F}^\Lambda$ is a kbm-frame. By the soundness of $\mathbf{KBM}$ with respect to the nonempty (Fact 4) class of kbm-frames, $\mathbf{KBM}$ is consistent, hence, $\{\top\}$ is a $c\Lambda$ theory, and, by Lemma 1, there exists a $mc\Lambda$ theory containing $\{\top\}$, so, $W^\Lambda \neq \varnothing$. $A^\Lambda \subseteq \mathcal{P}(W^\Lambda)$, the proofs for $(\mathbf{a1})$ to $(\mathbf{a5})$ are omitted, see [1] for the details.
As an indication of the spirit of the proof, we will provide a sketch of for $(\mathbf{n2})$. Consider any $\Gamma \in W^\Lambda$. We have to verify that $\mathcal{N}^\Lambda(\Gamma)$ is an A^Λ-collection of majorities of $\mathcal{R}^\Lambda(\Gamma)$. [$(\mathbf{m1})$]: $\top \in \Lambda$, so, by Remark 1, $\mathsf{B}\top \in \Lambda$, hence, $\mathsf{B}\top \in \Gamma$, therefore, by Def.5(iv), $\mathcal{R}^\Lambda(\Gamma) \cap [\top] \in \mathcal{N}^\Lambda(\Gamma)$. But, $[\top] = W^\Lambda$, so, $\mathcal{R}^\Lambda(\Gamma) = \mathcal{R}^\Lambda(\Gamma) \cap [\top] \in \mathcal{N}^\Lambda(\Gamma)$. [$(\mathbf{m2})$] is omitted,see [1]. [$(\mathbf{m3})$]: Consider a $T \in \mathcal{N}^\Lambda(\Gamma)$. Then, there is a formula φ s.t. $T = \mathcal{R}^\Lambda(\Gamma) \cap [\varphi]$ and $\mathsf{B}\varphi \in \Gamma$. Furthermore, let $S = \mathcal{R}^\Lambda(\Gamma) \cap [\psi]$ (for some formula ψ) be s.t. $T \subseteq S$. Then, by Lemma 3, $\mathsf{B}\psi \in \Gamma$, i.e., by Def.5(iv), $S = \mathcal{R}^\Lambda(\Gamma) \cap [\psi] \in \mathcal{N}^\Lambda(\Gamma)$. ∎

[3] We remind that $[\varphi] = \{\Delta \in W^\Lambda \mid \varphi \in \Delta\}$.

4 Conclusions and Future Work

In this paper, we have combined epistemic and doxastic logic with a notion of '*majority*' imported from default reasoning. What is really missing (and has been brought to our attention by an extremely insightful referee report) is a discussion on whether this kind of '*majoritarian*' belief is really a sort of belief. Being motivated by Knowledge Representation (and a notion of *belief*, loosely described as '*observable evidence*') we have not elaborated on the relation of this attitude to (a form of) '*acceptance*'. Of course, there is a vivid discussion in the philosophical literature [7,10,3] on the exact differences between the two notions of 'belief' and some aspects of this discussion are intimately connected to the KR scenarios of our motivation: *Planning* and (inspiration from) *default reasoning* [3]. There exist different perspectives on *belief* and *acceptance*; for instance P. Engel writes in [10] that "*I argue, as a number of other writers argue, that one should distinguish* ***a variety of belief-like attitudes***: ***believing proper*** *- a dispositional state which can have degrees -* ***holding true*** *- which can occur without understanding what one believes - and* ***accepting*** *- a practical and contextual attitude that has a role in deliberation and in practical reasoning*" (emphasis is ours), adopting thus acceptance as a form of belief. It would be very interesting and fruitful, both from the Philosophy and the KR perspective, to axiomatize logics focusing on a 'majoritarian' treatment of more fine-grained, belief-like attitudes. Motivated by this, we are currently pursuing an $\mathbf{S4.2}$-based epistemic logic (with a doxastic modality defined through the basic epistemic modality), enriched with a '*majoritarian*' modality of '*estimation*' (rather than acceptance) that certain facts hold. Of course, for all these logics, it should be interesting **(i)** to identify their computational complexity and **(ii)** to define and work with 'group' notions of belief, such as the collective attitudes discussed in [12,15] Another, *very interesting* topic of future research is to identify the relation of $\mathbf{KBM}$ to *probabilistic beliefs* and *probabilistic epistemic logics*, in particular with respect to possible applications in *decision systems*. This seems to be a much deeper question; as suggested to us, the '*belief as majority*' view calls immediately for a comparison to the '*belief held with probability strictly more than 0.5*' view. It is definitely a very natural and interesting avenue of future research.

Acknowledgments. We wish to thank an anonymous JELIA 2012 referee for providing a series of deep and insightful comments on the form of '*belief*' captured in this paper, along with some pointers in the philosophy literature and very natural questions for future research. Actually, the previous section has been entirely rewritten to include the suggestions of the referee; as said above, some interesting research on the above topics has already been launched. The third author gratefully acknowledges financial support by the *Greek Ministry of Education, Lifelong Learning and Religious Affairs* under the scheme of educational leave.

References

1. Askounis, D., Koutras, C.D., Zikos, Y.: Knowledge means 'all', belief means most. Technical Report, draft version (May 2012), `http://www.uop.gr/~ckoutras`, `http://users.uop.gr/%7Eckoutras/AKZ-KBM-Full.pdf`

2. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press (2001)
3. Bratman, M.: Practical reasoning and acceptance in a context. Mind 101(401), 1–16 (1992)
4. Chagrov, A., Zakharyashev, M.: Modal Logic. Oxford Logic Guides, vol. 35. Oxford University Press (1997)
5. Chang, C.C., Keisler, H.J.: *Model Theory*, 3rd edn. Studies in Logic and the Foundations of Mathematics, vol. 73. North-Holland, Amsterdam (1990)
6. Chellas, B.F.: Modal Logic, an Introduction. Cambridge University Press (1980)
7. Cohen, L.: An essay on Belief and Acceptance. Oxford University Press (1995)
8. Dubois, D., Welty, C.A., Williams, M.-A. (eds.): Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR 2004), Whistler, Canada, June 2-5. AAAI Press (2004)
9. Dŏsen, K.: Duality between modal algebras and neighbourhood frames. Studia Logica. 48(2), 219–234 (1989)
10. Engel, P.: Believing, holding true, and accepting. Philosophical Explorations: An International Journal for the Philosophy of Mind and Action 1(2), 140–151 (1998)
11. Gabbay, D.M., Woods, J. (eds.): Logic and the Modalities in the Twentieth Century. Handbook of the History of Logic, vol. 7. North-Holland (2006)
12. Gilbert, M.P.: Modeling collective belief. Synthese 73, 185–204 (1987)
13. Gochet, P., Gribomont, P.: Epistemic logic. In: Gabbay and Woods [11], vol. 7, pp. 99–195 (2006)
14. Goldblatt, R.: Mathematical Modal Logic: A View of its Evolution. In: Gabbay and Woods [11], vol. 7, pp. 1–98 (2006)
15. Hakli, R.: Group beliefs and the distinction between belief and acceptance. Cognitive Systems Research 7(2-3), 286–297 (2006)
16. Halpern, J.: Should knowledge entail belief? Journal of Philosophical Logic 25(5), 483–494 (1996)
17. Hintikka, J.: Knowledge and Belief: an Introduction to the Logic of the two notions. Cornell University Press, Ithaca (1962)
18. Jauregui, V.: Modalities, Conditionals and Nonmonotonic Reasoning. PhD thesis, Department of Computer Science and Engineering, University of New South Wales (2008)
19. Kracht, M., Wolter, F.: Normal modal logics can simulate all others. Journal of Symbolic Logic 64(1), 99–138 (1999)
20. Lenzen, W.: Recent Work in Epistemic Logic. North-Holland (1978)
21. Lenzen, W.: Epistemologische Betrachtungen zu [S4,S5]. Erkenntnis 14, 33–56 (1979)
22. Pacuit, E.: Neighborhood semantics for modal logic: an introduction. Course Notes for ESSLLI 2007 (2007)
23. Pacuit, E., Salame, S.: Majority logic. In: Dubois, et al. [8], pp. 598–605
24. Salame, S.: Majority Logic and Majority Spaces in contrast with Ultrafilters. PhD thesis, Graduate Center, City University of New York (2006)
25. Schlechta, K.: Defaults as generalized quantifiers. Journal of Logic and Computation 5(4), 473–494 (1995)
26. Schlechta, K.: Filters and partial orders. Logic Journal of the IGPL 5(5), 753–772 (1997)
27. Segerberg, K.: An essay in Clasical Modal Logic. Filosofiska Studies, Uppsala (1971)
28. Stalnaker, R.: On logics of knowledge and belief. Philosophical Studies 128(1), 169–199 (2006)
29. Voorbraak, F.: As Far as I Know - Epistemic Logic and Uncertainty. PhD thesis, Department of Philosophy, Utrecht University (1993)

Generalized DEL-Sequents

Guillaume Aucher[1], Bastien Maubert[2], and François Schwarzentruber[3]

[1] University of Rennes 1 - INRIA, France
guillaume.aucher@irisa.fr
[2] University of Rennes 1, France
bastien.maubert@irisa.fr
[3] ENS Cachan, France
francois.schwarzentruber@bretagne.ens-cachan.fr

Abstract. Let us consider a sequence of formulas providing partial information about an initial situation, about a set of events occurring sequentially in this situation, and about the resulting situation after the occurrence of each event. From this whole sequence, we want to infer more information, either about the initial situation, or about one of the events, or about the resulting situation after one of the events. Within the framework of Dynamic Epistemic Logic (DEL), we show that these different kinds of problems are all reducible to the problem of inferring what holds in the final situation after the occurrence of all the events. We then provide a tableau method deciding whether this kind of inference is valid. We implement it in LotrecScheme and show that these inference problems are NEXPTIME-complete. We extend our results to the cases where the accessibility relation is serial and reflexive and illustrate them with the coordinated attack problem.

1 Introduction

Assume that a sequence of n events has occured in a situation. We have some information about each event and about the resulting situation after the occurrence of each event, in the form of a sequence of formulas φ'_i and φ_i respectively:

$$\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_i, \varphi_i, \ldots, \varphi'_n, \varphi_n$$

Our aim is to infer some more information about one of the events or about one of the resulting situations from the rest of information provided by this sequence. This defines respectively two different kinds of inference problems.

In the first kind of inference problem, we want to infer more information about the i^{th} resulting situation. That is, given a formula ψ describing (incompletely) a situation, we want to verify whether or not we can infer that this property ψ necessarily held at the i^{th} resulting situation:

$$\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \models^{1}_{i} \psi \text{ ?}$$

In the second kind of inference problem, we want to infer more information about the i^{th} event. That is, given a formula ψ' describing (incompletely) this event, we want to verify whether or not we can infer that this property ψ' necessarily held during the occurrence of the i^{th} event:

$$\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \models^2_i \psi' \ ?$$

Solving these problems is relevant for *dynamic diagnosis* for instance (see [10] for an early survey of dynamic diagnosis). In this field, one is interested in looking for and verifying that plausible diagnoses of a faulty system 'fit' a given history that contains some abnormal behaviours. A diagnosis is a series of faults together with the time when they occur. Within our setting, this verification problem amounts to deciding about the satisfiability of a sequence of formulas.

We moreover assume that our situations involve several agents and we are interested in reasoning about the beliefs and knowledge of these agents. Our formulas φ'_i and φ_i will therefore express beliefs of several agents about events and about the resulting situations. For these reasons, we address our two problems above within the framework of Dynamic Epistemic Logic (DEL for short), since this logical framework is very well suited to express and reason about the beliefs of several agents in a dynamic setting.

The paper is organised as follows. In Section 2, we recall the core of dynamic epistemic logic and define our two kinds of inference problems. We show that these two kinds of inference problems are actually both reducible to the problem of inferring what holds in the final situation after the occurrence of all the events from the rest of the sequence. In Section 3, we provide a terminating, sound and complete tableau method that decides whether this kind of inference is valid. In Section 4, we show that our tableau method is optimal, first by proving that it is in NEXPTIME, and then by proving that our inference problem is NEXPTIME-hard. We also provide in this section a link to an implementation of our tableau method in LotrecScheme together with some details about its implementation. In Section 5, we extend our results to richer semantics where the accessibility relations are reflexive and serial. In Section 6, we apply our generalized DEL-sequents to the coordinated attack problem of the distributed system literature. Finally, in Section 7, we discuss some related work, and then conclude.[1]

2 Dynamic Epistemic Logic: DEL-Sequents

Following the methodology of DEL, we split the exposition of the logical framework into three subsections. We then define our generalized DEL-sequents in Section 2.4.

2.1 Representation of the Initial Situation: $\mathcal{L}$-Model

In the rest of this paper, Φ is a countably infinite set of propositional letters called *atomic facts* which describe static situations, and Agt is a finite set of agents.

[1] Note that all the proofs of this paper are available in a Technical Report at the following address: `http://hal.inria.fr/hal-00716074`.

An $\mathcal{L}$*-model* is a tuple $\mathcal{M} = (W, R, V)$ where:

- W is a non-empty set of possible worlds,
- $R : Agt \to 2^{W \times W}$ is a function assigning to each agent $a \in Agt$ an accessibility relation on W,
- $V : \Phi \to 2^W$ is a function assigning to each propositional letter of Φ a subset of W. The function V is called a valuation.

We write $w \in \mathcal{M}$ for $w \in W$, and $(\mathcal{M}, w)$ is called a pointed $\mathcal{L}$-model (w often represents the actual world). If $w, v \in W$, we write wR_av for $R(a)(w, v)$ and $R_a(w) = \{v \in W \mid wR_av\}$. Intuitively, wR_av means that in world w agent a considers that world v might be the actual world.

Then, we define the following epistemic language $\mathcal{L}$ that can be used to describe and state properties of $\mathcal{L}$-models:

$$\mathcal{L} : \varphi \quad ::= \quad p \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \varphi \quad | \quad B_a\varphi$$

where p ranges over Φ and a over Agt. We define $\varphi \vee \psi \overset{\text{def}}{=} \neg(\neg\varphi \wedge \neg\psi)$ and $\langle B_a \rangle\varphi \overset{\text{def}}{=} \neg B_a \neg\varphi$. The symbol $\top$ is an abbreviation for $p \vee \neg p$ for a chosen $p \in \Phi$. Let $\mathcal{M}$ be an $\mathcal{L}$-model, $w \in \mathcal{M}$ and $\varphi \in \mathcal{L}$. $\mathcal{M}, w \models \varphi$ is defined inductively as follows:

$$\begin{array}{lll}
\mathcal{M}, w \models p & \text{iff} & w \in V(p) \\
\mathcal{M}, w \models \neg\varphi & \text{iff} & \text{not } \mathcal{M}, w \models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models B_a\varphi & \text{iff} & \text{for all } v \in R_a(w),\ \mathcal{M}, v \models \varphi
\end{array}$$

The formula $B_a\varphi$ reads as "agent a believes φ". Its truth conditions are defined in such a way that agent a believes φ holds in a possible world when φ holds in all the worlds agent a considers possible. Dually, the formula $\langle B_a \rangle\varphi$ reads as "agent a considers φ is plausible". Agent a considers that φ is plausible in a possible world when φ holds in at least one of the worlds agent a considers possible.

2.2 Representation of the Event: $\mathcal{L}'$-Model

The propositional letters p' describing events are called *atomic events* and range over an infinite set Φ'. To each atomic event p', we assign a formula of the language $\mathcal{L}$, which is called the *precondition of* p'. This precondition corresponds to the property that should be true in any world w of an $\mathcal{L}$-model so that the atomic event p' can 'physically' occur in this world w. To do so we define a surjection $\text{Pre} : \Phi' \to \mathcal{L}$ that is called the *precondition function.* We take it surjective so that we have an atomic event for every possible precondition.

An $\mathcal{L}'$*-model* is a tuple $\mathcal{M}' = (W', R', V')$ where:

- W' is a non-empty set of possible events,
- $R' : Agt \to 2^{W' \times W'}$ is a function assigning to each agent $a \in Agt$ an accessibility relation on W',

– $V' : \Phi' \to 2^{W'}$ is a function assigning to each propositional letter of Φ' a subset of W' such that for all $w' \in W'$, there is at most one p' such that $w' \in V'(p')$ (Exclusivity).

We write $w' \in \mathcal{M}'$ for $w' \in W'$, and $(\mathcal{M}', w')$ is called a *pointed* $\mathcal{L}'$*-model* and w' represents the actual event of $(\mathcal{M}', w')$. If $w', u' \in W'$, we write $w'R'_a u'$ for $R'(a)(w', u')$ and $R'_a(w') = \{u' \in W' \mid w'R'_a u'\}$. Intuitively, $u' \in R'_a(w')$ means that while the possible event represented by w' is occurring, agent a considers possible that the possible event represented by u' is actually occurring. Our definition of an $\mathcal{L}'$-model is equivalent to the definition of an action signature in the logical framework of [6].[2]

Just as we defined a language $\mathcal{L}$ for $\mathcal{L}$-models, we also define a language $\mathcal{L}'$ for $\mathcal{L}'$-models ($\mathcal{L}'$ was already introduced in [8]):

$$\mathcal{L}' : \varphi' \quad ::= \quad p' \quad \mid \quad \neg\varphi' \quad \mid \quad \varphi' \wedge \varphi' \quad \mid \quad B_a\varphi'$$

where p' ranges over Φ' and a over Agt. In the sequel, formulas of $\mathcal{L}'$ are always indexed by the quotation mark $'$, unlike formulas of $\mathcal{L}$. The truth conditions of the language $\mathcal{L}'$ are identical to the ones of the language $\mathcal{L}$. Let $\mathcal{M}'$ be an $\mathcal{L}'$-model, $w' \in \mathcal{M}'$ and $\varphi' \in \mathcal{L}'$. $\mathcal{M}', w' \models \varphi'$ is defined inductively as follows:

$$\begin{array}{lll}
\mathcal{M}', w' \models p' & \text{iff} & w' \in V'(p') \\
\mathcal{M}', w' \models \neg\varphi' & \text{iff} & \text{not } \mathcal{M}', w' \models \varphi' \\
\mathcal{M}', w' \models \varphi' \wedge \psi' & \text{iff} & \mathcal{M}', w' \models \varphi' \text{ and } \mathcal{M}', w' \models \psi' \\
\mathcal{M}', w' \models B_a\varphi' & \text{iff} & \text{for all } u' \in R'_a(w'),\ \mathcal{M}', u' \models \varphi'
\end{array}$$

2.3 Update of the Initial Situation by the Event: Product Update

The precondition function Pre of the previous section induces a precondition function for $\mathcal{L}'$-models, which assigns to each possible event w' of an $\mathcal{L}'$-model a formula $\text{Pre}(w')$ of $\mathcal{L}$. The precondition function induced by the $\mathcal{L}'$-model $\mathcal{M}' = (W', R', V')$ is defined as follows: $\text{Pre}(w') = \text{Pre}(p')$ if there is p' such that $\mathcal{M}', w' \models p'$; $\text{Pre}(w') = \top$ otherwise.

We then redefine equivalently in our setting the BMS product update of [7] as follows. Let $(\mathcal{M}, w) = (W, R, V, w)$ be a pointed $\mathcal{L}$-model and let $(\mathcal{M}', w') = (W', R', V', w')$ be a pointed $\mathcal{L}'$-model such that $\mathcal{M}, w \models \text{Pre}(w')$. The *product update of* $(\mathcal{M}, w)$ *and* $(\mathcal{M}', w')$ is the pointed $\mathcal{L}$-model $(\mathcal{M}, w) \otimes (\mathcal{M}', w') = (W^\otimes, R^\otimes, V^\otimes, (w, w'))$ defined as follows:

$$\begin{aligned}
W^\otimes &= \{(u, u') \in W \times W' \mid \mathcal{M}, u \models \text{Pre}(u')\}, \\
R^\otimes_a(u, u') &= \{(v, v') \in W^\otimes \mid v \in R_a(u) \text{ and } v' \in R'_a(u')\}, \\
V^\otimes(p) &= \{(u, u') \in W^\otimes \mid \mathcal{M}, u \models p\}.
\end{aligned}$$

[2] Let $\Sigma = (W', R', (w'_1, \ldots, w'_n))$ be an action signature and let $\varphi_1, \ldots, \varphi_n \in \mathcal{L}$. The $\mathcal{L}'$*-model associated to* $(\Sigma, \varphi_1, \ldots, \varphi_n)$ is the tuple $M' = (W', R', V')$ where the valuation V' is defined as follows. We pick $q' \in \Phi'$ such that $Pre(q') = \top$, and for all $i \in \{1, \ldots, n\}$, we pick $p'_i \in \Phi'$ such that $Pre(p'_i) = \varphi_i$. Then, for all $i \in \{1, \ldots, n\}$ we set $V'(p'_i) = \{w'_i\}$, we also set $V'(q') = W' - \{w'_1, \ldots, w'_n\}$, and for all $p' \in \Phi' - \{q', p'_1, \ldots, p'_n\}$ we set $V'(p') = \emptyset$.

This product update yields a new $\mathcal{L}$-model $(\mathcal{M}, w) \otimes (\mathcal{M}', w')$ representing how the new situation which was previously represented by $(\mathcal{M}, w)$ is perceived by the agents after the occurrence of the event represented by $(\mathcal{M}', w')$.

2.4 Generalized DEL-Sequents

In this section we define two different inference relations on formulas of $\mathcal{L}$ and $\mathcal{L}'$ representing an initial situation, a series of events and resulting epistemic situations. One relation enables to infer information about one of the epistemic situations, the other about one of the events:

Definition 1. *Let $\varphi_0, \varphi_1, \ldots, \varphi_n, \psi \in \mathcal{L}$, $\varphi'_1, \ldots, \varphi'_n, \psi' \in \mathcal{L}'$. We define the logical consequence relations $\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \models^1_k \psi$ where $0 \leq k \leq n$, and $\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \models^2_k \psi'$ where $1 \leq k \leq n$, as follows:*

$$\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \models^1_k \psi \textit{ if}$$

for all pointed $\mathcal{L}$-models $(\mathcal{M}_0, w_0)$ and $\mathcal{L}'$-models $(\mathcal{M}'_j, w'_j)$, if we have for all $i \in \{0, \ldots, n\}$ that $(\mathcal{M}_i, w_i) = (\mathcal{M}_0, w_0) \otimes (\mathcal{M}'_1, w'_1) \otimes \ldots \otimes (\mathcal{M}'_i, w'_i)$ is defined[3] *and $\mathcal{M}_i, w_i \models \varphi_i$, and for all $j \in \{1, \ldots, n\}$ that $\mathcal{M}'_j, w'_j \models \varphi'_j$, then $\mathcal{M}_k, w_k \models \psi$.*

$$\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \models^2_k \psi' \textit{ if}$$

for all pointed $\mathcal{L}$-models $(\mathcal{M}_0, w_0)$ and $\mathcal{L}'$-models $(\mathcal{M}'_j, w'_j)$, if we have for all $i \in \{0, \ldots, n\}$ that $(\mathcal{M}_i, w_i) = (\mathcal{M}_0, w_0) \otimes (\mathcal{M}'_1, w'_1) \otimes \ldots \otimes (\mathcal{M}'_i, w'_i)$ is defined and $\mathcal{M}_i, w_i \models \varphi_i$, and for all $j \in \{1, \ldots, n\}$ that $\mathcal{M}'_j, w'_j \models \varphi'_j$, then $\mathcal{M}'_k, w'_k \models \psi'$.

In fact, those two DEL-sequent relations are interdefinable:

Proposition 1. *For all $\varphi_0, \ldots, \varphi_n \in \mathcal{L}$, $\varphi'_1, \ldots, \varphi'_n, \psi' \in \mathcal{L}'$ and $k \in \{1, \ldots, n\}$,*

$$\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \models^2_k \psi' \quad \textit{iff} \quad \varphi_0, \ldots, \varphi'_k \wedge \neg\psi', \top, \varphi'_{k+1}, \ldots, \varphi_n \models^1_k \neg\varphi_k$$

Moreover, the first relation can always be reduced to the case where information is inferred about the last situation:

Proposition 2. *For all $\varphi_0, \ldots, \varphi_n \in \mathcal{L}$, $\varphi'_1, \ldots, \varphi'_n \in \mathcal{L}'$, $k < n$ and $\psi \in \mathcal{L}$,*

$$\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \models^1_k \psi \quad \textit{iff} \quad \varphi_0, \ldots, \varphi_k \wedge \neg\psi, \ldots, \varphi'_n, \top \models^1_n \neg\varphi_n$$

Considering Propositions 1 and 2, in the rest of the paper we will only provide a tableau method for the DEL-sequent $\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \models^1_n \psi$.

In [3], we defined three kinds of logical consequence relations dealing with a single event, which are special cases of the general relations defined here. Let $\varphi_0, \varphi_1 \in \mathcal{L}$ and $\varphi' \in \mathcal{L}'$. It holds that:

$$\varphi_0, \varphi' \models \varphi_1 \text{ iff } \varphi_0, \varphi', \top \models^1_1 \varphi_1 \qquad \varphi', \varphi_1 \models^3 \varphi_0 \text{ iff } \top, \varphi', \varphi_1 \models^1_0 \varphi_0$$

$$\varphi_0, \varphi_1 \models^2 \varphi' \text{ iff } \varphi_0, \top, \varphi_1 \models^2_1 \varphi'$$

[3] When $i = 0$ we let $(\mathcal{M}_0, w_0) \otimes (\mathcal{M}'_1, w'_1) \otimes \ldots \otimes (\mathcal{M}'_i, w'_i) = (\mathcal{M}_0, w_0)$.

3 Tableau Method

We consider $2n+2$ formulas, $\varphi_0, \dots, \varphi_n, \psi \in \mathcal{L}$ and $\varphi'_1, \dots, \varphi'_n \in \mathcal{L}'$, and by $\mathcal{P} \subset \Phi'$ we denote the finite set of all atomic events appearing in one of the event formulas $\varphi'_1, \dots, \varphi'_n$. i ranges over $\{0, \dots, n\}$ and j ranges over $\{1, \dots, n\}$. We want to address the problem of deciding whether $\varphi_0, \varphi'_1, \varphi_1, \dots, \varphi'_n, \varphi_n \models_{\frac{1}{n}} \psi$ holds. To do so we equivalently decide whether this does not hold, *i.e* whether there exist a pointed $\mathcal{L}$-model $(\mathcal{M}_0, w_0)$ and n pointed $\mathcal{L}'$-models $(\mathcal{M}'_j, w'_j)$ such that for all i, $(\mathcal{M}_i, w_i) = (\mathcal{M}_0, w_0) \otimes (\mathcal{M}'_1, w'_1) \otimes \dots \otimes (\mathcal{M}'_i, w'_i)$ is defined and $\mathcal{M}_i, w_i \models \varphi_i$, for all j $\mathcal{M}'_j, w'_j \models \varphi'_j$, and $\mathcal{M}_n, w_n \models \neg\psi$. In other terms, deciding whether $\varphi_0, \varphi'_1, \varphi_1, \dots, \varphi'_n, \varphi_n \models_{\frac{1}{n}} \psi$ holds reduces to the following problem called the *satisfiability problem*:

- Input: $\varphi_0, \dots, \varphi_n, \psi \in \mathcal{L}$, $\varphi'_1, \dots, \varphi'_n \in \mathcal{L}'$ and $\mathrm{Pre}_{|\mathcal{P}}$, the restriction of Pre to the domain $\mathcal{P}$.
- Output: yes iff there exist a pointed $\mathcal{L}$-model $(\mathcal{M}_0, w_0)$ and n pointed $\mathcal{L}'$-models $(\mathcal{M}'_j, w'_j)$ such that for all j, $\mathcal{M}'_j, w'_j \models \varphi'_j$, for all i, $(\mathcal{M}_i, w_i) = (\mathcal{M}_0, w_0) \otimes (\mathcal{M}'_1, w'_1) \otimes \dots \otimes (\mathcal{M}'_i, w'_i)$ is defined and $\mathcal{M}_i, w_i \models \varphi_i$, and $\mathcal{M}_n, w_n \models \psi$.

In the rest of the paper, when the initial pointed model $(\mathcal{M}_0, w_0)$ and the pointed event models $(\mathcal{M}'_i, w'_i)$ are clear from the context, we shall write $\mathcal{M}_i$ for $\mathcal{M}_0 \otimes \mathcal{M}'_1 \otimes \dots \otimes \mathcal{M}'_i$ and w_i for $(w_0, w'_1, \dots, w'_i)$.

3.1 Tableau Method Description

Let $\mathfrak{Lab}$ be a countable set of labels designed to represent worlds of the initial epistemic model. For all integers i, let $\mathfrak{Lab}_i$ be a countable set of labels designed to represent events of the i^{th} event model. We suppose that $\mathfrak{Lab}$ and $\mathfrak{Lab}_i$ for all i are disjoint.

Our tableau method manipulates terms that we call *tableau terms* and they are of the following kind:

- $(\sigma\ \varphi)$ means that $\mathcal{M}_0, w_0 \models \varphi$, where w_0 is the world of the initial epistemic model $\mathcal{M}_0$ represented by σ and φ is a formula of $\mathcal{L}$;
- $(\sigma_i\ \varphi')$ means that $\mathcal{M}'_i, w'_i \models \varphi'$, where w'_i is the event of the i^{th} event model $\mathcal{M}'_i$ represented by σ_i, and φ' is a formula of $\mathcal{L}'$;
- $(\sigma\ \sigma_1 \dots\ \sigma_i\ \varphi)$ means that $\mathcal{M}_i, (w_0, w'_1, \dots, w'_i) \models \varphi$, where w_0 is the world of $\mathcal{M}_0$ represented by σ, $w'_k \in \mathcal{M}'_k$ is the event represented by σ_k, and φ is a formula of $\mathcal{L}$;
- $(\sigma\ \sigma_1 \dots\ \sigma_i\ 0)$ means that $(w_0, w'_1, \dots, w'_i)$ is not in $\mathcal{M}_i$, where w_0 is the world of $\mathcal{M}_0$ represented by σ and $w'_k \in \mathcal{M}'_k$ is the event represented by σ_k;
- $(R_a\ \sigma\ \sigma')$ means that the worlds w and u of $\mathcal{M}_0$ represented respectively by σ and σ' are such that wR_au;
- $(R^i_a\ \sigma_i\ \sigma'_i)$ means that in $\mathcal{M}'_i$, the events w' and u' represented by σ_i and σ'_i are such that $w'{R'}^i_a u'$;
- $\perp$ denotes an inconsistency.

We also use *generic labels* Σ to simplify notations: Σ can be either σ, σ_i or $\sigma\ \sigma_1\ \ldots\ \sigma_i$. $(R_a\ \Sigma\ \Sigma')$ is interpreted in the following way: if $\Sigma = \sigma$, $\Sigma' = \sigma'$ then $(R_a\ \Sigma\ \Sigma')$ denotes $(R_a\ \sigma\ \sigma')$, if $\Sigma = \sigma_i$, $\Sigma' = \sigma'_i$ then $(R_a\ \Sigma\ \Sigma')$ denotes $(R^i_a\ \sigma_i\ \sigma'_i)$, and if $\Sigma = \sigma\ \sigma_1\ \ldots\ \sigma_i$, $\Sigma' = \sigma'\ \sigma'_1\ \ldots\ \sigma'_i$ then $(R_a\ \Sigma\ \Sigma')$ denotes $(R_a\ \sigma\ \sigma')(R^1_a\ \sigma_1\ \sigma_1)\ldots(R^i_a\ \sigma_i\ \sigma_i)$.

A *tableau rule* is represented by a *numerator* $\mathcal{N}$ above a line and a finite list of *denominators* $\mathcal{D}_1, \ldots, \mathcal{D}_k$ below this line, separated by vertical bars:

$$\frac{\mathcal{N}}{\mathcal{D}_1 \mid \ldots \mid \mathcal{D}_k}$$

The numerator and the denominators are finite sets of tableau terms.

A *tableau for* a tuple $(\varphi_0, \ldots, \varphi_n, \varphi'_1, \ldots, \varphi'_n)$ of formulas is a finite tree with a set of tableau terms at each node. The root is $\Gamma^0 = \{(\sigma\ \ \varphi_0), (\sigma\ \ \sigma_1\ \ \varphi_1), \ldots, (\sigma\ \ \sigma_1\ \ \ldots\ \ \sigma_n\ \ \varphi_n), (\sigma_1\ \ \varphi'_1), \ldots, (\sigma_n\ \ \varphi'_n)\} \cup \{(\sigma_1\ p'^{+}_1), \ldots, (\sigma_n\ p'^{+}_n)\}$, where $p'^{+}_1, \ldots, p'^{+}_n$ are fresh new atomic events. Indeed, in our tableau method, when a new event label σ_i is added to the set Γ of a node, it is assigned a fresh new atomic event p'^{+}. By fresh we mean that they do not appear in any event formula φ' in Γ, and their precondition is also a fresh new atomic proposition that appears neither in any formula nor in any precondition of any atomic event p' of Γ. We denote it by $\text{Pre}(p'^{+}) = p^{+}$. We abuse notations by writing $p' \in \Gamma$ whenever p' occurs in a formula appearing in a tableau term in Γ. Those additional fresh atomic propositions and events are used in the tableau method to avoid the problematic case of events being built with trivial precondition.

A rule with numerator $\mathcal{N}$ is *applicable* to a node carrying a set Γ if Γ contains an instance of $\mathcal{N}$. If no rule is applicable, Γ is said to be *saturated.* We call a node n an *end node* if the set of formulas Γ it carries is saturated, or if $\bot \in \Gamma$. The tableau is extended as follows:

1. Choose a leaf node n carrying Γ where n is not an end node, and choose a rule ρ applicable to n.
2. (a) If ρ has only one denominator, add the appropriate instanciation to Γ.
   (b) If ρ has k denominators with $k > 1$, create k successor nodes for n, where each successor i carries the union of Γ with an appropriate instanciation of denominator $\mathcal{D}_i$.

A branch in a tableau is a path from the root to an end node. A branch is *closed* if its end node contains $\bot$, otherwise it is *open.* A tableau is *closed* if all its branches are closed, otherwise it is *open.* We write $\varphi_0, \varphi'_1, \varphi_1, \ldots, \varphi'_n, \varphi_n \overset{1}{\vdash} \psi$ if there is a closed tableau for $(\varphi_0, \ldots, \varphi_{n-1}, \varphi_n \wedge \neg\psi, \varphi'_1, \ldots, \varphi'_n)$.

3.2 Tableau Rules

– Common rules for epistemic formulas and event formulas:

$$\frac{(\Sigma\ \varphi \wedge \psi)}{(\Sigma\ \varphi)\ (\Sigma\ \psi)}\ \wedge \qquad \frac{(\Sigma\ \neg(\varphi \wedge \psi))}{(\Sigma\ \neg\varphi) \mid (\Sigma\ \neg\psi)}\ \neg\wedge \qquad \frac{(\Sigma\ \neg\neg\varphi)}{(\Sigma\ \varphi)}\ \neg \qquad \frac{(\Sigma\ p)(\Sigma\ \neg p)}{\bot}\ \bot$$

$$\frac{(\Sigma\ \langle B_a\rangle\varphi)}{(R_a\ \Sigma\ \Sigma^+)(\Sigma^+\ \varphi)(\sigma_1^+\ p_1'^+)\dots(\sigma_i^+\ p_i'^+)}\ \langle B_a\rangle \qquad \frac{(\Sigma\ B_a\varphi)(R_a\ \Sigma\ \Sigma')}{(\Sigma'\ \varphi)\mid(\Sigma'\ 0)}\ B_a$$

where p is in $\Phi \cup \Phi'$, $\sigma_1^+ \ \dots\ \sigma_i^+$ are the σ_k^+ in the fresh generic label Σ^+ (none if $\Sigma^+ = \sigma^+$) and $p_k'^+$ are fresh new atomic events with $\mathrm{Pre}(p_i'^+) = p_i^+$.

– Specific rule for event formulas:

$$\frac{(\sigma_i\ p')(\sigma_i\ p'')}{\bot}\ \mathrm{Excl}$$

where p', p'' are not fresh ($p', p'' \in \Gamma^0$) and $p' \neq p''$.

– Specific rules for epistemic formulas:

$$\frac{(\sigma\ \sigma_1\dots\ \sigma_i\ p)}{(\sigma\ p)}\ \leftarrow_1 \qquad \frac{(\sigma\ \sigma_1\dots\ \sigma_i\ \neg p)}{(\sigma\ \neg p)}\ \leftarrow_2$$

$$\frac{(\sigma\ 0)}{\bot}\ \mathrm{Pre}_0 \qquad \frac{(\sigma\ \sigma_1\dots\ \sigma_i\ \varphi)(\sigma_i\ p')}{(\sigma\ \sigma_1\dots\ \sigma_{i-1}\ \mathrm{Pre}(p'))}\ \mathrm{Pre}_1 \quad \varphi \neq 0,\ i > 0$$

$$\frac{(\sigma\ \sigma_1\dots\sigma_i\ 0)(\sigma_i\ p')}{(\sigma\ \sigma_1\dots\sigma_{i-1}\ \neg\mathrm{Pre}(p'))\mid(\sigma\ \sigma_1\dots\sigma_{i-1}\ 0)}\ \mathrm{Pre}_2 \quad i > 0$$

We explain informally the meaning of these rules. The boolean rules, rule $\langle B_a\rangle$ and rule B_a are classic. Rule Excl enforces the Exclusivity of event models. It does not apply to the fresh atomic events that we add at the creation of each event label, because a "meaningful" atomic event may be added to such a label; however these fresh events are removed from the final constructed models unless they are the only atomic event in their possible event. Rules $\leftarrow_1$ and $\leftarrow_2$ reflect the fact that for a world $(w_0, w_1', \dots, w_i')$ in $\mathcal{M}_i$, by definition of the update product, $\mathcal{M}_i, (w_0, w_1', \dots, w_i') \models p$ if, and only if, $\mathcal{M}_0, w_0 \models p$. Rule Pre_1 says that if $(w_0, w_1', \dots, w_i')$ is in $\mathcal{M}_i$ and $\mathcal{M}_i', w_i' \models p'$, then $(w_0, w_1', \dots, w_{i-1}')$ is in $\mathcal{M}_{i-1}$ and $\mathcal{M}_{i-1}, (w_0, w_1', \dots, w_{i-1}')$ verifies $\mathrm{Pre}(p')$. Rule Pre_2 says that if $(w_0, w_1', \dots, w_i')$ is *not* in $\mathcal{M}_i$ and $\mathcal{M}_i', w_i' \models p'$, either $(w_0, w_1', \dots, w_{i-1}')$ is not in $\mathcal{M}_{i-1}$, or it is in $\mathcal{M}_{i-1}$ but $\mathcal{M}_{i-1}, (w_0, w_1', \dots, w_{i-1}') \not\models \mathrm{Pre}(p')$. Finally, Rule Pre_0 is used to forbid the rightmost choice in Rule Pre_2 when $i = 1$. Indeed it would make no sense to say that the world associated to a label σ must not be in the initial model $\mathcal{M}_0$ when it has been created to be in $\mathcal{M}_0$.

Proposition 3 (Tableau method soundness and completeness). *For all $\varphi_0, \dots, \varphi_n, \psi \in \mathcal{L}$, for all $\varphi_1', \dots, \varphi_n' \in \mathcal{L}'$, $\varphi_0, \varphi_1', \varphi_1, \dots, \varphi_n', \varphi_n \overset{1}{\vdash} \psi$ iff $\varphi_0, \varphi_1', \varphi_1, \dots, \varphi_n', \varphi_n \models_{\overline{n}}^{\underline{1}} \psi$.*

4 NEXPTIME-Completeness and Implementation

The NEXPTIME-completeness of our simple DEL-sequents [3] extends to generalized DEL-sequents.

Proposition 4. *The satisfiability problem is NEXPTIME-complete*

An implementation of the tableau method can be found at:

`http://www.irisa.fr/prive/fschwarz/lotrecscheme/`.

The tableau rules are written in LotrecScheme [16] which is a term rewriting system designed for implementing tableau methods. The corresponding rules can be found directly in the software by clicking 'open' and 'generalized DEL-sequents'.

The implemented rules are similar to those presented in Subsection 3.2. There are two main differences. The first one is that we tag worlds with *ok* and $\neg ok$ to say respectively that the node belongs to the model or not. We use those two tags for a reason of efficiency. The second difference concerns the pattern-matching of a condition of a rule: as LotrecScheme does not enable an arbitrary number of terms to match in a condition, we adapt the method so that the numbers of terms in all rules are fixed. For instance, let us consider the rule B_a. $(R_a\ \Sigma\ \Sigma')$ is a macro to denote an arbitrary number of terms whereas in the implementation $(R_a\ \Sigma\ \Sigma')$ is not a macro but a term and we have to simulate the way relations are defined in the product update:

$$\frac{(R_a\ \Sigma\ \Sigma')(R_a\ a\ b)(\Sigma :: a\ ok)}{(R_a\ \Sigma :: a\ \Sigma' :: b)} \text{ and } \frac{(R_a\ \Sigma :: a\ \Sigma' :: b)}{(R_a\ \Sigma\ \Sigma')(R_a\ a\ b)}$$

where a, b, Σ and Σ' are terms, and where $\Sigma :: a$ is the concatenation of Σ and a, and $\Sigma' :: b$ is the concatenation of Σ' and b. Σ and Σ' are sequences of labels, and a and b are labels representing worlds of an event model.

5 Extension to Other Semantics

In this section, we investigate the complexity of the satisfiability problem with semantics where the accessibility relations are also assumed to be reflexive or serial (an accessibility relation R_a is *reflexive* when for all $w \in W$, $w \in R_a(w)$, and *serial* when for all $w \in W$, there is $u \in W$ such that $u \in R_a(w)$).

$$\frac{(\Sigma\ \varphi)}{(R_a\ \Sigma\ \Sigma)}\ T \qquad\qquad \frac{(\Sigma\ B_a\varphi)}{(R_a\ \Sigma\ \Sigma^+)(\Sigma^+\ \varphi)}\ D$$

For reflexivity, we add the rule T above to the tableau method, where Σ can be σ or σ_i (reflexivity of product models stems from initial and event models being reflexive). Rule T is sound and complete with respect to reflexive models.

For seriality, we add the rule D above to the tableau method, where Σ can be σ, σ_i or $\sigma\ \sigma_1\ \ldots\ \sigma_i$. Rule D is sound and complete with respect to serial models. There is no problem of termination because we add a successor to a node if, and only if there is a modal formula in it.

Proposition 5. *The satisfiability problem when the relations are reflexive or serial is NEXPTIME-complete.*

6 Example: Coordinated Attack Problem

In this section, we assume that there are only two agents a and b. We extend our epistemic language $\mathcal{L}$ with the common knowledge operator $C_{a,b}\varphi$. The truth conditions of the common knowledge operator are defined as follows:

$$\mathcal{M}, w \models C_{a,b}\varphi \quad \text{iff for all } v \in (R_a \cup R_b)^+(w),\ \mathcal{M}, v \models \varphi,$$

where $(R_a \cup R_b)^+$ is the transitive closure of $(R_a \cup R_b)$.

Intuitively, $C_{a,b}\varphi$ is an abbreviation of an infinite conjunction (see [13] for more details): $C_{a,b}\varphi = E_{a,b}\varphi \wedge E^2_{a,b}\varphi \wedge E^3_{a,b}\varphi \wedge \ldots$, where $E^k_{a,b}\varphi$ is defined inductively as follows: $E^1_{a,b}\varphi = B_a\varphi \wedge B_b\varphi$ and $E^{k+1}_{a,b}\varphi = B_aE^k\varphi \wedge B_bE^k\varphi$. The definitions of DEL-sequents of Definition 1 can easily be adapted to this extended language with common knowledge.

The coordinated attack problem from the distributed systems folklore can be described informally as follows. Two generals need to attack their common enemy simultaneously if they want to win. However, their only way to communicate is by means of a messenger, and this messenger may be captured at any time between the two camps. If we assume that the messenger is really lucky and never gets caught on that particular night, how long will it take for the generals to coordinate an attack?

We can model this problem within our framework. Assume that general a has decided to attack at dawn. General a then sends a messenger to general b to inform him of his decision. The content of the first message sent by general a to general b is represented by the propositional letter *attack* standing for 'general a has decided to attack at dawn'. This message eventually reaches general b, but general a does not know it yet. This event is represented by an atomic event p'_1 standing for 'general b receives the decision of general a to attack at dawn'. Its precondition is $\text{Pre}(p'_1) = attack$. The only information we have about this event p'_1 is that general b knows about its occurrence: $B_bp'_1$. As a result of this event, general b now knows that general a has decided to attack: $B_b attack$. However, general a does not know it, so they still cannot coordinate a simultaneous attack. Therefore, general b sends an acknowledgement to general a. This message eventually reaches general a. This event is represented by the atomic event p'_2 standing for 'general a receives the first acknowledgement of general b'. The precondition of atomic event p'_2 is $\text{Pre}(p'_2) = B_b attack$. This time, general b does not know that his message has been delivered. Therefore, the only information we have about this event p'_2 is that general a knows about its occurrence: $B_ap'_2$. As a result of this event, general a now knows that general b knows that general a has decided to attack: $B_aB_b attack$. However, there is still no common knowledge that general a has decided to attack. This informal reasoning could go on indefinitely. It shows that common knowledge that general a has decided to attack cannot be attained.

The above informal reasoning can be formalized within our framework in a natural way. To achieve this aim, we define for all $k \in \mathbb{N}^*$ the atomic event p'_k whose precondition is $\text{Pre}(p'_k) = B_a B_b \dots B_a B_b attack$ if k is odd and $\text{Pre}(p'_k) = B_b B_a B_b \dots B_a B_b attack$ if k is even ($k-1$ knowledge operators are nested alternatively in $\text{Pre}(p'_k)$). The statement that after any finite number of messages exchanged, it is impossible to infer that there is common knowledge that general a has decided to attack is formalized by the fact that, for all $k \in \mathbb{N}^*$:

$$\begin{array}{ll} B_a attack, B_b p'_1, \top, B_a p'_2, \dots, B_a p'_{k-1}, \top, B_b p'_k, \top \not\models_{\frac{1}{k}} C_{a,b} attack & \text{if } k \text{ is odd} \\ B_a attack, B_b p'_1, \top, B_a p'_2, \dots, B_b p'_{k-1}, \top, B_a p'_k, \top \not\models_{\frac{1}{k}} C_{a,b} attack & \text{if } k \text{ is even} \end{array}$$

This result is itself due to the fact that for all $k \in \mathbb{N}^*$, $B_a attack, B_b p'_1, \top, B_a p'_2, \dots, B_b p'_k, \top \not\models_{\frac{1}{k}} E^{k+1}_{a,b} attack$ if k is odd, and $B_a attack, B_b p'_1, \top, B_a p'_2, \dots, B_a p'_k, \top \not\models_{\frac{1}{k}} E^{k+1}_{a,b} attack$ if k is even.

We illustrate our tableau method by proving this fact for $k = 1$ in Figure 1. Remark that $E^2_{a,b} attack = B_a(B_a attack \wedge B_b attack) \wedge B_b(B_a attack \wedge B_b attack)$. Also, we only show one branch of the full tableau, the one which is open.

$$\begin{tikzcd}
\Gamma_0 = \{(\sigma\ B_a attack), (\sigma_1\ B_b p'_1), (\sigma_1\ p'^{+}_1), (\sigma\ \sigma_1\ \neg E^2_{a,b})\} \arrow[d, "\text{Pre}_1 \text{ and } \neg\wedge"] \\
\Gamma_1 = \Gamma_0 \cup \{(\sigma\ p^{+}_1)\} \cup \{(\sigma\ \sigma_1\ \langle B_a\rangle \neg(B_a attack \wedge B_b attack))\} \arrow[d, "\langle B_a\rangle"] \\
\Gamma_2 = \Gamma_1 \cup \{(R_a\ \sigma\ \sigma'), (R^1_a\ \sigma_1\ \sigma'_1), (\sigma'\ \sigma'_1\ \neg(B_a attack \wedge B_b attack)), (\sigma'_1\ p'^{+}_2)\} \arrow[d, "\text{Pre}_1,\ B_a \text{ and } \neg\wedge"] \\
\Gamma_3 = \Gamma_2 \cup \{(\sigma'\ p^{+}_2)\} \cup \{(\sigma'\ attack)\} \cup \{(\sigma'\ \sigma'_1\ \langle B_b\rangle \neg attack)\} \arrow[d, "\langle B_b\rangle"] \\
\Gamma_4 = \Gamma_3 \cup \{(R_b\ \sigma'\ \sigma''), (R^1_b\ \sigma'_1\ \sigma''_1), (\sigma''\ \sigma''_1\ \neg attack)), (\sigma''_1\ p'^{+}_3)\} \arrow[d, "\text{Pre}_1 \text{ and } \leftarrow_1"] \\
\Gamma_5 = \Gamma_4 \cup \{(\sigma''\ p^{+}_3)\} \cup \{(\sigma\ \neg attack)\}
\end{tikzcd}$$

Fig. 1. Open branch of the tableau proving that $B_a attack, B_b p'_1, \top \not\models_{\frac{1}{1}} E^2_{a,b} attack$

7 Conclusion

7.1 Related Work

In [12], Dupin de Saint-Cyr and Lang define an operator of "extrapolation". This operator takes as input a temporal formula, which corresponds to a sequence of observations under the form of propositional formulas indexed by time stamps, and yields as output another temporal formula, which corresponds semantically to the *preferred* sequences of states which satisfy the input temporal formula. The authors follow an internal approach. In [9], Booth and Nittka address a similar problem but with an imperfect external approach (see [1] for more details on the different modeling approaches). They are interested in inferring what the agent believed (or will believe) at a given moment, based on a sequence of observations consisting of responses that the agent has given to a sequence of belief revision inputs. Both papers deal with situations involving a single agent. Our approach is different from both approaches, because we do not strive to "extrapolate" new information from existing observations by resorting to an argument of minimal change. Instead, we are only interested in inferring some *necessary* property that follows from these existing observations.

Some tableau methods have been proposed for DEL, but only for public announcement logic [4,11] and hybrid public anouncement logic [14]. A terminating tableau method has also been proposed for the full BMS framework in [14] by encoding the reduction axioms as tableau rules. However, none of these tableau methods can somehow address the two problems raised in the introduction, because the BMS language of [6] does not allow for partial and incomplete descriptions of events: an event model or a formula announced publicly specifies *completely* how all the agents perceive the occurrence of the corresponding event. In particular, it is impossible to model the coordinated attack problem.

In [5], a sequent calculus has been developed, yet in an algebraic setting, making a systematic comparison difficult. Their sequents $m_1, \dots, q_1, \dots, A_1, \dots, m_k, \dots, q_l, \dots, A_n \vdash \delta$ are arbitrarily long and consist of different types of formulas which can contain propositions $m_1, \dots, m_k$, events $q_1, \dots, q_l$ and agents $A_1, \dots, A_n$, and which resolve into a single proposition *or* event δ. A sequent calculus has also been proposed for public announcement logic [15] which, alike our tableau terms, refers explicitely to possible worlds and relations.

7.2 Concluding Remarks

As we said in the previous subsection, our tableau method can infer necessary information which was somehow already encoded in the sequence. In fact, our method allows us to *verify* in a *complexity-optimal* way that a piece of information about a sequence of events does follow from this sequence. It also allows us to check that a given epistemic plan (viewed as a sequence of event properties) yields an epistemic goal. Even if it does not provide an algorithm to synthesize it ([2] deals more with synthesis), it can still be instrumental in finding out some of its necessary properties. It can also be used during the design of the epistemic plan itself to check that we are designing it in the 'right' direction.

As the title of this paper suggests, we have generalized our previous work [3] to arbitrary long sequences of events, and extended it to the case of reflexive or serial models.[4] Also we illustrate our method by encoding the coordinated

[4] A tableau rule is missing in [3]. This error is corrected in the paper available on the HAL archive at `http://hal.inria.fr/docs/00/64/64/81/PDF/M4M11.pdf`

attack problem. However, other generalizations still need to be done, such as the addition of a common knowledge operator in the language (as illustrated in Section 6) and the integration of ontic events.

References

1. Aucher, G.: An internal version of epistemic logic. Studia Logica 94(1), 1–22 (2010)
2. Aucher, G.: DEL-sequents for regression and epistemic planning. Journal of Applied Non-Classical Logics (to appear, 2012)
3. Aucher, G., Maubert, B., Schwarzentruber, F.: Tableau method and NEXPTIME-completeness of DEL-sequents. Electronic Notes in Theoretical Computer Science 278, 17–30 (2011)
4. Balbiani, P., van Ditmarsch, H., Herzig, A., de Lima, T.: Tableaux for public announcement logic. Journal of Logic and Computation 20(1), 55–76 (2010)
5. Baltag, A., Coecke, B., Sadrzadeh, M.: Algebra and sequent calculus for epistemic actions. In: Proceedings of Workshop on Logic and Communication in Multi-Agent Systems (LCMAS 2004), pp. 60–78 (2004)
6. Baltag, A., Moss, L.: Logic for epistemic programs. Synthese 139(2), 165–224 (2004)
7. Baltag, A., Moss, L., Solecki, S.: The logic of common knowledge, public announcement, and private suspicions. In: Gilboa, I. (ed.) Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 1998), pp. 43–56 (1998)
8. Baltag, A., Moss, L., Solecki, S.: The logic of public announcements, common knowledge and private suspicions. Technical report, Indiana University (1999)
9. Booth, R., Nittka, A.: Reconstructing an agent's epistemic state from observations about its beliefs and non-beliefs. J. Log. Comput. 18(5), 755–782 (2008)
10. Brusoni, V., Console, L., Terenziani, P., Dupré, D.T.: A spectrum of definitions for temporal model-based diagnosis. Artificial Intelligence 102(1), 39–79 (1998)
11. de Boer, M.: KE tableaux for public anouncement logic. In: Proceedings of Formal Approaches to Multi-Agent Systems Workshop (FAMAS 2007), Durham, UK (2007)
12. de Saint-Cyr, F.D., Lang, J.: Belief extrapolation (or how to reason about observations and unpredicted change). Artif. Intell. 175(2), 760–790 (2011)
13. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about knowledge. MIT Press (1995)
14. Hansen, J.U.: Terminating tableaux for dynamic epistemic logic. Electronic Notes in Theoretical Computer Science 262, 141–156 (2010)
15. Maffezioli, P., Negri, S.: A gentzen-style analysis of public announcement logic. In: Proceedings of the International Workshop on Logic and Philosophy of Knowledge, Communication and Action, pp. 293–313 (2010)
16. Schwarzentruber, F.: Lotrecscheme. Electronic Notes in Theoretical Computer Science 278, 187–199 (2011)

Deciding the Bisimilarity Relation between Datalog Goals

Philippe Balbiani and Antoun Yaacoub

Institut de Recherche en Informatique de Toulouse, CNRS — Université de Toulouse
118 route de Narbonne, 31062 Toulouse Cedex 9, France
{balbiani,yaacoub}@irit.fr

Abstract. We introduce the concept of bisimulation between Datalog goals: two Datalog goals are bisimilar with respect to a given Datalog program when their SLD-trees, considered as relational structures, are bisimilar. We address the problem of deciding whether two given goals are bisimilar with respect to given programs. When the given programs are hierarchical or restricted, this problem is decidable in 2EXPTIME.

Keywords: Logic programming, Datalog, Equivalence of goals, Bisimulation, Decision method, Computational complexity.

1 Introduction

Within the context of a given programming language, it is essential to associate a semantics to programs written in it. This semantics induces an equivalence relation between programs. Hence, the decision problem "given two programs, determine whether they are equivalent" is at the heart of the study of program semantics.

In logic programming, several ideas of equivalence of logic programs are in competition. These ideas are based on the various semantics of logic programs. For example, if one considers the least Herbrand model semantics, then two given logic programs are said to be equivalent iff their least Herbrand models are equal. Such equivalence relations between programs have been intensively studied in the past [6,7,11,12]. Nevertheless, in some cases, they fail to adequately provide the right tool for comparing logic programs. For instance, in the case of perpetual processes, most of them are inadequate to offer frameworks in which to reason about programs. The truth is that two perpetual processes may have the same declarative semantics (defined in terms of least model) for example and, still have very different behaviors. And the various equivalence relations considered in [4,5,8,10] do not take into account the computational aspects of logic programming. The goal of this paper is to suggest the use of equivalence relations between logic programs that take into account the shape of the SLD-trees that these programs give rise to. This idea is not new: in automata theory, for instance, many variants of the equivalence relation of bisimilarity have been defined in order to promote the idea that automata with the same trace-based semantics should sometimes not be considered as equivalent if they are not bisimilar [13].

In this paper, in order to simplify matter, we consider logic programs of a very simple kind: Datalog programs. Furthermore, comparing two given Datalog programs and taking into account the shape of the SLD-trees they give rise to necessitates the comparison of infinitely many SLD-trees. Thus, in a first approach, we restrict our study to the comparison of two given Datalog goals. We will say that, with respect to a fixed Datalog program P, two given goals are equivalent when their SLD-trees are bisimilar.

In this paper, we investigate the computability and complexity of the equivalence problem between Datalog goals. In particular, we examine the complexity of the following decision problems:

- given two Datalog goals F, G and a hierarchical Datalog program P, determine if the SLD-trees of $P \cup F$ and $P \cup G$ are bisimilar.
- given two Datalog goals F, G and a restricted Datalog program P, determine if the SLD-trees of $P \cup F$ and $P \cup G$ are bisimilar.

In section 2 of this paper, we will present some basic notions about Datalog programs, syntax and semantics. In section 3, we will introduce the concept of bisimulation between Datalog goals. An undecidability result concerning Prolog programs will be given on section 4. In section 5, we will address the problem of deciding whether two given goals are bisimilar with respect to a given hierarchical program. At the end of the section, computational issues will be studied. Note that for hierarchical programs, the SLD-tree for a goal $\leftarrow G$ contains only finite branches. In section 6, we will address the same questions as in section 5 by considering here restricted programs. These programs allow a specific kind of recursion in the clauses. Finally, we will conclude by proposing an open question concerning to decide whether two given goals are bisimilar with respect to a *nonvariable introducing* logic program or a *single variable occurrence* logic program.

2 Datalog Programs

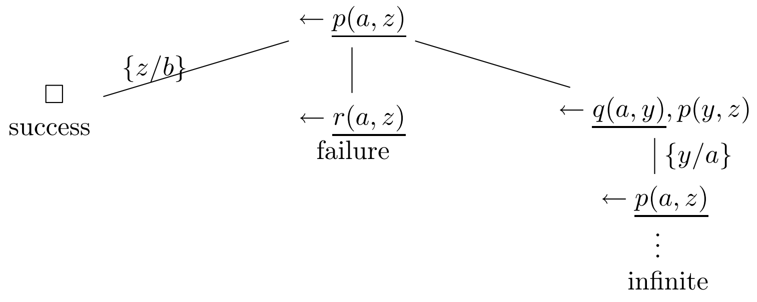
In many respects, Datalog is a simplified version of Prolog. A Datalog program consists of a finite set of Horn clauses of the form $A_0 \leftarrow A_1, \cdots, A_n$ where each A_i is a literal of the form $p(t_1, \cdots, t_k)$ such that p is a predicate symbol of arity k and the $t_1, \cdots, t_k$ are terms. A term is either a constant or a variable. The left-hand side of a Datalog-clause is called its head and the right-hand side is called its body. Any Datalog program must satisfy the following condition: each variable which occurs in the head of a clause must also occur in the body of the same clause. A Datalog program P is said to be hierarchical iff there exists a mapping l associating a nonnegative integer $l(p)$ to every predicate symbol p occurring in P and such that for all clauses $A_0 \leftarrow A_1, \cdots, A_n$ in P, if each A_i is a literal of the form $p_i(t_1, \cdots, t_{k_i})$ then $l(p_0) > l(p_1), \cdots, l(p_n)$. The dependency graph of a Datalog program P is the graph (N, E) where N is the set of all predicate symbols occurring in P and E is the adjacency relation defined on N as follows: pEq iff P contains a clause $A_0 \leftarrow A_1, \cdots, A_n$ such that A_0 is a literal of the form $p(\cdots)$ and for some $1 \leq i \leq n$, A_i is a literal of the form $q(\cdots)$. Let E^* be the reflexive transitive closure of E. A Datalog program P is said to be restricted iff for all clauses $A_0 \leftarrow A_1, \cdots, A_n$ in P and for all $1 \leq i \leq n-1$, if A_0 is of the form $p(\cdots)$ and A_i is of the form $q(\cdots)$, then not qE^*p. A program P is *nonvariable introducing* (in short *nvi*) if for every clause $A \leftarrow B_1, \cdots, B_n$ in P, every variable that occurs in $B_1, \cdots, B_n$ occurs also in A. A program P is *single variable occurrence* (in short *svo*) if for every clause $A \leftarrow B_1, \cdots, B_n$ in P, no variable in $B_1, \cdots, B_n$ occurs more than once. A Datalog goal is a Horn clause of the form $G = \leftarrow B_1, \cdots, B_m$. Its size, denoted $|G|$, is m. There exists a goal of size 0, the empty goal (denoted $\Box$). The proof theoretic analysis of formalisms such as Datalog is oriented towards the following decision problem: given a Datalog program P and a Datalog goal $\leftarrow B_1, \cdots, B_m$, determine whether there exists a ground substitution θ such that for every $1 \leq i \leq m$, $B_i\theta$ can be inferred from P. The inference rule we will rely on in this paper is the so-called SLD-resolution principle and the computation rule is the one which always selects the leftmost atom in a goal. Consider a Datalog clause of the form $A_0 \leftarrow A_1, \cdots, A_n$ and a Datalog goal $\leftarrow B_1, \cdots, B_m$. If a substitution θ exists such that, $A_0\theta = B_1\theta$ then the Datalog goal $\leftarrow A_1\theta, \cdots, A_n\theta, B_2\theta, \cdots, B_m\theta$ is called a resolvent of $A_0 \leftarrow A_1, \cdots, A_n$ and $\leftarrow B_1, \cdots, B_m$. Let P be a Datalog program and G be a Datalog goal. An SLD-derivation of $P \cup \{G\}$ consists of (finite or infinite) sequences $G_0, G_1, \cdots$ of goals, $C_1, C_2, \cdots$ of P's clauses and $\theta_1, \theta_2, \cdots$ of most general unifiers such that $G_0 = G$ and for all $i \geq 1$, G_i is a resolvent of C_i and G_{i-1} using θ_i. An SLD-refutation of $P \cup \{G\}$ is a finite SLD-derivation of $P \cup \{G\}$ which has the empty goal $\Box$ as the last goal in the derivation. An SLD-tree for $P \cup \{G\}$ is a labeled tree satisfying the following conditions:

– The root node is labeled with G.
– Let $\leftarrow B_1, \cdots, B_m$ $(m \geq 1)$ be the label of a node in the tree. Then for every clause $A_0 \leftarrow A_1, \cdots, A_n$ in P and for every most general unifier θ of B_1 and A_0, the resolvent $\leftarrow A_1\theta, \cdots, A_n\theta, B_2\theta, \cdots, B_m\theta$ of $\leftarrow B_1, \cdots, B_m$ and $A_0 \leftarrow A_1, \cdots, A_n$ via θ is the label of a child of $\leftarrow B_1, \cdots, B_m$.

Each branch of the SLD-tree corresponds to an SLD-derivation in a natural way.

Example 1. Let P be the program:
$r(b, b) \leftarrow,$
$q(a, a) \leftarrow,$
$p(a, b) \leftarrow,$
$p(x, y) \leftarrow r(x, y),$
$p(x, z) \leftarrow q(x, y), p(y, z)$

and G be the goal $\leftarrow p(a, z)$. Remark that P is restricted. The next figure shows the SLD-tree for this goal. The selected atoms are underlined and the success, failure and infinite branches are shown.



$\leftarrow \underline{p(a,z)}$
$\{z/b\}$
$\square$
success
$\leftarrow \underline{r(a,z)}$
failure
$\leftarrow \underline{q(a,y)}, p(y,z)$
$\{y/a\}$
$\leftarrow \underline{p(a,z)}$
infinite

Note that the above SLD-tree is infinite and has success branches corresponding to the substitution answer $\{z/b\}$.

3 Bisimulation

In this section, we introduce the central concept of the paper: bisimulation. Quite simply, a bisimulation is a binary relation between goals such that related goals have "equivalent" SLD-trees. Let P be a Datalog program. A binary relation $\mathcal{Z}$ between Datalog goals is said to be a P-bisimulation iff it satisfies the following conditions for all Datalog goals F_1, G_1 such that $F_1 \mathcal{Z} G_1$:

- $F_1 = \square$ iff $G_1 = \square$,
- For each resolvent F_2 of F_1 and a clause in P, there exists a resolvent G_2 of G_1 and a clause in P such that $F_2 \mathcal{Z} G_2$,
- For each resolvent G_2 of G_1 and a clause in P, there exists a resolvent F_2 of F_1 and a clause in P such that $F_2 \mathcal{Z} G_2$.

It is a rather easy observation that the set of all P-bisimulations is closed under taking arbitrary unions. This shows that:

Proposition 1. *There exists a maximal P-bisimulation, namely the binary relation $\mathcal{Z}^P_{max}$ between Datalog goals defined as follows: $F_1 \mathcal{Z}^P_{max} G_1$ iff there exists a P-bisimulation $\mathcal{Z}$ such that $F_1 \mathcal{Z} G_1$.*

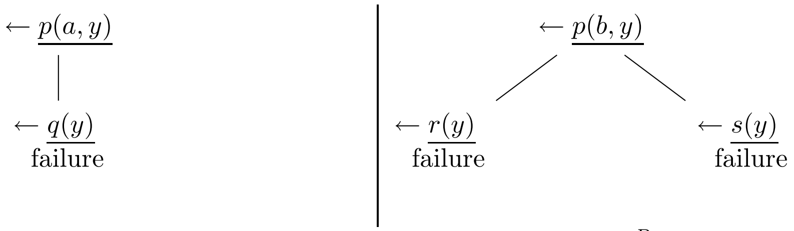
Obviously, the identity relation between Datalog goals is a P-bisimulation. Moreover, for all P-bisimulations $\mathcal{Z}$, $\mathcal{Z}^{-1}$ (the inverse of $\mathcal{Z}$) is a P-bisimulation. Finally, for all P-bisimulations $\mathcal{Z}, \mathcal{T}$, $\mathcal{Z} \circ \mathcal{T}$ (the composition of $\mathcal{Z}$ and $\mathcal{T}$) is a P-bisimulation. It follows immediately that:

Proposition 2. *$\mathcal{Z}^P_{max}$ is an equivalence relation on the set of all Datalog goals.*

Example 2. Let P be the following program:
$p(a,y) \leftarrow q(y)$,
$p(b,y) \leftarrow r(y)$,
$p(b,y) \leftarrow s(y)$
and let F, G be respectively the following goals $\leftarrow p(a,y)$ and $\leftarrow p(b,y)$. Note that P is hierarchical and restricted.

Let Z be the binary relation between goals such that:

$\leftarrow p(a,y)\ \mathcal{Z} \leftarrow p(b,y)$,
$\leftarrow q(y)\ \mathcal{Z} \leftarrow r(y)$,
$\leftarrow q(y)\ \mathcal{Z} \leftarrow s(y)$.



$\leftarrow \underline{p(a,y)}$
$\leftarrow \underline{q(y)}$
failure
$\leftarrow \underline{p(b,y)}$
$\leftarrow \underline{r(y)}$
failure
$\leftarrow \underline{s(y)}$
failure

Obviously, $\mathcal{Z}$ is a *P-bisimulation.* Since $F\ \mathcal{Z}\ G$, then $F\ \mathcal{Z}^P_{max}\ G$.

In this paper, we address the following decision problem : (π) given a Datalog program P and Datalog goals F_1, G_1, determine whether $F_1 \mathcal{Z}^P_{max} G_1$.

4 Undecidability of Bisimulation for Prolog Programs

As is well known, the SLD-resolution principle and the concepts of resolvent, SLD-derivation, SLD-refutation and SLD-tree have also been considered within the context of Prolog programs and Prolog goals. In addition to the predicate symbols, constants and variables composing the alphabet of Datalog, the alphabet of Prolog also includes function symbols allowing the use of terms inductively defined as follows: *(i)* constants are terms; *(ii)* variables are terms; *(iii)* expressions of the form $f(t_1, \cdots, t_h)$ where f is a function symbol of arity h and $t_1, \cdots, t_h$ are terms are terms. As a result, one may also define in Prolog the binary relations between goals similar to the bisimulation defined in Datalog. Nevertheless,

Proposition 3. *It is undecidable, given a Prolog program P and Prolog goals F_1, G_1, to determine whether $F_1\ \mathcal{Z}^P_{max}\ G_1$.*

Proof. Let us consider the following decision problem: (π_1) given a Prolog program P with exactly one binary clause (i.e. a clause such that its body contains at most one atom) and a Prolog goal $\leftarrow B$, determine whether the SLD-tree for $P \cup \{\leftarrow B\}$ contains an infinite branch. Let $(P, \leftarrow B)$ be an instance of π_1. We consider new constants a, b, c and a new predicate symbol p of arity 2. Let P' be the least Prolog program containing P and the following Horn clauses:

$p(a,y) \leftarrow B$,
$p(b,y) \leftarrow B$,
$p(b,y) \leftarrow p(c,y)$,
$p(c,y) \leftarrow p(c,y)$.

We demonstrate that the following conditions are equivalent:

(i) the SLD-tree for $P \cup \{\leftarrow B\}$ contains an infinite branch;
(ii) $\leftarrow p(a,y) \mathcal{Z}_{max}^{P'} \leftarrow p(b,y)$.

$(i) \Rightarrow (ii)$ Suppose the SLD-tree for $P \cup \{\leftarrow B\}$ contains an infinite branch. Since P consists of exactly one binary clause, then the SLD-tree for $P \cup \{\leftarrow B\}$ is equal to a unique infinite branch. Obviously, the SLD-tree for $P' \cup \{\leftarrow p(a,y)\}$ is also equal to a unique infinite branch whereas the SLD-tree for $P' \cup \{\leftarrow p(b,y)\}$ is equal to a pair of infinite branches. Hence, $\leftarrow p(a,y) \mathcal{Z}_{max}^{P'} \leftarrow p(b,y)$.

$(ii) \Rightarrow (i)$ Suppose $\leftarrow p(a,y) \mathcal{Z}_{max}^{P'} \leftarrow p(b,y)$. Obviously, the SLD-tree for $P' \cup \{\leftarrow p(b,y)\}$ contains an infinite branch. Hence, the SLD-tree for $P' \cup \{\leftarrow p(a,y)\}$ also contains an infinite branch. Hence, the SLD-tree for $P \cup \{\leftarrow B\}$ contains an infinite branch.

Since (π_1) is undecidable [3], then it is undecidable, given a Prolog program P and Prolog goals F_1, G_1, to determine whether $F_1 \ \mathcal{Z}_{max}^{P} \ G_1$. □

A question arises here, how can we restore this decision problem to decidability? One can think (as we have done) about considering specific classes of logic programs, by restricting the language of logic programming. In the setting of Datalog programs for example, the main difficulty concerning (π) comes from loops or infinite branches in SLD trees. We will address this issue in the following sections.

5 Decidability of Bisimulation for Hierarchical Programs

We now study the computational complexity of the following decision problem: (π_{hie}) given an hierarchical Datalog program P and Datalog goals F_1, G_1, determine whether $F_1 \mathcal{Z}_{max}^{P} G_1$. In this respect, let P be a hierarchical Datalog program. Remark that the Datalog program considered in Example 1 is not hierarchical whereas the Datalog program considered in Example 2 is hierarchical.

In `Algorithm 1`, `bothempty(`F_1`,`G_1`)` is a Boolean function returning `true` iff $F_1 = \square$ and $G_1 = \square$, whereas `bothfail(`F_1`,`G_1`)` is a Boolean function returning `true` iff $F_1 \neq \square$, `successor(`F_1`)=`$\emptyset$, $G_1 \neq \square$ and `successor(`G_1`)=`$\emptyset$. Moreover, `successor(.)` is a function returning the set of all resolvents of its argument with a clause of P whereas `get-element(.)` is a function removing one element from the set of elements given as input and returning it. In order to demonstrate the decidability of (π_{hie}), we need to prove the following lemmas for all Datalog goals F_1, G_1:

Lemma 1 (Termination). `bisim1(`F_1`,`G_1`)` *terminates.*

Lemma 2 (Completeness). *If* $F_1 \mathcal{Z}_{max}^{P} G_1$*, then* `bisim1(`$F_1$`,`$G_1$`)` *returns* `true`*.*

Lemma 3 (Soundness). *If* `bisim1(`F_1`,`G_1`)` *returns* `true`*, then* $F_1 \mathcal{Z}_{max}^{P} G_1$*.*

Let $\ll$ be the binary relation on the set of all pairs of Datalog goals defined by: $(F_2, G_2) \ll (F_1, G_1)$ iff

– the SLD-tree for F_1 is deeper than the SLD-tree for F_2,
– the SLD-tree for G_1 is deeper than the SLD-tree for G_2.

The depth of an SLD-tree is the depth of its longest branch. Remark that in this section, all SLD-trees are finite.

Obviously, $\ll$ is a partial order on the set of all pairs of goals. Since P is hierarchical, then $\ll$ is well-founded.

Algorithm 1. function bisim1(F_1,G_1)

```
begin
    if $bothempty(F_1, G_1)$ or $bothfail(F_1, G_1)$ then
        return $true$
    else
        $SF \longleftarrow successor(F_1)$
        $SG \longleftarrow successor(G_1)$
        if $SF \neq \emptyset$ and $SG \neq \emptyset$ then
            $SF' \longleftarrow SF$
            while $SF' \neq \emptyset$ do
                $F_2 \longleftarrow get\text{-}element(SF')$
                $found\text{-}bisim \longleftarrow false$
                $SG' \longleftarrow SG$
                while $SG' \neq \emptyset$ and $found\text{-}bisim = false$ do
                    $G_2 \longleftarrow get\text{-}element(SG')$
                    $found\text{-}bisim \longleftarrow bisim1(F_2, G_2)$
                if $found\text{-}bisim = false$ then
                    return $false$
            $SG' \longleftarrow SG$
            while $SG' \neq \emptyset$ do
                $G_2 \longleftarrow get\text{-}element(SG')$
                $found\text{-}bisim \longleftarrow false$
                $SF' \longleftarrow SF$
                while $SF' \neq \emptyset$ and $found\text{-}bisim = false$ do
                    $F_2 \longleftarrow get\text{-}element(SF')$
                    $found\text{-}bisim \longleftarrow bisim1(G_2, F_2)$
                if $found\text{-}bisim = false$ then
                    return $false$
            return $true$
        else
            return $false$
end
```

Proof of Lemma 1. The proof is done by $\ll$-induction on (F_1, G_1). Let (F_1, G_1) be such that for all (F_2, G_2), if $(F_2, G_2) \ll (F_1, G_1)$ then `bisim1(`F_2`,`G_2`)` terminates. Since every recursive call to `bisim1` that is performed along the execution of `bisim1(`F_1`,`G_1`)` is done with respect to a pair (F_2, G_2) of goals such that $(F_2, G_2) \ll (F_1, G_1)$, then `bisim1(`$F_1$`,`$G_1$`)` terminates. □

Proof of Lemma 2. Let us consider the following property: ($Prop_1(F_1, G_1)$) if $F_1 \mathcal{Z}^P_{max} G_1$ then `bisim1(`F_1`,`G_1`)` returns `true`. Again, we proceed by $\ll$-induction. Suppose (F_1, G_1) is such that for all (F_2, G_2), if $(F_2, G_2) \ll (F_1, G_1)$ then $Prop_1(F_2, G_2)$. Let us show that $Prop_1(F_1, G_1)$. Suppose $F_1 \mathcal{Z}^P_{max} G_1$. Hence, for all successors F_2 of F_1, there exists a successor G_2 of G_1 such that $F_2 \mathcal{Z}^P_{max} G_2$, and conversely. Seeing that the logic program is hierarchical, then $(F_2, G_2) \ll (F_1, G_1)$. By induction hypothesis, $Prop_1(F_2, G_2)$. Since $F_2 \mathcal{Z}^P_{max} G_2$, then `bisim1(`$F_2$`,`$G_2$`)` returns `true`. As a result, one sees that `bisim1(`F_1`,`G_1`)` returns `true`. □

Proof of Lemma 3. It suffices to demonstrate that the binary relation $\mathcal{Z}$ defined as follows between Datalog goals is a bisimulation: $F_1 \mathcal{Z} G_1$ iff `bisim1(`F_1`,`G_1`)` returns `true`. Let F_1, G_1 be Datalog goals such that $F_1 \mathcal{Z} G_1$. Hence, `bisim1(`F_1`,`G_1`)` returns `true`. Thus, obviously, $F_1 = \Box$ iff $G_1 = \Box$, and the first condition characterizing bisimulations holds for $\mathcal{Z}$. Now, suppose that F_2 is a resolvent of F_1 and a clause in P. Since `bisim1(`F_1`,`G_1`)` returns `true`, then there exists a resolvent G_2 of G_1 and a clause in P such that `bisim1(`F_2`,`G_2`)` returns `true`, i.e. $F_2 \mathcal{Z} G_2$. As a result, the second condition characterizing bisimulations holds for $\mathcal{Z}$. The third condition characterizing bisimulations holds for $\mathcal{Z}$ too, as the reader can quickly check. Thus $\mathcal{Z}$ is a bisimulation. □

As a consequence of lemmas 1 – 3, we have:

Theorem 1. `Algorithm 1` *is a sound and complete decision procedure for* (π_{hie}).

It follows that (π_{hie}) is decidable. Moreover,

Theorem 2. (π_{hie}) *is in 2EXPTIME.*

Proof. Let P be a hierarchical Datalog program and G be a goal. Let n be the maximal number of atoms in the clauses of P or in G, and t be the number of level mapping in P. In fact, the maximal depth of a branch in an SLD-tree is equal to $n\times$(the maximal depth of a branch at level $t-1$) which is in turn equal to $n \times (1 + n \times (1+$ the maximal depth of a branch at level $t-2$)). Thus, by iterating the same operation until level 1, we conclude that the maximal depth D of a branch in an SLD-tree cannot exceed $\sum_{i=1}^{t} n^i$. Remark that $\sum_{i=1}^{t} n^i \leq t \times n^t$. Let s be the maximal number of clauses that defines a predicate. Hence, the branching degree of the SLD-tree for $P \cup \{G\}$ is bounded by s. Remark that our algorithm uses twice two nested loops. In fact, for a maximal depth D, the time complexity of the algorithm is approximately equal to $2 \times s^2 \times$ (the time complexity of the algorithm for a depth $D-1$) which is in turn equal to $4 \times s^4 \times$ (the time complexity of the algorithm for a depth $D-2$). Thus, by iterating the same operation until depth 1, one can show that for a depth D, the time complexity of our algorithm is about $2^D \times s^{2\times D} \leq 2^{t\times n^t} \times s^{2\times t\times n^t}$. Note that for a hierarchical program P and a goal G, the number of nodes in the corresponding SLD-tree is about $s^{t\times n^t}$. □

Concerning the exact complexity of (π_{hie}), we do not know whether (π_{hie}) is 2EXPTIME-hard or (π_{hie}) is in EXPSPACE.

6 Decidability of Bisimulation for Restricted Programs

We now study the computational complexity of the following decision problem: (π_{res}) given a restricted Datalog program P and Datalog goals F_1, G_1, determine whether $F_1 \mathcal{Z}_{max}^P G_1$. In this respect, let P be a restricted Datalog program. In `Algorithm 2`, `bothempty(`F_i`,`G_i`)` and `bothfail(`F_i`,`G_i`)` are similar to the corresponding functions used in `Algorithm 1`, whereas `occur((`$F_1 \Rightarrow \cdots \Rightarrow F_i$`),(`$G_1 \Rightarrow \cdots \Rightarrow G_i$`))` is a Boolean function returning `true` iff there exists substitutions σ, τ such that $F_l = F_k\sigma$, $G_l = G_k\tau$ for some $1 \leq k < l \leq i$. As the reader can see, `Algorithm 2` is very similar to `Algorithm 1`. The main difference lies in the introduction of the `occur` test in the first conditional instruction. Remark that the Datalog programs considered in Example 1 and Example 2 are restricted.

In order to demonstrate the decidability of (π_{res}), we need to prove the following lemmas for all Datalog goals F_1, G_1.

Lemma 4 (Termination). `bisim2((`F_1`),(`G_1`))` *terminates.*

Lemma 5 (Completeness). *If* $F_1 \mathcal{Z}_{max}^P G_1$*, then* `bisim2((`$F_1$`),(`$G_1$`))` *returns* `true`*.*

Lemma 6 (Soundness). *If* `bisim2((`F_1`),(`G_1`))` *returns* `true`*, then* $F_1 \mathcal{Z}_{max}^P G_1$*.*

Let $\prec$ be the binary relation on the set of all pairs of SLD-derivations defined by: $((F_1 \Rightarrow \cdots \Rightarrow F_i), (G_1 \Rightarrow \cdots \Rightarrow G_i)) \prec ((F'_1 \Rightarrow \cdots \Rightarrow F'_j), (G'_1 \Rightarrow \cdots \Rightarrow G'_j))$ iff $i > j$, $(F_1 \Rightarrow \cdots \Rightarrow F_j) = (F'_1 \Rightarrow \cdots \Rightarrow F'_j)$, $(G_1 \Rightarrow \cdots \Rightarrow G_j) = (G'_1 \Rightarrow \cdots \Rightarrow G'_j)$, there exists no substitutions σ, τ such that $F'_l = F'_k\sigma$, $G'_l = G'_k\tau$ for some $1 \leq k < l \leq i$. Obviously, $\prec$ is a partial order on the set of all pairs of SLD-derivations. Let us demonstrate that $\prec$ is well-founded. Suppose $\prec$ is not well-founded. Hence, there exists infinite SLD-derivations $(F_1 \Rightarrow F_2 \Rightarrow \cdots)$, $(G_1 \Rightarrow G_2 \Rightarrow \cdots)$ such that for all substitutions σ, τ, $F_l \neq F_k\sigma$ or $G_l \neq G_k\tau$ for all $1 \leq k < l$. This contradicts the following claim.

Claim. Let $(F_1 \Rightarrow F_2 \Rightarrow \cdots)$, $(G_1 \Rightarrow G_2 \Rightarrow \cdots)$ be infinite SLD-derivations. There exists $M \geq 1$ and there exists substitutions σ, τ such that $F_M = F_k\sigma$, $G_M = G_k\tau$ for some $1 \leq k < M$.

Proof. According to [2, Corollary 4.14], there exists $N \geq 1$ such that for all $j \geq 1$, $|F_j| \leq N$ and $|G_j| \leq N$. Let $\cong$ be the binary relation on the set of all pairs of goals of size bounded by N defined by: $(F, G) \cong (F', G')$ iff

- there exists substitutions σ, τ such that $F\sigma = F'$, $G\tau = G'$,
- there exists substitutions σ', τ' such that $F'\sigma' = F$, $G'\tau' = G$.

Obviously, $\cong$ is an equivalence relation on the set of all pairs of goals of size bounded by N. Seeing that our Datalog language has no function symbols and possesses finitely many predicate symbols and constants, $\cong$ determines a finite number of equivalence relations. Thus, one of these equivalence classes is repeated infinitely often in the sequence $(F_1, G_1), (F_2, G_2), \cdots$. Therefore, there exists $M \geq 1$ and there exists substitutions σ, τ such that $F_M = F_k\sigma$, $G_M = G_k\tau$ for some $1 \leq k < M$. $\square$

Algorithm 2. function bisim2($(F_1 \Rightarrow \cdots \Rightarrow F_i)$,$(G_1 \Rightarrow \cdots \Rightarrow G_i)$)

```
begin
    if $bothempty(F_i, G_i)$ or $bothfail(F_i, G_i)$ or
    $occur((F_1 \Rightarrow \cdots \Rightarrow F_i), (G_1 \Rightarrow \cdots \Rightarrow G_i))$ then
        return $true$
    else
        $SF \longleftarrow successor(F_i)$
        $SG \longleftarrow successor(G_i)$
        if $SF \neq \emptyset$ and $SG \neq \emptyset$ then
            $SF' \longleftarrow SF$
            while $SF' \neq \emptyset$ do
                $F' \longleftarrow get\text{-}element(SF')$
                $found\text{-}bisim \longleftarrow false$
                $SG' \longleftarrow SG$
                while $SG' \neq \emptyset$ and $found\text{-}bisim = false$ do
                    $G' \longleftarrow get\text{-}element(SG')$
                    $found\text{-}bisim \longleftarrow bisim2((F_1 \Rightarrow \cdots \Rightarrow F_i \Rightarrow F'), (G_1 \Rightarrow$
                    $\cdots \Rightarrow G_i \Rightarrow G'))$
                if $found\text{-}bisim = false$ then
                    return $false$
            $SG' \longleftarrow SG$
            while $SG' \neq \emptyset$ do
                $G' \longleftarrow get\text{-}element(SG')$
                $found\text{-}bisim \longleftarrow false$
                $SF' \longleftarrow SF$
                while $SF' \neq \emptyset$ and $found\text{-}bisim = false$ do
                    $F' \longleftarrow get\text{-}element(SF')$
                    $found\text{-}bisim \longleftarrow bisim2((G_1 \Rightarrow \cdots \Rightarrow G_i \Rightarrow G'), (F_1 \Rightarrow$
                    $\cdots \Rightarrow F_i \Rightarrow F'))$
                if $found\text{-}bisim = false$ then
                    return $false$
            return $true$
        else
            return $false$
end
```

Proof of Lemma 4. In order to show the termination of `bisim2`, it suffices to repeat the proof of the termination of `bisim1`, with $\prec$ instead of $\ll$. $\square$

Proof of Lemma 5. Let us consider the following property: $(Prop_2((F'_1 \Rightarrow \cdots \Rightarrow F'_j),(G'_1 \Rightarrow \cdots \Rightarrow G'_j)))$ if $F'_j \mathcal{Z}^P_{max} G'_j$ then `bisim2(`$(F'_1 \Rightarrow \cdots \Rightarrow F'_j)$`,`$(G'_1 \Rightarrow \cdots \Rightarrow G'_j)$`)` returns `true`. Again, we proceed by induction, this time with $\prec$ instead of $\ll$. Suppose $((F'_1 \Rightarrow \cdots \Rightarrow F'_j),(G'_1 \Rightarrow \cdots \Rightarrow G'_j))$ is such that for all $((F_1 \Rightarrow \cdots \Rightarrow F_i),(G_1 \Rightarrow \cdots \Rightarrow G_i))$, if $((F_1 \Rightarrow \cdots \Rightarrow F_i),(G_1 \Rightarrow \cdots \Rightarrow G_i)) \prec ((F'_1 \Rightarrow \cdots \Rightarrow F'_j),(G'_1 \Rightarrow \cdots \Rightarrow G'_j))$ then $Prop_2((F_1 \Rightarrow \cdots \Rightarrow F_i),(G_1 \Rightarrow \cdots \Rightarrow G_i))$. Let us show that $Prop_2((F'_1 \Rightarrow \cdots \Rightarrow F'_j),(G'_1 \Rightarrow \cdots \Rightarrow G'_j))$. Suppose $F'_j \mathcal{Z}^P_{max} G'_j$. Hence, for all successors F' of F'_j, there exists a successor G' of G'_j such that $F' \mathcal{Z}^P_{max} G'$, and conversely. Seeing that $((F'_1 \Rightarrow \cdots \Rightarrow F'_j \Rightarrow F'),(G'_1 \Rightarrow \cdots \Rightarrow G'_j \Rightarrow G')) \prec ((F'_1 \Rightarrow \cdots \Rightarrow F'_j),(G'_1 \Rightarrow \cdots \Rightarrow G'_j))$, then by induction hypothesis, $Prop_2((F'_1 \Rightarrow \cdots \Rightarrow F'_j \Rightarrow F'),(G'_1 \Rightarrow \cdots \Rightarrow G'_j \Rightarrow G'))$. Since $F' \mathcal{Z}^P_{max} G'$, then `bisim2(`$(F'_1 \Rightarrow \cdots \Rightarrow F'_j \Rightarrow F')$`,`$(G'_1 \Rightarrow \cdots \Rightarrow G'_j \Rightarrow G')$`)` returns `true`. As a result, one sees that `bisim2(`$(F'_1 \Rightarrow \cdots \Rightarrow F'_j)$`,`$(G'_1 \Rightarrow \cdots \Rightarrow G'_j)$`)` returns `true`. $\square$

Proof of Lemma 6. It suffices to demonstrate that the binary relation $\mathcal{Z}$ defined as follows between Datalog goals is a bisimulation: $F\mathcal{Z}G$ iff there exists SLD-derivations $(F_1 \Rightarrow \cdots \Rightarrow F_i)$, $(G_1 \Rightarrow \cdots \Rightarrow G_i)$ such that $F_i = F$, $G_i = G$ and `bisim2(`$(F_1 \Rightarrow \cdots \Rightarrow F_i)$`,`$(G_1 \Rightarrow \cdots \Rightarrow G_i)$`)` returns `true`. Let F, G be Datalog goals such that $F\mathcal{Z}G$. Hence there exists SLD-derivations $(F_1 \Rightarrow \cdots \Rightarrow F_i)$, $(G_1 \Rightarrow \cdots \Rightarrow G_i)$ such that $F_i = F$, $G_i = G$ and `bisim2(`$(F_1 \Rightarrow \cdots \Rightarrow F_i)$`,`$(G_1 \Rightarrow \cdots \Rightarrow G_i)$`)` returns `true`. Thus, it is easy to verify that the first condition characterizing bisimulations holds for $\mathcal{Z}$ as $F_i = \square$ iff $G_i = \square$. Let F' be a resolvent of F_i and a clause in P. Since `bisim2(`$(F_1 \Rightarrow \cdots \Rightarrow F_i)$`,`$(G_1 \Rightarrow \cdots \Rightarrow G_i)$`)` returns `true`, then there exists a resolvent G' of G_i and a clause in P such that `bisim2(`$(F_1 \Rightarrow \cdots \Rightarrow F_i \Rightarrow F')$`,`$(G_1 \Rightarrow \cdots \Rightarrow G_i \Rightarrow G')$`)` returns `true`, i.e. $F'\mathcal{Z}G'$. Consequently, the second condition characterizing bisimulations holds for $\mathcal{Z}$. For the third condition characterizing bisimulations, a proof similar to the one presented for the second condition applies here. Thus $\mathcal{Z}$ is a bisimulation. $\square$

As a consequence of lemmas 4 – 6, we have:

Theorem 3. `Algorithm 2` *is a sound and complete decision procedure for* (π_{res}).

It follows that (π_{res}) is decidable. Moreover,

Theorem 4. (π_{res}) *is in 2EXPTIME.*

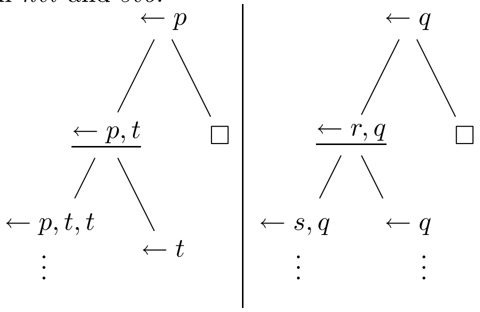
Proof. Let P be a restricted Datalog program and G be a goal. Let n be the maximal number of atoms in the clauses of P or in G, p be the number of predicate symbols in P, a be the maximal arity of the predicate symbols in P, and c be the number of constants in P. Thus, the number of variables in P is about $n \times a$, the number of ground atoms is bounded by $p \times c^a$ and the total number of atoms is bounded by $p \times (c + n \times a)^a$. Moreover, as the number of goals of size n is bounded by $p^n \times (c + n \times a)^{a \times n}$, the number of pairs of goals of size n is bounded by $p^{2\times n} \times (c + n \times a)^{2\times a \times n}$. Then, according to the claim of page 9. The maximal length of SLD-derivations that are considered in `Algorithm 2` will be smaller than $M = p^{2\times n} \times (c + n \times a)^{2\times a \times n}$. Now, replacing D by M in the proof of Theorem 2, one can demonstrate that the time complexity of our algorithm is about $2^M \times s^{2\times M} \leq 2^{p^{2\times n}\times(c+n\times a)^{2\times a\times n}} \times s^{2\times p^{2\times n}\times(c+n\times a)^{2\times a\times n}}$. □

In practice, we can limit the tests performed by `occur` in such a way that `occur(`$(F_1 \Rightarrow \cdots \Rightarrow F_i)$`,`$(G_1 \Rightarrow \cdots \Rightarrow G_i)$`)` returns `true` iff there exists substitutions σ, τ such that $F_i = F_k\sigma$, $G_i = G_k\tau$ for some $1 \leq k < i$.

7 Conclusion

In this paper, we have introduced the concept of bisimulation between datalog goals: two Datalog goals are bisimilar with respect to a given program when their SLD-trees are bisimilar. As proved in Section 4, deciding whether two given goals are bisimilar with respect to a given general logic program is undecidable. Hence, a natural question is to restrict the language of logic programming as done in Section 5 and Section 6. Thus, when the given logic program is hierarchical or restricted, the problem of deciding whether two given goals are bisimilar becomes decidable in 2EXPTIME. The proof of decidability of bisimulation problem for restricted logic program that we presented in Section 6 is based on techniques that were developed in [2] for detecting loops in logic programming. We tried to prove the same result for *nvi* programs and *svo* programs. Unfortunately, applying the techniques of loop detection developed in [2] does not seem to allow us to determine if two given goals are bisimilar with respect to an *nvi* or an *svo* logic program.

Example 3. Let P be the following *nvi*, *svo* program:
$p \leftarrow p, t;$
$p \leftarrow;$
$q \leftarrow r, q;$
$q \leftarrow;$
$r \leftarrow;$
$r \leftarrow s;$
$s \leftarrow$
and let F, G be respectively the following goals $\leftarrow p$ and $\leftarrow q$. Remark that P is both *nvi* and *svo*.



Obviously, F and G are not bisimilar with respect to P. To see this, it suffices to verify that, since the goal $\leftarrow p, t$ has an empty successor whereas all successors of the goal $\leftarrow r, q$ are non empty, then $\leftarrow p, t$ and $\leftarrow r, q$ are not bisimilar with respect to P.

Nevertheless, if one applies the techniques of loop detection developed in [2], then one obtains the following bisimilar reduced SLD-trees:

$$\begin{tikzcd}[column sep=small]
 & \leftarrow p \arrow[dl, dash] \arrow[dr, dash] & & & \leftarrow q \arrow[dl, dash] \arrow[dr, dash] & \\
\underline{\underline{\leftarrow p, t}} & & \square & \underline{\underline{\leftarrow r, q}} & & \square
\end{tikzcd}$$

Acknowledgments. Both authors want to express their appreciation to Yannick Chevalier for his kind help in the course of this research.

References

1. Bol, R.N.: Loop Checking in Partial Deduction. J. Logic Programming 16, 25–46 (1993)
2. Bol, R.N., Apt, K.R., Klop, J.W.: An Analysis of Loop Checking Mechanisms for Logic Programs. Theoretical Computer Science 86, 35–79 (1991)
3. Devienne, P., Lebègue, P., Routier, J.-C.: Halting Problem of One Binary Horn Clause is Undecidable. In: Enjalbert, P., Wagner, K.W., Finkel, A. (eds.) STACS 1993. LNCS, vol. 665, pp. 48–57. Springer, Heidelberg (1993)
4. Gabbrielli, M., Levi, G., Meo, M.C.: Observable Behaviors and Equivalences of Logic Programs 122, 1–29 (1992)
5. Harland, J.: On Normal Forms and Equivalence for Logic Programs. In: Proceedings of the Joint International Conference and Symposium on Logic Programming, pp. 146–160 (1992)
6. Hennessy, M., Milner, R.: On Observing Nondeterminism and Concurrency. In: de Bakker, J.W., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 299–309. Springer, Heidelberg (1980)
7. Hoare, C.A.R.: Communicating sequential processes. Commun. ACM 21, 666–677 (1978)
8. Lifschitz, V., Pearce, D., Valverde, A.: Strongly Equivalent Logic Programs. ACM Transactions on Computational Logic (2000)
9. Lloyd, J.W.: Foundations in Logic Programming. Springer (1984)
10. Maher, M.: Equivalences of Logic Programs. In: Shapiro, E. (ed.) ICLP 1986. LNCS, vol. 225, pp. 410–424. Springer, Heidelberg (1986)
11. Milner, R.: Calculi for synchrony and asynchrony. Theoretical Computer Science 25, 267–310 (1983)
12. Park, D.: Concurrency and Automata on Infinite Sequences. In: Proceedings of the 5th GI-Conference on Theoretical Computer Science, pp. 167–183. Springer, UK (1981)
13. Sangiorgi, D.: On the origins of bisimulation and coinduction. In: ACM Trans. Program. Lang. Syst., pp. 1–41. ACM, USA (2009)

Inconsistency Management for Traffic Regulations: Formalization and Complexity Results[⋆]

Harald Beck, Thomas Eiter, and Thomas Krennwallner

Institute of Information Systems, Vienna University of Technology
Favoritenstrasse 9–11, A-1040 Vienna, Austria
{hbeck,eiter,tkren}@kr.tuwien.ac.at

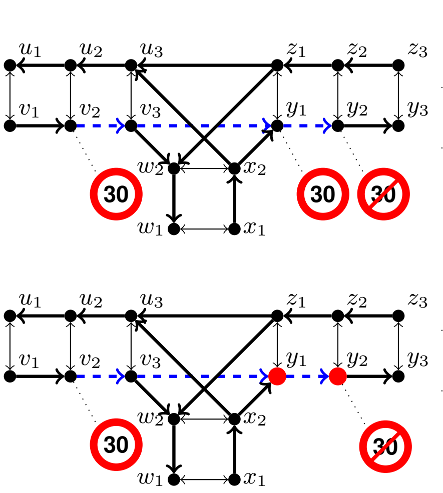
Abstract. Smart Cities is a vision driven by the availability of governmental data that fosters many challenging applications. One of them is the management of inconsistent traffic regulations, i.e., the handling of inconsistent traffic signs and measures in urban areas such as wrong sign posting, or errors in data acquisition in traffic sign administration software. We investigate such inconsistent traffic scenarios and formally model traffic regulations using a logic-based approach for traffic signs and measures, and logical theories describe emerging conflicts on a graph-based street model. Founded on this model, we consider major reasoning tasks including consistency testing, diagnosis, and repair, and we analyze their computational complexity for different logical representation formalisms. Our results provide a basis for an ongoing implementation of the approach.

1 Introduction

The advent of the World Wide Web and distributed systems brought numerous new methods for intelligent management of data and knowledge. With initiatives such as Open Government Data (http:/opengovernmentdata.org/) the idea of Smart Cities has been gaining interest in research communities, with many innovative applications in ecological and city planning areas. Local governments manage their posted traffic signs and measures using software tools, i.e., authorities enact rules how traffic on urban streets and places should be regulated, and employees increasingly maintain this information with the help of specialized software. An important task is the management of inconsistent traffic regulations.

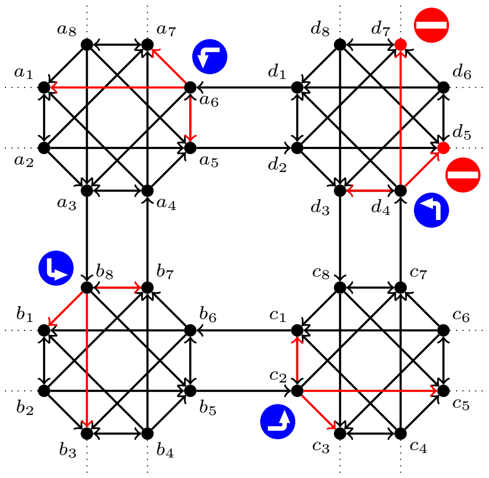
Example 1. Consider the T-junction in the top of Fig. 1a. It has three arms, each represented by two parallel lanes: u_3 to u_1 and v_1 to v_3, w_2 to w_1 and x_1 to x_2, and y_1 to y_3 and z_3 to z_1. We can turn from one arm to each other and may reverse between nodes connected by edges with two arrows. The traffic signs at v_2, y_1, and y_2 symbolize a correct sign posting for a speed limit measure of 30 km/h, indicated by the dashed blue path from v_2 to y_2. The *effect* expressed by both the measure and the signs is that along the edges (v_2, v_3), (v_3, y_1), (y_1, y_2), the maximum speed allowed for any road user is 30 km/h. The recurrent start sign at y_1 is necessary, as road users coming from x_2, turning into the lane starting at y_1, would otherwise be unaware of the speed

⋆ Supported by PRISMA solutions EDV-Dienstleistungen GmbH, and the Austrian Science Fund (FWF) projects P20841 and P24090.



u_1 u_2 u_3 z_1 z_2 z_3
v_1 v_2 v_3 y_1 y_2 y_3
w_2 x_2 w_1 x_1
30

(a) Top: correct sign posting for a speed limit measure (dashed blue path). Bottom: Inconsistency: no recurrence of start sign at y_1



a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8
b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8
c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8
d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8

(b) Loop caused by mandatory left turns

Fig. 1. Traffic Regulation Scenarios

limit. This situation is shown in the bottom of Fig. 1a. The effect of the start sign at v_2 can only be propagated to the arm starting at y_1, since y_1 is also reachable from the arm ending in x_2. We get an inconsistent traffic regulation due to two conflicts: the speed limit effect ends at y_1 without an end sign, and the effect discontinuing at y_2 does not properly start.

Such inconsistencies create problems in daily traffic. Officials are confronted with legal issues (e.g., challenging of speeding tickets) when two dissenting speed limits are announced. Even more delicate is the aspect of legal responsibility in case of accidents caused by wrong sign posting. Different from that, errors in the data acquisition in traffic sign software may lead to wrong assumptions on the state of traffic regulations. Tools that detect, prohibit, and correct such errors are in need to help public administration with their traffic management tasks.

In order to gain new insights from available sign posting data, formal methods from knowledge representation and reasoning proved to be a key to attack issues that arise when data is inconsistent [16,12,10]. Many issues arise in the context of traffic regulations. Traffic measures, i.e., intended constraints given as regulations on the traffic, may oppose the state of traffic sign posting, which can be seen as real-world constraints that announce what is allowed on the street. One natural question is how to find inconsistencies when combining traffic measures and street signs. Such questions become even more complex in dynamic environments, i.e., when so-called *active traffic management* comes into play. For instance, variable-message signs on motorways manage the traffic flow by varying speed limits based on events like traffic congestions, or weather conditions like fog or black ice. Contradicting speed limits may be posted by operators of such message signs, leading to aforementioned legal issues.

Finding such errors is not trivial in real life situations and many subtle inconsistencies may occur. When an inconsistency is found, one usually wants to diagnose and repair it. To the best of our knowledge, there is no automated support for inconsistency finding in complex traffic regulations. Already the seemingly simple scenario in Example 1 shows the need for (semi-)automatic tool support in traffic regulation maintenance software. Different from the issues above is the problem of modeling transportation and traffic in a formal representation. Legal texts are ambiguous and often implicitly understood, and no single characterization has yet shown to be advantageous over others.

This motivates this work, which makes the following contributions:

- We analyze the problem domain and identify main concepts and notions such as traffic signs, measures, effects, and inconsistencies in traffic regulation orders.
- Building upon well-known literature in abductive reasoning and model-based diagnosis [16,12,10], we develop a formal model using predicate logic for traffic signs and measures, and introduce the formal notion of a *traffic regulation problem*.
- For traffic regulation problems, we consider major reasoning tasks, viz. inconsistency detection, diagnosis, correspondence between measures and signs, and repair.
- We then study and characterize the computational complexity of these reasoning tasks, for different representation formalisms. In particular, we consider first-order (FO) logic under domain closure (as the domain of discourse is fixed), and answer set programming (ASP). The latter is convenient for developing executable specifications, and provides attractive features that can be used for default rules and exception handling.

This work is embedded in an industrial context, dealing with specialized software, which is used by local government departments and allows for the visualization and administration of traffic regulations. The results of this research may assist to find inconsistencies and should give a clear advantage over simple traffic sign acquisition and storage tools. A prototype implementation using answer set programming is in progress.

2 Domain Analysis

In this section we briefly analyze the domain of traffic regulations, measures and signs.

A *traffic regulation* is a legal document describing how road users can use the street and how these usages can be restricted by means of *traffic signs*. The legal act to introduce new traffic signs, or to remove existing ones, is a *traffic regulation order*, which comes in form of a document describing (in natural language) a *traffic measure* that has to be taken to reach a desired effect, i.e., a restriction of road usage. This measure has to be *announced* by traffic signs and becomes legally effective as soon as the corresponding signs are posted on the street. We view road markings as special cases of traffic signs.

The restrictions described by measures and signs include speed limits, driving bans, parking or halting bans, prohibited or mandatory driving directions, information about zones like residential areas and pedestrian zones, motorways, and so on. We base our work on the Austrian traffic regulation and its potential measures and signs, which can be found at http://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10011336. However, we focus on general aspects that are not bound to regional differences.

Inconsistencies. In general, a set of traffic regulation orders, resp. the resulting measures and signs, can lead to *conflicts* with respect to the traffic regulation. The aim of our work is to *detect* such inconsistencies, to *diagnose* and to *repair* them.

For instance, in Austria it is not allowed that a motorway overlaps with a residential area. In view of traffic signs this means that, when driving on a motorway, the end sign must precede the start sign of the residential area. In addition to such illustrative cases, complications arise quickly when many different kinds of restrictions are expressed.

What we understand by a conflict does not necessarily stem from the traffic regulation, but can also come from supplementary documents of expert knowledge such as traffic planning experience. It is thus our aim to provide a system that can detect different sorts of conflicts in a modular and easily extendable way. Whenever a conflict is detected, we want to provide the user with diagnostic information, explaining which measures or signs caused it. Finally, we want to offer a repair mechanism that suggests by which modifications compliance with the specification can be established.

Data Model and Approach. To achieve these goals, we first need a street model based on which we can express measures and signs, and the restrictions expressed by them. We will view streets as *directed graphs*, where edges represent the potential direction of traffic. Each edge will get a unique label to discern whether it represents a part of a lane, a turn over a junction or a U-turn. Any digital street map from which this view can be generated can be used as potential database.

By an *effect* of both measures and signs we understand the implicit restrictions they express. To reflect measures and signs (from a database or user input) in the street graph, we will use predefined labels on the edges (for measures) and nodes (for signs). Similarly, we will represent arising inconsistencies by associating nodes with specific *conflict labels*. Both the mapping from measures and signs to effects and from effects to conflicts will be established in a modular way by means of logic formulas. The conflict labeling can be used to visualize inconsistencies on a street map, followed by user interaction in connection with diagnosis and repair.

Challenges. Many conflicts will not be strictly illegal as defined by the traffic regulatory orders or additional legal documents, but arise from expert knowledge or common sense. Consequently, both the inclusion and the kind of definition of many conflicts will be a matter of preference, and shall in principle be configurable by domain experts.

To show the need for advanced reasoning support, consider the traffic regulation problem in Fig. 1b, where a *loop* is induced by four mandatory left turns. We consider loops as special dead ends which we want to detect whenever a node v has a way in, but no way out. We say a node v has a *way in*, if it is predefined as *in-node*, or if it is reachable from an in-node. Similarly, a node v has a *way out*, if it is a predefined *out-node*, or an out-node is reachable from v.

A node w is *reachable* from v, if (i) (v, w) is an edge, where neither the node v is prohibited for traffic (e.g., through a no-entry sign), nor the edge itself (e.g., through a mandatory turn in a different direction), or (ii) if a node x is reachable from v, from which w is reachable.

Example 2. The mandatory left turns in Fig. 1b induce a loop along the nodes $L = \{a_6, a_3, b_8, b_5, c_2, c_7, d_4, d_1\}$. Respective in-nodes and out-nodes are not depicted and assumed to be reachable from the nodes with dotted lines. For instance, from a_1, an out-node is reachable, and a_2 is reachable from an in-node. Each mandatory left turn prohibits the right turn, U-turn, and edge straight ahead over the junction. E.g., the mandatory left turn at a_6 prohibits moves along the edges (a_6, a_7), (a_6, a_5) and (a_6, a_1).

We try to keep the street model as simple as possible. Reversing along lanes is not a typical road usage. Therefore, we do not model any intermediate nodes along streets (and thus no U-turns within lanes), unless we need to represent a sign, or the start or end of a measure. In reality, we could in principle escape the loop in Example 2 by reversing somewhere along a lane. Since this is not supposed to be necessary, we still want an evaluation to report that there are problems, by noting conflicts on the nodes $v \in L$.

Intuitively, the unique minimal explanation (i.e., a *diagnosis*) for each of these problematic nodes—which can be seen as one conflict across many nodes—consists of all four mandatory left turns. Note that additional signs that do not restrict the reachability, like speed limits, would not change this diagnosis.

To *repair* the scenario, i.e., make modifications such that the result is free of conflicts, we may delete one of the mandatory left turns on nodes a_6, b_8, or c_2. However, other possibilities exist (see Section 6).

3 Formal Model

In this section we formalize our data model and formulate a traffic regulation problem based on it. Throughout, we assume that a version of predicate logic $\mathcal{L}$ with negation is fixed, in which the desired specification can be expressed (e.g., FO logic or ASP).

Definition 1 (Street graph). *A* (street) graph *is a connected, labeled, directed graph $G = (V, E, \ell)$ of nodes V, edges $E \subseteq V \times V$, and a labeling function ℓ that assigns each edge $(v, w) \in E$ a unique label $\ell(v, w) \in \{left, straight, right, lane, uturn\}$.*

We identify G with the set of atoms $e(t, v, w)$, where t is the label of the edge (v, w). Several assumptions about the structure of these graphs are made. For instance, we intend to model junctions by means of edges with labels *left*, *right* and *straight*. For each such edge (v, w), all incoming edges $(x, v) \in E$ to node v are labeled *uturn* or *lane*.

Example 3 (cont'd). Fig. 1a suggests how the edge labels ought to be used. For instance, the edge (v_3, y_1) models the direction straight ahead over a junction and thus gets the label *straight*. All other edges (v_i, v_{i+1}) and (y_i, y_{i+1}) are labeled with *lane*. The incoming street from below has a turn to the right starting at x_2 and ending at y_1, which will be modeled by an atom $e(right, x_2, y_1)$. Similarly, we use $e(left, x_2, u_3)$ for the left turn at x_2. The edges with arrows on both ends depict U-turns in both directions.

In the formulation of measures in traffic regulation orders concepts like street names, addresses and cardinal points are used to describe the intended topological dimensions. We assume that for the description at hand, a preprocessing (or specification) maps this scope to edges. We thus reduce measure descriptions to sets of such "atomic measures."

To describe measures, signs, their effects, as well as conflicts, we build upon disjoint sets of ground terms M, S, F and C called the *measure types, sign types, effect types* and *conflict types*, respectively. For instance, M may contain a set of terms $spl(k)$ for each speed limit value k that is needed, e.g., $spl(5)$, $spl(10)$, $\ldots$, $spl(130)$ in Austria.

Definition 2 (Measures, Signs, Effects, Conflicts). *Given a street graph G, we define the following sets of atoms:*

- *Measures* $M_G = \{m(t, v, w) \mid t \in \mathsf{M}, (v, w) \in E\}$*;*
- *Signs* $S_G = \{s(t, v) \mid t \in \mathsf{S}, v \in V\}$*;*
- *Input* $I_G = M_G \cup S_G$*;*
- *Effects* $F_G = \{f(t, v, w) \mid t \in \mathsf{F}, (v, w) \in E\}$*; and*
- *Conflicts* $C_G = \{c(t, v) \mid t \in \mathsf{C}, v \in V\}$.

For instance, to represent the prohibited case that a motorway overlaps with a residential area at a node v we might use $c(overlap(motorway, residential\text{–}area), v)$. Similarly, the fact that one is caught in a dead end or loop at u can be represented as $c(no\text{-}way\text{-}out, u)$.

Definition 3 (Scenario). *Let G be a street graph, $M \subseteq M_G$ be a set of measures on G, and $S \subseteq S_G$ be a set of signs on G. Then, $Sc = (G, M, S)$ is called a* scenario.

Example 4 (cont'd). In Fig. 1a, the dashed blue path from v_2 to y_2 symbolizes a 30 km/h speed limit measure. We formalize this as a set of atomic measures $\{m(spl(30), v_2, v_3), m(spl(30), v_3, y_1), m(spl(30), y_1, y_2)\}$. The depicted traffic signs are defined at nodes as the set $\{s(start(spl(30)), v_1), s(start(spl(30)), y_1), s(end(spl(30)), y_2)\}$.

Effects and Conflicts. The meaning of both measures and signs is captured by a mapping of the according languages to a common target language of effects. To assist modular composition, we define $\overline{X}_Y = X \cup \{\neg x \mid x \in Y \setminus X\}$ as the *closed world operator* applied to a set of ground atoms X relative to a *base set* $Y \supseteq X$. We always use the according base set of Definition 2, and thus omit the subscript, e.g., $\overline{M}$ for a set measures M on G abbreviates $\overline{M}_{M_G}$. The base set assumed for the completion $\overline{G}$ of any graph G is the set of all atoms $e(t, v, w)$. We introduce another operator Cn that maps between atoms. Let X and Y be sets of atoms (on G) and let T be a set of formulas in $\mathcal{L}$.

Definition 4 ($Cn_G(T, X, Y)$). *The* Y-consequences of T and X (on G) *is the set of atoms* $Cn_G(T, X, Y) = \{y \in Y \mid T \cup \overline{G} \cup \overline{X} \models y\}$.

Here, $\models$ is the (logical) consequence relation in the underlying logic $\mathcal{L}$. The closed world operator makes sure that atoms that are not entailed are set to false, and thus ensures that valuations of atoms in Y are unique. This restriction will enable modular composition by means of a two-stage approach, which we will describe next.

Definition 5 (Effect mapping). *An* effect mapping *is a set P of formulas in $\mathcal{L}$ that associates with each input $I \subseteq I_G$ on a street graph G the set $\mathcal{F}_G^P(I) = Cn_G(P, I, F_G)$ of atoms, called* effects of I (on G).

We implicitly assume that effect mappings use the ranges of terms appropriately.

Example 5. The first-order sentence

$$\forall k, x, y\, (m(spl(k), x, y) \supset f(max\text{-}speed(k), x, y))$$

of an effect mapping P captures the meaning of speed limit (spl) measures. We informally describe when this effect label is obtained by signs: first, an edge (x, y) is labeled with $max\text{-}speed(k)$, if an start sign $s(start(spl(k), x))$ is placed at x. From there, the effect is propagated in the direction of traffic, i.e., along the edges with label $lane$, until an end sign or a junction is reached. For the latter case, let $e(lane, u', u)$ be the last edge before the junction and $e(straight, u, v)$ be the next edge in the direction ahead. The effect continues after the crossroads on the (unique) edge $e(lane, v, w)$ only if another start sign is posted on v, or no edge (x, v) with label $left$ or $right$ permitted for traffic exists (and neither an end sign nor the start sign for a different speed limit is at v).

The effect mapping uses measures and signs on a graph to derive effects. Likewise, these effect atoms will then be used to infer conflicts by means of a specification.

Definition 6 (Conflict specification). *A* conflict specification over an effect mapping P *is a set* Sp *of formulas in* $\mathcal{L}$ *that associates with each input* $I \subseteq I_G$ *on a street graph* G *the set* $\mathcal{C}_G^{P,Sp}(I) = Cn_G(Sp, \mathcal{F}_G^P(I), C_G)$ *of atoms, called* conflicts of I (on G).

Thus, the setup to compute conflicts based on effects given a conflict specification, is the same as computing effects from measures and signs, given an effect mapping. The first stage builds a context-dependent model of the input, the second stage establishes the basis for reasoning tasks. There is no explicit support for query answering on top of conflicts; however, queries on aspects of interest may be encoded in the conflict specification, using designated conflicts and formulas defining them (e.g. rules) in Sp. This way, given an input I on a graph G, querying for a certain conflict type (or aspect of interest) $t \in \mathsf{C}$ amounts to computing the set $\{c(t, v) \in \mathcal{C}_G^{P,Sp}(I) \mid v \in V\}$.

Example 6 (cont'd). Fig. 1a (bottom) depicts the situation in which the intended speed limit is not sufficiently announced. Road users coming from node x_2, turning right into the lane starting at y_1 are not informed about the speed limit. Hence, according to the sign posting, the $max\text{-}speed(30)$ effect cannot be associated with edge (y_1, y_2). Since we have a $max\text{-}speed(30)$ effect until node y_1 but no end sign mapped to it, we have a conflict which we may represent as $c(no\text{-}end(max\text{-}speed(30)), y_1)$. The end sign posted at y_2 leads to a second conflict, since there is no "open" effect anymore: $c(cant\text{-}end(max\text{-}speed(30)), y_2)$. Using answer set programming, with uppercase letters denoting variables as usual, the latter conflict can be defined by the rule

$$c(cant\text{-}end(F), V_2) \leftarrow in\text{-}dir(V_1, V_2), s(end(T), V_2), m2f(T, F), \text{not}\, f(F, V_1, V_2)\ ;$$

where $in\text{-}dir$ represents an edge of type $straight$ or $lane$, the atom $s(end(T), V_2)$ stands for a traffic sign posted at node V_2, ending a measure of type T, and $m2f$ encodes domain knowledge that T is associated with effect type F.

Note that ASP solvers like DLV support query functionalities in the aforementioned sense. In the previous example, we might ask $c(cant\text{-}end(F), V)?$ and get the terms $max\text{-}speed(30), y_2$ as result, matching $c(cant\text{-}end(max\text{-}speed(30)), y_2)$.

Definition 7 (Traffic Regulation Problem). *Let* Sp *be a conflict specification over an effect mapping* P*, and let* Sc *be a scenario. Then, the pair* $\Pi = (Sp, P)$ *is called a* traffic regulation *and the pair* (Π, Sc) *a* traffic regulation problem.

4 Reasoning Tasks

We now use the preceding definitions to specify some practically relevant use cases in form of reasoning tasks. In the sequel, we let $\mathcal{T} = (\Pi, Sc)$ be a traffic regulation problem with a traffic regulation $\Pi = (Sp, P)$ and a scenario $Sc = (G, M, S)$, and $I = M \cup S$.

Definition 8 (Inconsistency). *The* conflicts of $\mathcal{T}$ *are given by* $\mathcal{C}(\mathcal{T}) = \mathcal{C}_G^{P,Sp}(I)$. *If* $\mathcal{C}(\mathcal{T}) \neq \emptyset$*, we call* $\mathcal{T}$ inconsistent.

Additionally, we call every set of measures or signs $X \subseteq I_G$ on graph G *inconsistent*, if $\mathcal{C}_G^{P,Sp}(X)$ is non-empty. Given an inconsistent $\mathcal{T}$, we are interested which part of the input, i.e., which hypotheses, explain the conflict observations.

Definition 9 (Diagnosis). *For inconsistent* $\mathcal{T}$*, a* diagnosis *of a set of conflicts* $C \subseteq \mathcal{C}(\mathcal{T})$ *is a set* $J \subseteq I$*, such that* $C \subseteq \mathcal{C}_G^{P,Sp}(J)$.

To see the relation of diagnosis with the usual notion of abductive diagnosis [15,3], we recall the definition of the latter.[1] An *abductive diagnosis problem (ADP)* is a triple $\langle T, H, O\rangle$, where T is a set of formulas in $\mathcal{L}$, called the theory, and H and O are sets of literals, called the hypotheses and observations, respectively. A *(complete) abductive diagnosis for* $\langle T, H, O\rangle$ is a set $A \subseteq H$, such that $T \cup \overline{A} \not\models \bot$ and $T \cup \overline{A} \models O$. We note that an input $J \subseteq I$ is the abductive diagnosis for the ADP $\langle P \cup \overline{G}, I, \mathcal{F}_G^P(J)\rangle$.

Proposition 1. *Let* $C \subseteq \mathcal{C}(\mathcal{T})$ *and* $J \subseteq I$*. The effects* $\mathcal{F}_G^P(J)$ *are an abductive diagnosis for the ADP* $\langle Sp \cup \overline{G}, \mathcal{F}_G^P(I), C\rangle$ *iff* J *is a consistent diagnosis of* C*, i.e.,* $Sp \cup \overline{G} \cup \overline{\mathcal{F}_G^P(J)} \not\models \bot$.

Since I is always a trivial (but non-informative) diagnosis for any set of conflicts, we are interested in (subset-)*minimal* diagnoses. We omit a formal definition of Π serving the forthcoming examples.

Example 7 (cont'd). The missing sign at y_1 leads to two conflicts. The minimal diagnosis for the missing sign $\{c(\textit{no-end}(\textit{max-speed}(30)), y_1)\}$ is $\{s(\textit{start}(\textit{spl}(30)), v_2)\}$. Independently, the other conflict $\{c(\textit{cant-end}(\textit{max-speed}(30)), y_2)\}$ is minimally explained by $\{s(\textit{end}(\textit{spl}(30)), y_2)\}$.

Usually, we desire that measures and signs in a scenario express the same effects.

Definition 10 (Correspondence). *A set of measures* M *and a set of signs* S correspond *with respect to* P *and* G*, if it holds that* $\mathcal{F}_G^P(M) = \mathcal{F}_G^P(S)$.

The next example shows the significance of correspondence besides consistency.

Example 8 (cont'd). Suppose a no-right turn sign on x_2 is added to the traffic regulation problem in Fig. 1a, bottom. This results in a consistent scenario, since in this case, the node y_1 can only be reached from v_3 (reversing at z_1 along the U-turn (z_1, y_1) is disregarded). Hence, another start sign at y_1 is not necessary and the effect propagation of the start sign at v_2 continues through y_1. However, the prohibition of traffic along the edge (x_2, y_1) as supported by the no-right turn sign is not supported by a corresponding measure. This problem can only be seen by additionally testing for correspondence.

For each of the definitions in this section, we immediately obtain a *reasoning task* which requires the computation of the respective concept.

Repair. Complementary to diagnoses explaining the cause for inconsistency, a natural question is how to *repair* an inconsistent traffic regulation problem, i.e., by means of which deletions and additions of measures and signs consistency can be established. In the general case, we might delete and add both measures and signs.

[1] Strictly speaking, we present a slightly modified version using the closed world operator.

Definition 11 (Repair). *A* repair *of an (inconsistent) $\mathcal{T}$ is a pair (I^-, I^+) such that $I^- \subseteq I$, $I^+ \subseteq I_G \setminus I$, and $\mathcal{C}_G^{P,Sp}(I') = \emptyset$, where $I' = (I \backslash I^-) \cup I^+$.*

A repair yields a traffic regulation problem $\mathcal{T}'$ replacing I with consistent I' in $\mathcal{T}$.

Example 9 (cont'd). A repair for the inconsistent $\mathcal{T}$ with the traffic regulation scenario as shown in Fig. 1a (bottom) is $(\emptyset, \{s(start(spl(30), x_4)\})$.

Usually, one requires correspondence after a repair, i.e., that $M' = I' \cap M_G$ and $S' = I' \cap S_G$ correspond with respect to P and G; we call such repairs *strict*. Furthermore, the candidate space (I^-, I^+) might be restricted; in this way, further practically relevant reasoning tasks can be formulated as special cases of repair. For instance, if we are given a scenario with consistent M, but inconsistent S, we may *adjust* the signs by restricting the repair to modify only signs. Or, related to data imports, we may want to *generate* measures from scratch, given only signs, or vice versa a sign posting, given only measures. Note that these additional reasoning tasks need no separate implementation.

5 Computational Complexity

In this section, we analyze the computational complexity of decision problems associated to the reasoning tasks above. In particular, we consider for a given traffic regulation problem $\mathcal{T} = (\Pi, Sc)$, with $\Pi = (Sp, P)$ and $Sc = (G, M, S)$

- CONS: decide whether $\mathcal{T}$ is consistent, i.e., $\mathcal{C}(\mathcal{T}) = \mathcal{C}_G^{P,Sp}(I) = \emptyset$;
- UMINDIAG: decide, given a set $C \subseteq \mathcal{C}(\mathcal{T})$ of conflicts, whether C has a unique $\subseteq$-minimal diagnosis, i.e., a single minimal $J \subseteq I$ such that $C \subseteq \mathcal{C}_G^{P,Sp}(J)$;
- CORR: decide whether M and S correspond, i.e., $\mathcal{F}_G^P(M) = \mathcal{F}_G^P(S)$;
- REPAIR: decide, given $\mathcal{T}$ is inconsistent, whether some admissible repair exists, i.e., some $I^+, I^- \subseteq I_G$ such that $\mathcal{C}_G^{P,Sp}((I \setminus I^-) \cup I^+) = \emptyset$ and a polynomial-time admissibility predicate $\mathcal{A}(I^+, I^-)$ holds.

We consider these problems for different mapping formalisms $\mathcal{L}$, viz. (1) FO predicate logic under domain closure (FOL+DCA), i.e., an axiom $\forall x. \bigvee_{i=1}^{n}(x = c_i)$, where $c_1, \ldots, c_n$ is the (finite) set of constant symbols; and (2) (function-free) Answer Set Programs[2] under cautious consequence, i.e., $P \models \alpha$ iff α is true in all answer sets of P; here, we consider various classes, including (a) stratified programs (ASP$^{\neg_s}$), (b) normal programs (i.e., arbitrary negation, ASP$^{\neg}$), and (c) disjunctive programs (with head disjunction and arbitrary negation, ASP$^{\vee,\neg}$).

We assume that the reader is familiar with the basic concepts of complexity theory (cf. [14]),and recall that P^O (resp. NP^O) is (nondeterministic) polynomial time computability with an oracle for complexity class O, and Σ_i^p, $i \geq 1$, are classes of the polynomial hierarchy where $\Sigma_1^p = \mathsf{NP}$ and $\Sigma_{i+1}^p = \mathsf{NP}^{\Sigma_i^p}$. Furthermore, $\mathsf{P}_{\|}^O$ is the restriction of P^O that all oracle queries are independent of each other, i.e., they are evaluable in parallel.

The computational complexity of the atom entailment problem in these logics (IMPL), i.e., deciding $T \models \alpha$ for a set of formulas (resp. program) T and an atom α, is shown

[2] The use of function symbols as in the examples is convenient but not essential for this domain.

Table 1. Complexity of reasoning tasks (general case / bounded predicate arities); unless stated otherwise, entries are completeness results

Logic $\mathcal{L}$	IMPL	CONS	CORR	UMINDIAG	REPAIR
FO+DCA	co-NExp / PSpace	$\mathsf{P}_{\parallel}^{\mathsf{NExp}}$ / PSpace			$\mathsf{NP}^{\mathsf{NExp}}$ / PSpace
$\text{ASP}^{\neg_s}$	Exp / P^{NP}	Exp / P^{NP}		Exp / in $\mathsf{P}_{\parallel}^{\Sigma_2^p}$, Π_2^p-hard	Exp / Σ_2^p
$\text{ASP}^{\neg}$	co-NExp / Π_2^p	$\mathsf{P}_{\parallel}^{\mathsf{NExp}}$ / $\mathsf{P}_{\parallel}^{\Sigma_2^p}$		$\mathsf{P}_{\parallel}^{\mathsf{NExp}}$ / in $\mathsf{P}_{\parallel}^{\Sigma_3^p}$, Π_3^p-hard	$\mathsf{NP}^{\mathsf{NExp}}$ /Σ_3^p
$\text{ASP}^{\vee,\neg}$	co-$\mathsf{NExp}^{\mathsf{NP}}$ / Π_3^p	$\mathsf{P}_{\parallel}^{\mathsf{NExp}^{\mathsf{NP}}}$ / $\mathsf{P}_{\parallel}^{\Sigma_3^p}$		$\mathsf{P}_{\parallel}^{\mathsf{NExp}^{\mathsf{NP}}}$ / in $\mathsf{P}_{\parallel}^{\Sigma_4^p}$, Π_4^p-hard	$\mathsf{NP}^{\mathsf{NExp}^{\mathsf{NP}}}$ / Σ_4^p

in the first column of Table 1. Besides the general case, also the one of bounded predicate arities (BPA) is printed, i.e., when the arities of predicates is bounded by some constant. Briefly, as for FOL+DCA, a countermodel of $T \models \alpha$ (of exponential size) can be guessed and verified in polynomial space (in the size of T and α); hardness follows, e.g., from the complexity of satisfiability of the Bernays-Schönfinkel fragment of FOL. Under BPA, the model guess has polynomial size, and thus the whole countermodel check is feasible in polynomial space; PSpace-hardness is inherited from evaluation of a given FOL formula over a finite structure. For the ASP^X languages, see [4,5].

Theorem 1. *For* CONS, UMINDIAG, CORR, *and* REPAIR *the results in Table 1 hold.*

In the rest of this section, we explain the results and outline how they can be derived. We make use of the following known fact; let $\mathsf{P}^{O}_{\parallel[k]}$ be the restriction of P^{O} such that the oracle calls amount to k rounds of parallel (independent) oracle calls.

Lemma 1. *For* $O = \mathsf{NExp}, \mathsf{NExp}^{\mathsf{NP}}, \Sigma_i^p$, $i \geq 1$, *and constant* k, $\mathsf{P}^{O}_{\parallel[k]} = \mathsf{P}^{O}_{\parallel[1]} = \mathsf{P}^{O}_{\parallel}$.

CONS. To decide problem CONS, we must test whether some of the (polynomially many) conflict facts $c(t, v)$ is derivable from the conflict specification Sp over the effect mapping P. To this end, we can determine for each of the (polynomially many) effect facts $f(t, v, w) \in F_G$ whether $f(t, v, w) \in \mathcal{F}_G^P(I) = Cn_G(P, I, F_G)$ with an oracle for IMPL, and then decide whether $c(t, v) \in \mathcal{C}_G^{P,Sp}(I) = Cn_G(Sp, \mathcal{F}_G^P(I), C_G)$ with an oracle for IMPL. This is a polynomial time computation with two rounds of parallel evaluation of oracle queries with complexity O; this puts the problem in the respective complexity class $\mathsf{P}^{O}_{\parallel[2]}$. For $O = \mathsf{Exp}, \mathsf{PSpace}$, this class coincides with O and for $O = \mathsf{P}^{\mathsf{NP}}$ with P^{NP} (as oracles can be simulated, in case of an P^{NP} oracle with an O oracle); for the other classes, by the lemma it coincides with $\mathsf{P}^{O}_{\parallel}$.

The $\mathsf{P}^{O}_{\parallel}$-hardness results for CONS are derived by a reduction from the following $\mathsf{P}^{O}_{\parallel}$-complete problem EVEN: given instances $I_0, \ldots, I_{2n+1}$, $n \geq 0$, of a (fixed) O-complete problem, decide whether the number of yes-instances among them is even. Here, without loss of generality, we can assume that all yes-instances precede all no-instances. To encode this problem, we use problem IMPL and restrict (wolog) to instances $I_j : T^{(j)} \models \alpha_j$ in a way such that their answers amount to $P \models \phi_j$ for effect mapping P and effect ϕ_j. We then design the conflict specification Sp to derive a conflict fact χ if and only if the maximum index of a yes-instance is even. To this end, we can use in the ASP cases rules $\chi \leftarrow \phi_{2j}, \text{not}\, \phi_{2j+1}$, where $0 \leq j \leq n$, and in the FOL case simply the formula $\chi \leftrightarrow \bigvee_{j=0}^{n} \phi_{2j} \wedge \neg\phi_{2j+1}$.

UMinDiag. To test whether some set $J \subseteq I$ is a diagnosis, we need to check $C \subseteq \mathcal{C}_G^{P,Sp}(I)$; similarly as deciding Cons, the latter problem can be shown to be in $\mathsf{P}_{\|}^{C}$. For $O = \mathsf{Exp}, \mathsf{PSpace}$ it is then clear that an algorithm can cycle through all $J \subseteq I$ and compute two minimal diagnoses in exponential time (resp. polynomial space), provided two exist (one always exists). For other O, the following more involved method works.

Like for Cons, we first compute with a round of parallel O oracle calls the effects $\mathcal{F}_G^P(I)$. Then, in a next round, we ask oracles, for $k = 0, \ldots, |I|$, whether for all $J \subseteq I$ of size $|J| = k$, it holds that $C \not\subseteq \mathcal{C}_G^{P,Sp}(J)$; for the considered $O \supseteq \mathsf{NExp}$, each oracle call is in O (exponentially many J's can be considered without complexity increase), while for the other classes it is in NP^O. The smallest k for which the oracle answers "no" is the size k^* of a *smallest* (in terms of cardinality) diagnosis, and thus of some minimal diagnosis. In a further round we then ask an oracle whether for all $J_1, J_2 \subseteq I$ such that $|J_1| = k^*$ and $J_1 \not\subseteq J_2$ it holds that either $C \not\subseteq \mathcal{C}_G^{P,Sp}(J_1)$ or $C \not\subseteq \mathcal{C}_G^{P,Sp}(J_2)$; the answer will be "yes" iff a unique minimal diagnosis exists. Overall, the method uses three rounds of O (resp., NP^O) oracle calls, which puts the problem in the class $\mathsf{P}_{\|[3]}^{O}$ resp. $\mathsf{P}_{\|[3]}^{\mathsf{NP}^O}$; this class, however, by the lemma coincides with $\mathsf{P}_{\|}^{O}$ resp. $\mathsf{P}_{\|}^{\mathsf{NP}^O}$; and for $O = \mathsf{P}^{\mathsf{NP}}$, we have $\mathsf{NP}^{\mathsf{P}^{\mathsf{NP}}} = \mathsf{NP}^{\mathsf{NP}}$, thus $\mathsf{P}_{\|}^{\mathsf{NP}^{\mathsf{P}^{\mathsf{NP}}}} = \mathsf{P}_{\|}^{\Sigma_2^p}$.

The hardness results for $O \supseteq \mathsf{NExp}$ are obtained by a reduction of the complement of the Even problem (which is also $\mathsf{P}_{\|}^{O}$-hard) that is a variant of the reduction for the Cons problem. The conflict rules are extended to $\chi \leftarrow \psi_{2j}, \text{not}\, \psi_{2j+1}, in_1, 0 \leq j \leq n$, and a further rule $\chi \leftarrow in_0$ is added, where in_0 and in_1 are fresh input facts. Then, $J = \{in_0\}$ is a minimal diagnosis, and it is the single one iff $J = \{in_1\}$ is not a diagnosis, which is the case iff the Even instance is a no-instance. For the remaining cases of O, one can show hardness for co-NP^O by a reduction from evaluation of suitable QBFs; however, hardness for NP^O, let alone for $\mathsf{P}_{\|}^{\mathsf{NP}^O}$ is not apparent.

Corr. It is easy to see that Corr is solvable in polynomial time with parallel oracle queries "$f(t, w, v) \in \mathcal{F}_G^P(I)$?" for $P = P^S, P^M$, which have complexity O of problem Impl for the underlying logic; this shows membership in $\mathsf{P}_{\|}^{O}$. The hardness results for $O \neq \mathsf{PSpace}, \mathsf{Exp}$ are shown similarly as for problem Cons by a reduction from the Even problem. We use effect mappings P^M and P^S, which consist of P from there with additional rules (resp. implications for FOL+DCA) as follows: P^M contains $\phi'_j \leftarrow \phi_0, \ldots, \phi_{2j-1}$ and P^S contains $\phi'_j \leftarrow \phi_0, \ldots, \phi_{2j}$, for $0 \leq j \leq n$, where ϕ'_j is a new effect atom. Assuming that I_{2n+1} is a no-instance, the specifications will correspond, i.e., $\mathcal{F}_G^{P^M}(I) = \mathcal{F}_G^{P^S}(I)$, iff the number of yes-instances among $I_0, \ldots, I_{2n+1}$ is even.

Repair. To solve Repair, we can guess some change $I^+, I^- \subseteq I_G$ and then check whether $\mathcal{A}(I^+, I^-)$ holds and there are no conflicts for input $I' = (I \setminus I^-) \cup I^+$. For Impl complexity $O = \mathsf{Exp}, \mathsf{PSpace}$, this stays within O, as one can cycle through all I', and for the other complexity classes O, by the results for Cons, the problem is in $\mathsf{NP}^{\mathsf{P}_{\|}^{O}} = \mathsf{NP}^O$. The hardness results for these O can be obtained by reductions from a reasoning problem for ASP$^{\neg}$ resp. ASP$^{\vee,\neg}$ programs: given a set of facts F, a program P and a fact α, does there exist some $F' \subseteq F$ such that $P \cup F' \models \alpha$ (where $\models$ is cautious consequence); this problem is, as shown with slight extensions of the respective proofs for cautious consequence [4,5], complete for NP^O. Finally, the results for REPAIR remain unchanged if only strict repairs are considered as CORR has lower complexity.

6 Discussion and Conclusion

Since comprehensible specifications play a major role in this problem domain, we demand that the implementation should be *declarative*. An imperative way of programming such rules will quickly lead to deeply nested conditionals with intransparent dependencies. Further, we need a high degree of *modularity* to enable changes to specifications independent of the implementation of reasoning tasks. Since the domain comprises many patterns with exceptions and special cases, some sort of *default reasoning* would be desirable; e.g., traffic is permitted in a certain direction, *unless* it is explicitly prohibited.

Answer Set Programming (ASP) [9] is a natural choice for writing such declarative, executable specifications. Due to the availability of efficient solvers such as DLV [11] and Potassco [8], the ASP paradigm gains increasing popularity [2]. Furthermore, thanks to modularity properties of answer set semantics, it is easy to modularly compose programs P and Sp from a traffic regulation $\Pi = (P, Sp)$ into a single program $P \cup Sp$, provided that P has a unique answer set (as given, e.g., with stratified programs), where rules $\neg x \leftarrow \text{not } x$ for $\overline{X}$ are included (resp., if recursion through negation would emerge, rules $\neg x' \leftarrow \text{not } x$ and $x' \leftarrow x$, and x is replaced in Sp by x').

Implementation. In cooperation with domain experts, we have written prototypical ASP-programs to evaluate the consistency of scenarios, to diagnose conflicts and to test correspondence using partial realizations of the traffic regulation. In addition, we also dealt with repairs. We have used both the DLV system and Potassco to run these programs on a number of different scenarios, and obtained satisfactory initial results. Further development, regarding a representation of the traffic regulations and engineering the program for scalable execution, is ongoing; the fact that ASP programs serve as executable specification is very helpful for developing the representation.

The use of DLV and the flexible optimization constructs available in its language by weak constraints, has in fact also led us to experiment with preferred notions of diagnoses and repairs. The possibilities range from generic preferences, like favoring deletions over additions, to the encoding of very specific domain knowledge.

Example 10. Deleting the mandatory left turn at d_4 in Fig. 1b gives an alternative repair possibility for Example 2. The options to continue to drive straight over the junctions towards node d_7, as well as turning right towards node d_5, are not available due to the no-entry signs. According to the depicted street model, there is a way out from node d_4 via the U-turn to node d_3, from which an out-node node is reachable via nodes c_8 and c_3. Arguably, it is reasonable to still classify the situation as loop, since paths should not use U-turns. A road user arriving at d_4 for the first time would not know that she will eventually come back to node d_4 (by taking the supposed path).

Thus, one might have different categories of loop conflicts like "strong loop" and "weak loop," where the latter allows for escapes via U-turns. In this sense, Fig. 1b represents a strong loop, and the repair where the sign at d_4 is removed represents a weak loop. The repair might be less preferred. If yet chosen, we might wish to warn road users who eventually will have to take a U-turn, through a no-through-road sign. However, the ideal position to do so is not obvious. Alternatively, we could add mandatory U-turn signs before junctions one has to eventually return to. While this might often be the desired solution, it is formally not optimal, since it is more restrictive than necessary.

Summary and Outlook. To date, tools for advanced inconsistency management of traffic regulations are lacking. We presented a logic-based approach to this problem, which is highly relevant in future dynamic regulation settings. We formalized the notion of traffic regulation problem and defined major reasoning tasks on it, whose computational complexity we characterized for different logic languages. Complexity results on abduction from logic programs [6] are not applicable, as the language setting and problems studied there (in view of Prop. 1, consistent diagnosis existence with no effect mapping) are different. Our results provide a basis for implementation and show that some reasoning tasks may be hosted in the underlying logic language, while for others one needs a more expressive one. Finally, we briefly addressed an ASP-based prototype.

The implementation is part of an ongoing industrial project on management of traffic regulation data with *PRISMA solutions GmbH*.[3] Currently, real world traffic regulation data is administrated by software that shall be enhanced by the methods we described, and the resulting application shall be used by several Austrian regions, starting with Lower Austria and Vienna. On the theoretical side, an investigation of preference and properties of traffic regulation problems, under possible restrictions of the constituents, remains to be done. Also other logic formalisms, e.g., defeasible logic [1] (which like $\mathrm{ASP}^{\neg_s}$ allows for efficient reasoning [13]), or HEX-programs [7] may be considered.

References

1. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Trans. Comput. Logic 2(2), 255–287 (2001)
2. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Commun. ACM 54(12), 92–103 (2011)
3. Console, L., Torasso, P.: Automated diagnosis. Intelligenza Artificiale 3(1-2), 42–48 (2006)
4. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and Expressive Power of Logic Programming. ACM Comput. Surv. 33(3), 374–425 (2001)
5. Eiter, T., Faber, W., Fink, M., Woltran, S.: Complexity Results for Answer Set Programming with Bounded Predicate Arities. Ann. Math. Artif. Intell. 51(2-4), 123–165 (2007)
6. Eiter, T., Gottlob, G., Leone, N.: Abduction From Logic Programs: Semantics and Complexity. Theoretical Computer Science 189(1-2), 129–177 (1997)
7. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer set programming. In: IJCAI, pp. 90–96 (2005)
8. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., Schneider, M.T.: Potassco: The Potsdam answer set solving collection. AI Commun. 24(2), 107–124 (2011)
9. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. Next Generat. Comput. 9(3-4), 365–386 (1991)

[3] http://www.prisma-solutions.at

10. de Kleer, J., Kurien, J.: Fundamentals of model-based diagnosis. In: IFAC Symposium SAFEPROCESS 2003, pp. 25–36. Elsevier (2003)
11. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV System for Knowledge Representation and Reasoning. ACM TOCL 7(3), 499–562 (2006)
12. Lucas, P.: Symbolic diagnosis and its formalisation. Knowl. Eng. Rev. 12, 109–146 (1997)
13. Maher, M.J.: Propositional defeasible logic has linear complexity. Theory Pract. Log. Program. 1(6), 691–711 (2001)
14. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
15. Poole, D.: Normality and faults in logic-based diagnosis. In: IJCAI, pp. 1304–1310 (1989)
16. Poole, D.: Representing diagnosis knowledge. Ann. Math. Artif. Intell. 11, 33–50 (1994)

Conditional Epistemic Planning

Mikkel Birkegaard Andersen, Thomas Bolander, and Martin Holm Jensen

Technical University of Denmark

Abstract. Recent work has shown that Dynamic Epistemic Logic (DEL) offers a solid foundation for automated planning under partial observability and non-determinism. Under such circumstances, a plan must branch if it is to guarantee achieving the goal under all contingencies (strong planning). Without branching, plans can offer only the possibility of achieving the goal (weak planning). We show how to formulate planning in uncertain domains using DEL and give a language of conditional plans. Translating this language to standard DEL gives verification of both strong and weak plans via model checking. In addition to plan verification, we provide a tableau-inspired algorithm for synthesising plans, and show this algorithm to be terminating, sound and complete.

1 Introduction

Whenever an agent deliberates about the future with the purpose of achieving a goal, she is engaging in the act of planning. When planning, the agent has a view of the environment and knowledge of how her actions affect the environment. Automated Planning is a widely studied area of AI, in which problems are expressed along these lines. Many different variants of planning, with different assumptions and restrictions, have been studied. In this paper we consider planning under uncertainty (nondeterminism and partial observability), where exact states of affairs and outcomes of actions need not be known by the agent. We formulate such scenarios in an epistemic setting, where states, actions and goals are infused with the notions of knowledge from Dynamic Epistemic Logic (DEL). Throughout this exposition, our running example, starting with Example 1, follows the schemings of a thief wanting to steal a precious diamond.

Example 1. After following carefully laid plans, a thief has almost made it to her target: The vault containing the invaluable Pink Panther diamond. Standing outside the vault ($\neg v$), she now deliberates on how to get her hands on the diamond (d). She knows the light inside the vault is off ($\neg l$), and that the Pink Panther is on either the right (r) or left ($\neg r$) pedestal inside. Obviously, the diamond cannot be on both the right *and* left pedestal, but nonetheless the agent may be uncertain about its location. This scenario is represented by the epistemic model in Figure 1. The edge between w_1 and w_2 signifies that these worlds are indistinguishable to the agent. For visual clarity we omit reflexive edges (each world is always reachable from itself). We indicate with a string the valuation at world w, where an underlined proposition p signifies that p does *not* hold at w.

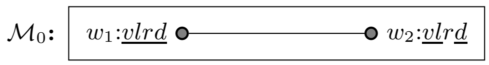


Fig. 1. The initial situation. The thief is uncertain about whether r holds.

The agent's goal is to obtain the jewel and to be outside the vault. She can enter and leave the vault, flick the light switch and snatch the contents of either the right or left pedestal. Her aim is to come up with a, possibly conditional, plan, such that she achieves her goal.

By applying DEL to scenarios such as the above, we can construct a procedure for the line of reasoning that is of interest to the thief. In the following section we recap the version of DEL relevant to our purposes. Section 3 formalises notions from planning in DEL, allowing verification of plans (using model checking) as either weak or strong solutions. In Section 4 we introduce an algorithm for plan synthesis (i.e. generation of plans). Further we show that the algorithm is terminating, sound and complete.

2 Dynamic Epistemic Logic

Dynamic epistemic logics describe knowledge and how actions change it. These changes may be epistemic (changing knowledge), ontic (changing facts) or both. The work in this paper deals only with the single-agent setting, though we briefly discuss the multi-agent setting in Section 5. As in Example 1, agent knowledge is captured by epistemic models. Changes are encoded using event models (defined below). The following concise summary of DEL is meant as a reference for the already familiar reader. The unfamiliar reader may consult [12,13] for a thorough treatment.

Definition 1 (Epistemic Language). *Let a set of propositional symbols P be given. The language $\mathcal{L}_{\text{DEL}}(P)$ is given by the following BNF:*

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid K\phi \mid [\mathcal{E}, e]\,\phi$$

where $p \in P$, $\mathcal{E}$ denotes an event model *on $\mathcal{L}_{\text{DEL}}(P)$ as (simultaneously) defined below, and $e \in D(\mathcal{E})$. K is the epistemic modality and $[\mathcal{E}, e]$ the dynamic modality. We use the usual abbreviations for the other boolean connectives, as well as for the dual dynamic modality $\langle\mathcal{E}, e\rangle\,\phi := \neg\,[\mathcal{E}, e]\,\neg\phi$. The dual of K is denoted $\tilde{K}$. $K\phi$ reads as "the (planning) agent knows ϕ" and $[\mathcal{E}, e]\,\phi$ as "after all possible executions of $(\mathcal{E}, e)$, ϕ holds".*

Definition 2 (Epistemic Models). *An* epistemic model *on $\mathcal{L}_{\text{DEL}}(P)$ is a tuple $\mathcal{M} = (W, \sim, V)$, where W is a set of* worlds*, $\sim$ is an equivalence relation (the* epistemic relation*) on W, and $V : P \to 2^W$ is a* valuation*. $D(\mathcal{M}) = W$ denotes the* domain *of $\mathcal{M}$. For $w \in W$ we name $(\mathcal{M}, w)$ a* pointed epistemic model*, and refer to w as the* actual world *of $(\mathcal{M}, w)$.*

To reason about the dynamics of a changing system, we make use of *event models*. The formulation of event models we use in this paper is due to van Ditmarsch and Kooi [12]. It adds ontic change to the original formulation of [5] by adding postconditions to events.

Definition 3 (Event Models). *An* event model *on* $\mathcal{L}_{\text{DEL}}(P)$ *is a tuple* $\mathcal{E} = (E, \sim, pre, post)$*, where*

- E *is a set of* (basic) events*,*
- $\sim \subseteq E \times E$ *is an equivalence relation called the* epistemic relation*,*
- $pre : E \to \mathcal{L}_{\text{DEL}}(P)$ *assigns to each event a* precondition*,*
- $post : E \to (P \to \mathcal{L}_{\text{DEL}}(P))$ *assigns to each event a* postcondition*.*

$D(\mathcal{E}) = E$ *denotes the* domain *of* $\mathcal{E}$*. For* $e \in E$ *we name* $(\mathcal{E}, e)$ *a* pointed event model*, and refer to* e *as the* actual event *of* $(\mathcal{E}, e)$*.*

Definition 4 (Product Update). *Let* $\mathcal{M} = (W, \sim, V)$ *and* $\mathcal{E} = (E, \sim', pre, post)$ *be an epistemic model resp. event model on* $\mathcal{L}_{\text{DEL}}(P)$*. The* product update *of* $\mathcal{M}$ *with* $\mathcal{E}$ *is the epistemic model denoted* $\mathcal{M} \otimes \mathcal{E} = (W', \sim'', V')$*, where*

- $W' = \{(w, e) \in W \times E \mid \mathcal{M}, w \models pre(e)\}$*,*
- $\sim'' = \{((w, e), (v, f)) \in W' \times W' \mid w \sim v \text{ and } e \sim' f\}$*,*
- $V'(p) = \{(w, e) \in W' \mid \mathcal{M}, w \models post(e)(p)\}$ *for each* $p \in P$*.*

Definition 5 (Satisfaction Relation). *Let a pointed epistemic model* $(\mathcal{M}, w)$ *on* $\mathcal{L}_{\text{DEL}}(P)$ *be given. The satisfaction relation is given by the usual semantics, where we only recall the definition of the dynamic modality:*

$$\mathcal{M}, w \models [\mathcal{E}, e]\,\phi \qquad \textit{iff } \mathcal{M}, w \models pre(e) \textit{ implies } \mathcal{M} \otimes \mathcal{E}, (w, e) \models \phi$$

where $\phi \in \mathcal{L}_{\text{DEL}}(P)$ *and* $(\mathcal{E}, e)$ *is a pointed event model. We write* $\mathcal{M} \models \phi$ *to mean* $\mathcal{M}, w \models \phi$ *for all* $w \in D(\mathcal{M})$*. Satisfaction of the dynamic modality for non-pointed event models* $\mathcal{E}$ *is introduced by abbreviation, viz.* $[\mathcal{E}]\,\phi := \bigwedge_{e \in \mathcal{D}(\mathcal{E})} [\mathcal{E}, e]\,\phi$*.*

Throughout the rest of this paper, all languages (sets of propositional symbols) and all models (sets of possible worlds) considered are implicitly assumed to be finite.

3 Conditional Plans in DEL

One way to sum up automated planning is that it deals with the *reasoning side of acting* [14]. When planning under uncertainty, actions can be nondeterministic and the states of affairs partially observable. In the following, we present a formalism expressing planning under uncertainty in DEL, while staying true to the notions of automated planning. We consider a system similar to that of [14, sect. 17.4], which motivates the following exposition. The type of planning detailed here is *offline*, where planning is done before acting. All reasoning must therefore be based on the agent's initial knowledge.

3.1 States and Actions: The Internal Perspective

Automated planning is concerned with achieving a certain goal state from a given initial state through some combination of available actions. In our case, states are epistemic models. These models represent situations from the perspective of the

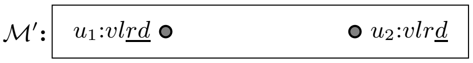


Fig. 2. A model consisting of two information cells

planning agent. We call this the *internal perspective*—the modeller is modelling itself. The internal perspective is discussed thoroughly in [2,11].

Generally, an agent using epistemic models to model its own knowledge and ignorance, will not be able to point out the actual world. Consider the epistemic model $\mathcal{M}_0$ in Figure 1, containing two indistinguishable worlds w_1 and w_2. Regarding this model to be the planning agent's own representation of the initial state of affairs, the agent is of course not able to point out the actual world. It is thus natural to represent this situation as a non-pointed epistemic model. In general, when the planning agent wants to model a future (imagined) state of affairs, she does so by a non-pointed model.

The equivalence classes (wrt. $\sim$) of a non-pointed epistemic model are called the *information cells* of that model (in line with the corresponding concept in [6]). We also use the expression *information cell* on $\mathcal{L}_{\mathrm{DEL}}(P)$ to denote any connected epistemic model on $\mathcal{L}_{\mathrm{DEL}}(P)$, that is, any epistemic model consisting of a single information cell. All worlds in an information cell satisfy the same K-formulas (formulas of the form $K\phi$), thus representing the same situation as seen from the agent's internal perspective. Each information cell of a (non-pointed) epistemic model represents a possible state of knowledge of the agent.

Example 2. Recall that our jewel thief is at the planning stage, with her initial information cell $\mathcal{M}_0$. She realises that entering the vault and turning on the light will reveal the location of the Pink Panther. Before actually performing these actions, she can rightly reason that they will lead her to know the location of the diamond, though whether that location is left or right cannot be determined (yet).

Her representation of the possible outcomes of going into the vault and turning on the light is the model $\mathcal{M}'$ in Figure 2. The information cells $\mathcal{M}' \restriction \{u_1\}$ and $\mathcal{M}' \restriction \{u_2\}$ of $\mathcal{M}'$ are exactly the two distinguishable states of knowledge the jewel thief considers possible prior turning the light on in the vault.

In the DEL framework, actions are naturally represented as event models. Due to the internal perspective, these are also taken to be non-pointed. For instance, in a coin toss action, the agent cannot beforehand point out which side will land face up.

Example 3. Continuing Example 2 we now formalize the actions available to our thieving agent as the event models in Figure 3. We use the same conventions for edges as we did for epistemic models. For a basic event e we label it $\langle pre(e), post(e)\rangle$.[1]

The agent is endowed with four actions: $\mathsf{take\_left}$, resp. $\mathsf{take\_right}$, represent trying to take the diamond from the left, resp. right, pedestal; the diamond is obtained only if it is on the chosen pedestal. Both actions require the agent to

[1] For a proposition p whose truth value does not change in e we assume the identity mapping $post(e)(p) = p$, as is also the convention in automated planning.

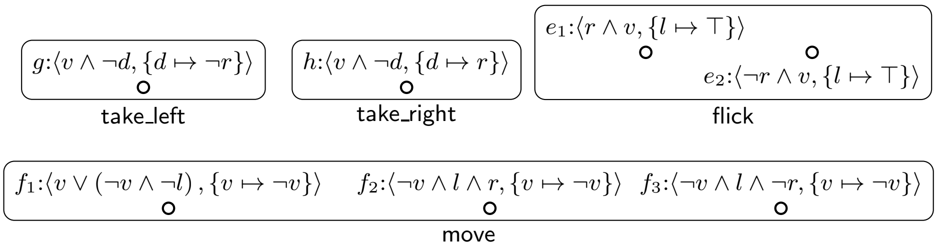

g:$\langle v \wedge \neg d, \{d \mapsto \neg r\}\rangle$
take_left
h:$\langle v \wedge \neg d, \{d \mapsto r\}\rangle$
take_right
e_1:$\langle r \wedge v, \{l \mapsto \top\}\rangle$
e_2:$\langle \neg r \wedge v, \{l \mapsto \top\}\rangle$
flick
f_1:$\langle v \vee (\neg v \wedge \neg l), \{v \mapsto \neg v\}\rangle$
f_2:$\langle \neg v \wedge l \wedge r, \{v \mapsto \neg v\}\rangle$
f_3:$\langle \neg v \wedge l \wedge \neg r, \{v \mapsto \neg v\}\rangle$
move

Fig. 3. Event models representing the actions of the thief

be inside the vault and not holding the diamond. flick requires the agent to be inside the vault and turns the light on. Further, it reveals which pedestal the diamond is on. move represents the agent moving in or out of the vault, revealing the location of the diamond provided the light is on.

It can be seen that the epistemic model $\mathcal{M}'$ in Example 2 is the result of two successive product updates, namely $\mathcal{M}_0 \otimes \mathsf{move} \otimes \mathsf{flick}$.

3.2 Applicability, Plans and Solutions

Reasoning about actions from the initial state as in Example 3 is exactly what planning is all about. We have however omitted an important component in the reasoning process, one which is crucial. The notion of *applicability* in automated planning dictates when the outcomes of an action are defined. The idea translates to DEL by insisting that no world the planning agent considers possible is eliminated by the product update of an epistemic model with an event model.

Definition 6 (Applicability). *An event model $\mathcal{E}$ is said to be* applicable *in an epistemic model $\mathcal{M}$ if $\mathcal{M} \models \langle \mathcal{E} \rangle \top$.*

This concept of applicability is easily shown to be equivalent with the one defined in [11] when restricting the latter to the single-agent case. However, for our purposes of describing plans as formulas, we need to express applicability as formulas as well. The discussion in [16, sect. 6.6] also notes this aspect, insisting that actions must be meaningful. The same sentiment is expressed by our notion of applicability.

The situation in Example 2 calls for a way to express conditional plans. Clearly, our agent can only snatch the jewel from the correct pedestal conditioned on how events unfold when she acts. To this end we introduce a language for conditional plans allowing us to handle such contingencies.

Definition 7 (Plan Language). *Given a finite set* A *of event models on $\mathcal{L}_{\mathrm{DEL}}(P)$, the* plan language $\mathcal{L}_{\mathrm{P}}(P, \mathsf{A})$ *is given by:*

$$\pi ::= \mathcal{E} \mid \mathsf{skip} \mid \mathsf{if}\ K\phi\ \mathsf{then}\ \pi\ \mathsf{else}\ \pi \mid \pi ; \pi$$

where $\mathcal{E} \in \mathsf{A}$ and $\phi \in \mathcal{L}_{\mathrm{DEL}}(P)$. We name members π of this language plans*, and use* if $K\phi$ then π *as shorthand for* if $K\phi$ then π else skip.

The reading of the plan constructs are "do $\mathcal{E}$", "do nothing", "if $K\phi$ then π, else π'", and "first π then π'" respectively. Note that the condition of the $\mathsf{if\text{-}then\text{-}else}$ construct is required to be a K-formula. This is to ensure that the planning agent can only make her choices of actions depend on worlds that are distinguishable to her (cf. the discussion of the internal perspective in Section 3.1). The idea is similar to the *meaningful plans* of [16], where branching is only allowed on *epistemically interpretable formulas.*

An alternative way of specifying conditional plans is *policies*, where (in our terminology) each information cell maps to an event model [14, Sect. 16.2]. There are slight differences between the expressiveness of conditional plans and policies (e.g. policies can finitely represent repetitions); our main motivation for not using policies is that it would require an enumeration of each information cell of the planning domain.

Definition 8 (Translation). *We define a* strong translation $[\![\cdot]\!]_s \cdot$ *and a* weak translation $[\![\cdot]\!]_w \cdot$ *as functions from* $\mathcal{L}_\mathrm{P}(P, \mathsf{A}) \times \mathcal{L}_\mathrm{DEL}(P)$ *into* $\mathcal{L}_\mathrm{DEL}(P)$ *by:*

$$\begin{aligned}
&[\![\mathcal{E}]\!]_s\, \phi := \langle\mathcal{E}\rangle \top \wedge [\mathcal{E}]\, K\phi \\
&[\![\mathcal{E}]\!]_w\, \phi := \langle\mathcal{E}\rangle \top \wedge \langle\mathcal{E}\rangle\, K\phi \\
&[\![\mathsf{skip}]\!]_\cdot\, \phi := \phi \\
&[\![\mathsf{if}\ \phi'\ \mathsf{then}\ \pi\ \mathsf{else}\ \pi']\!]_\cdot\, \phi := (\phi' \to [\![\pi]\!]_\cdot\, \phi) \wedge (\neg\phi' \to [\![\pi']\!]_\cdot\, \phi) \\
&[\![\pi;\pi']\!]_\cdot\, \phi := [\![\pi]\!]_\cdot\, ([\![\pi']\!]_\cdot\, \phi)
\end{aligned}$$

Plans describe the manner in which actions are carried out. We interpret plans π relative to a formula ϕ and want to answer the question of whether or not π achieves ϕ. Using Definition 8 we can answer this question by verifying truth of the DEL formula provided by the translations. This is supported by the results of Section 4. We concisely read $[\![\pi]\!]_s\, \phi$ as "π achieves ϕ", and $[\![\pi]\!]_w\, \phi$ as "π may achieve ϕ" (elaborated below). By not specifying separate semantics for plans our framework is kept as simple as possible. Note that applicability (Definition 6) is built into the translations through the occurrence of the conjunct $\langle\mathcal{E}\rangle \top$ in both the strong translation $[\![\mathcal{E}]\!]_s\, \phi$ and the weak translation $[\![\mathcal{E}]\!]_w\, \phi$.

The difference between the two translations relate to the *robustness* of plans: $[\![\pi]\!]_s\, \phi$, resp. $[\![\pi]\!]_w\, \phi$, means that every step of π is applicable and that following π always leads, resp. may lead, to a situation where ϕ is known.

Definition 9 (Planning Problems and Solutions). *Let P be a finite set of propositional symbols. A planning problem on P is a triple $\mathcal{P} = (\mathcal{M}_0, \mathsf{A}, \phi_g)$ where*
- $\mathcal{M}_0$ *is an information cell on* $\mathcal{L}_\mathrm{DEL}(P)$ *called the* initial state*.*
- A *is a finite set of event models on* $\mathcal{L}_\mathrm{DEL}(P)$ *called the* action library*.*
- $\phi_g \in \mathcal{L}_\mathrm{DEL}(P)$ *is the* goal (formula)*.*

We say that a plan $\pi \in \mathcal{L}_\mathrm{P}(P, \mathsf{A})$ *is a* strong solution *to* $\mathcal{P}$ *if* $\mathcal{M}_0 \models [\![\pi]\!]_s\, \phi_g$*, a* weak solution *if* $\mathcal{M}_0 \models [\![\pi]\!]_w\, \phi_g$ *and not a solution otherwise.*

Planning problems are defined with the sentiment we've propagated in our examples up until now. The agent is presently in $\mathcal{M}_0$ and wishes ϕ_g to be the case.

To this end, she reasons about the actions (event models) in her action library A, creating a conditional plan. Using model checking, she can verify whether this plan is either a weak or strong solution, since plans translate into formulas of $\mathcal{L}_{\mathrm{DEL}}(P)$. Further, [12] gives reduction axioms for DEL-formulas, showing that any formula containing the dynamic modality can be expressed as a formula in (basic) epistemic logic. Consequently, plan verification can be seen simply as epistemic reasoning about $\mathcal{M}_0$.

Example 4. We continue our running example by discussing it formally as a planning problem and considering the solutions it allows. The initial state is still $\mathcal{M}_0$, and the action library $\mathsf{A} = \{\mathsf{flick}, \mathsf{move}, \mathsf{take\_left}, \mathsf{take\_right}\}$. We discuss the plans below and their merit for our thief.

- $\pi_1 = \mathsf{flick}; \mathsf{move}; \mathsf{if}\ Kr\ \mathsf{then}\ \mathsf{take\_right}\ \mathsf{else}\ \mathsf{take\_left}; \mathsf{move}$
- $\pi_2 = \mathsf{move}; \mathsf{take\_right}; \mathsf{move}$
- $\pi_3 = \mathsf{move}; \mathsf{flick}; \mathsf{take\_right}; \mathsf{move}$
- $\pi_4 = \mathsf{move}; \mathsf{flick}; \mathsf{if}\ Kr\ \mathsf{then}\ \mathsf{take\_right}\ \mathsf{else}\ \mathsf{take\_left}; \mathsf{move}$

We consider two planning problems varying only on the goal formula, $\mathcal{P}_1 = (\mathcal{M}_0, \mathsf{A}, d \wedge \neg v)$ and $\mathcal{P}_2 = (\mathcal{M}_0, \mathsf{A}, \tilde{K}d \wedge \neg v)$. In $\mathcal{P}_1$ her goal is to obtain the diamond and be outside the vault, whereas in $\mathcal{P}_2$ she wishes to be outside the vault *possibly* having obtained the diamond.

Let $\pi_1' = \mathsf{move}; \mathsf{if}\ Kr\ \mathsf{then}\ \mathsf{take\_right}\ \mathsf{else}\ \mathsf{take\_left}; \mathsf{move}$ and note that $\pi_1 = \mathsf{flick}; \pi_1'$. Using the strong translation of π_1, we get $\mathcal{M}_0 \models [\![\pi_1]\!]_s\, \phi_g$ iff $\mathcal{M}_0 \models \langle \mathsf{flick} \rangle \top \wedge [\mathsf{flick}]\, [\![\pi_1']\!]_s\, \phi_g$. As $\mathcal{M}_0 \models \langle \mathsf{flick} \rangle \top$ does not hold, π_1 is not a solution. This is expected, since flicking the switch in the initial state is not an applicable action. Verifying that π_2 is a strong solution to $\mathcal{P}_2$ amounts to checking if $\mathcal{M}_0 \models [\![\pi_2]\!]_s\, \tilde{K}d \wedge \neg v$ which translates to

$$\mathcal{M}_0 \models \langle \mathsf{move} \rangle \top \wedge [\mathsf{move}] \Big(\langle \mathsf{take\_right} \rangle \top \wedge [\mathsf{take\_right}] \Big(\langle \mathsf{move} \rangle \top \wedge [\mathsf{move}] \Big(\tilde{K}d \wedge \neg v \Big) \Big) \Big)$$

With the same approach we can conclude that π_2 is not a solution to $\mathcal{P}_1$, π_3 is a weak solution to $\mathcal{P}_1$ and $\mathcal{P}_2$, and π_4 is a strong solution to $\mathcal{P}_1$ and $\mathcal{P}_2$.

4 Plan Synthesis

We now show how to synthesise conditional plans for solving planning problems. To synthesise plans, we need a mechanism for coming up with formulas characterising information cells for if-then-else constructs to branch on. Inspired by [8,9], these are developed in the following. Proofs are omitted, as they are straightforward and similar to proofs in the aforementioned references.

Definition 10 (Characterising Formulas). *Let $\mathcal{M} = (W, \sim, V)$ denote an information cell on $\mathcal{L}_{\mathrm{DEL}}(P)$. We define for all $w \in W$ a formula ϕ_w by: $\phi_w = \bigwedge_{p \in V(w)} p \wedge \bigwedge_{p \in P - V(w)} \neg p$. We define the* characterising formula *for $\mathcal{M}$, $\delta_{\mathcal{M}}$, as follows: $\delta_{\mathcal{M}} = K(\bigwedge_{w \in W} \tilde{K}\phi_w \wedge K \bigvee_{w \in W} \phi_w)$.*

Lemma 1. *Let $\mathcal{M}$ be an information cell on $\mathcal{L}_{\text{DEL}}(P)$. Then for all epistemic models $\mathcal{M}' = (W', \sim', V')$ and all $w' \in W'$ we have that $(\mathcal{M}', w') \models \delta_{\mathcal{M}}$ if and only if there exists a $w \in \mathcal{D}(\mathcal{M})$ such that $(\mathcal{M}, w) \leftrightarroweq (\mathcal{M}', w')$.*[2]

4.1 Planning Trees

For brevity, proofs have been omitted from this section. We encourage the reader to consult the long version [1] in which full proofs are given.

When synthesising plans, we explicitly construct the search space of the problem as a labelled AND-OR tree, a familiar model for planning under uncertainty [14]. Our AND-OR trees are called *planning trees.*

Definition 11. *A* planning tree *is a finite, labelled* AND-OR *tree in which each node n is labelled by an epistemic model $\mathcal{M}(n)$, and each edge (n, m) leaving an* OR*-node is labelled by an event model $\mathcal{E}(n, m)$.*

Planning trees for planning problems $\mathcal{P} = (\mathcal{M}_0, \mathsf{A}, \phi_g)$ are constructed as follows. Let the initial planning tree T_0 consist of just one OR-node $root(T_0)$ with $\mathcal{M}(root(T_0)) = \mathcal{M}_0$ (the root labels the initial state). A planning tree for $\mathcal{P}$ is then any tree that can be constructed from T_0 by repeated applications of the following non-deterministic tree expansion rule.

Definition 12 (Tree Expansion Rule). *Let T be a planning tree for a planning problem $\mathcal{P} = (\mathcal{M}_0, \mathsf{A}, \phi_g)$. The tree expansion rule is defined as follows. Pick an* OR*-node n in T and an event model $\mathcal{E} \in \mathsf{A}$ applicable in $\mathcal{M}(n)$ with the proviso that $\mathcal{E}$ does not label any existing outgoing edges from n. Then:*

1. *Add a new node m to T with $\mathcal{M}(m) = \mathcal{M}(n) \otimes \mathcal{E}$, and add an edge (n, m) with $\mathcal{E}(n, m) = \mathcal{E}$.*
2. *For each information cell $\mathcal{M}'$ in $\mathcal{M}(m)$, add an* OR*-node m' with $\mathcal{M}(m') = \mathcal{M}'$ and add the edge (m, m').*

The tree expansion rule is similar in structure to—and inspired by—the expansion rules used in tableau calculi, e.g. for modal and description logics [15]. Note that the expansion rule applies only to OR-nodes, and that an applicable event model can only be used once at each node.

Considering single-agent planning a two-player game, a useful analogy for planning trees are game trees. At an OR-node n, the agent gets to pick any applicable action $\mathcal{E}$ it pleases, winning if it ever reaches an epistemic model in which the goal formula holds (see the definition of solved nodes further below). At an AND-node m, the environment responds by picking one of the information cells of $\mathcal{M}(m)$—which of the distinguishable outcomes is realised when performing the action.

Example 5. In Fig. 4 is a planning tree for a variant of the Pink Panther planning problem, this one where the thief is already inside the vault. The root is n_0. Three applications of the tree expansion rule have been made, the labels on edges indicating the chosen action. n_0, n_l and n_r are OR-nodes. n_0', n_l' and n_r' are AND-nodes. The

[2] Here $(\mathcal{M}, w) \leftrightarroweq (\mathcal{M}', w)$ denotes that $(\mathcal{M}, w)$ and $(\mathcal{M}', w)$ are bisimilar according to the standard notion of bisimulation on pointed epistemic models.

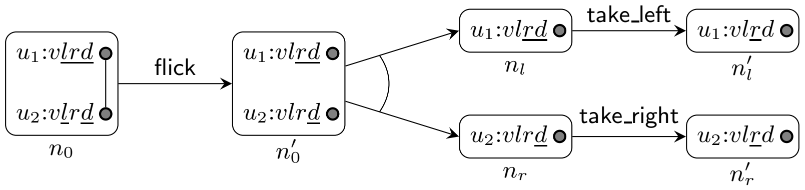


Fig. 4. Planning tree for a variant of the Pink Panther problem

child nodes of the latter two AND-nodes have been omitted, as their information cell is the same as that of their parent nodes. Pay particular attention to how flick reveals the location of the diamond. In the initial state, $\mathcal{M}(n_0) \models \neg Kr \wedge \neg K\neg r$, while $\mathcal{M}(n_0') \models Kr \vee K\neg r$, $\mathcal{M}(n_l) \models K\neg r$ and $\mathcal{M}(n_r) \models Kr$.

Without restrictions on the tree expansion rule, even very simple planning problems might be infinitely expanded. Finiteness of trees (and therefore termination) is ensured by the following blocking condition.

$\mathcal{B}_1$ The tree expansion rule may not be applied to a node n for which there exists an ancestor node m with $\mathcal{M}(m) \leftrightarrows \mathcal{M}(n)$.[3]

A planning tree for a planning problem $\mathcal{P}$ is called $\mathcal{B}_1$-*saturated* if no more expansions are possible satisfying condition $\mathcal{B}_1$.

Lemma 2 (Termination). *Any procedure that builds a $\mathcal{B}_1$-saturated planning tree for a planning problem $\mathcal{P}$ by repeated application of the tree expansion rule terminates.*

Definition 13 (Solved Nodes). *Let T be any (not necessarily saturated) planning tree for a planning problem $\mathcal{P} = (\mathcal{M}_0, \mathsf{A}, \phi_g)$. By recursive definition, a node n in T is called* solved *if one of the following holds:*
- *$\mathcal{M}(n) \models \phi_g$ (the node satisfies the goal formula).*
- *n is an* OR-*node having at least one solved child.*
- *n is an* AND-*node having all its children solved.*

Continuing the game tree analogy, we see that a solved node corresponds is one for which there exists a winning strategy. Regardless of the environment's choice, the agent can achieve its goal. Let T and $\mathcal{P}$ be as above. Below we show that when a node n is solved, it is possible to construct a (strong) solution to the planning problem $(\mathcal{M}(n), \mathsf{A}, \phi_g)$. In particular, if the root node is solved, a strong solution to $\mathcal{P}$ can be constructed. As it is never necessary to expand a solved node, nor any of its descendants, we can augment the blocking condition $\mathcal{B}_1$ in the following way.

$\mathcal{B}_2$ The tree expansion rule may not be applied to a node n if one of the following holds: 1) n is solved; 2) n has a solved ancestor; 3) n has an ancestor node m with $\mathcal{M}(m) \leftrightarrows \mathcal{M}(n)$.

[3] Here $\mathcal{M}(m) \leftrightarrows \mathcal{M}(n)$ denotes that $\mathcal{M}(m)$ and $\mathcal{M}(n)$ are bisimilar according to the standard notion of bisimulation between non-pointed epistemic models.

In the following, we will assume that all planning trees have been built according to $\mathcal{B}_2$. One consequence is that a solved OR-node has exactly one solved child. We make use of this in the following definition.

Definition 14 (Plans for Solved Nodes). *Let T be any planning tree for $\mathcal{P} = (\mathcal{M}_0, \mathsf{A}, \phi_g)$. For each solved node n in T, a plan $\pi(n)$ is defined recursively by:*

- *if $\mathcal{M}(n) \models \phi_g$, then $\pi(n) = \mathsf{skip}$.*
- *if n is an* OR*-node and m its solved child, then $\pi(n) = \mathcal{E}(n, m); \pi(m)$.*
- *if n is an* AND*-node with children $m_1, \ldots, m_k$, then $\pi(n) =$*
  $\mathsf{if}\ \delta_{\mathcal{M}(m_1)}\ \mathsf{then}\ \pi(m_1)\ \mathsf{else\ if}\ \delta_{\mathcal{M}(m_2)}\ \mathsf{then}\ \pi(m_2)\ \mathsf{else}\ \cdots\ \mathsf{if}\ \delta_{\mathcal{M}(m_k)}\ \mathsf{then}\ \pi(m_k)$

Example 6. For the goal of achieving the diamond, $\phi_g = d$, we have that the root n_0 of the planning tree of Figure 4 is solved, as both n'_l and n'_r satisfy the goal formula. Definition 14 gives us $\pi(n_0) = \mathsf{flick}; \mathsf{if}\ \delta_{\mathcal{M}(n_l)}\ \mathsf{then}\ \mathsf{take\_left}; \mathsf{skip}\ \mathsf{else\ if}\ \delta_{\mathcal{M}(n_r)}\ \mathsf{then}\ \mathsf{take\_right}; \mathsf{skip}$. This plan can easily be shown to be a strong solution to the planning problem of achieving d from the initial state $\mathcal{M}(n_0)$. In our soundness result below, we show that plans of solved roots are always strong solutions to their corresponding planing problems.

Theorem 1 (Soundness). *Let T be a planning tree for a problem $\mathcal{P}$ such that $root(T)$ is solved. Then $\pi(root(T))$ is a strong solution to $\mathcal{P}$.*

In addition to soundness, we also have completeness.

Theorem 2 (Completeness). *If there is a strong solution to the planning problem $\mathcal{P} = (\mathcal{M}_0, \mathsf{A}, \phi_g)$, then a planning tree T for $\mathcal{P}$ can be constructed, such that $root(T)$ is solved.*

4.2 Strong Planning Algorithm

With all the previous in place, we now have an algorithm for synthesising strong solutions for planning problems $\mathcal{P}$, given as follows.

```
StrongPlan($\mathcal{P}$)
1  Let $T$ be the plan. tree only consisting of $root(T)$ labelled by the init. state of $\mathcal{P}$.
2  Repeatedly apply the tree expansion rule of $\mathcal{P}$ to $T$ until it is $\mathcal{B}_2$-saturated.
3  If $root(T)$ is solved, return $\pi(root(T))$, otherwise return FAIL.
```

Theorem 3. StrongPlan($\mathcal{P}$) *is a terminating, sound and complete algorithm for producing strong solutions to planning problems. Soundness means that if* StrongPlan($\mathcal{P}$) *returns a plan, it is a strong solution to $\mathcal{P}$. Completeness means that if $\mathcal{P}$ has a strong solution,* StrongPlan($\mathcal{P}$) *will return one.*

Proof. Termination comes from Lemma 2 (with $\mathcal{B}_1$ replaced by the stronger condition $\mathcal{B}_2$), soundness from Theorem 1 and completeness from Theorem 2 (given any two saturated planning trees T_1 and T_2 for the same planning problem, the root node of T_1 is solved iff the root node of T_2 is).

4.3 Weak Planning Algorithm

With few changes, the machinery already in place gives an algorithm for synthesising weak solutions. Rather than requiring all children of an AND-node be solved, we require only one. This corresponds to the notion of weak, defined in Definition 8. Only one possible execution need lead to the goal.

Definition 15 (Weakly Solved Nodes). *A node n is called* weakly solved *if either $\mathcal{M}(n) \models \phi_g$ or n has at least one solved child.*

We keep the tree expansion rule, but make use of a new blocking condition $\mathcal{B}_3$ using Definition 15 rather than Definition 13.

Definition 16 (Plans for Weakly Solved Nodes). *Let T be any planning tree for $\mathcal{P} = (\mathcal{M}_0, \mathsf{A}, \phi_g)$. For each weakly solved node n in T, a plan $\pi_w(n)$ is defined recursively by:*

- *if $\mathcal{M}(n) \models \phi_g$, then $\pi_w(n) = \mathsf{skip}$*
- *if n is an OR-node and m its weakly solved child, then $\pi_w(n) = \mathcal{E}(n,m); \pi_w(m)$*
- *if n is an AND-node and m its weakly solved child, then $\pi_w(n) = \pi_w(m)$*

The algorithm for weak planning is defined as follows.

```
WeakPlan($\mathcal{P}$)
1  Let $T$ be the plan. tree only consisting of $root(T)$ labelled by the init. state of $\mathcal{P}$.
2  Repeatedly apply the tree expansion rule of $\mathcal{P}$ to $T$ until it is $\mathcal{B}_3$-saturated.
3  If $root(T)$ is weakly solved, return $\pi_w(root(T))$, otherwise return FAIL.
```

Theorem 4. WeakPlan($\mathcal{P}$) *is a terminating, sound and complete algorithm for producing weak solutions to planning problems.*

5 Related and Future Work

In this paper, we have presented a syntactic characterisation of weak and strong solutions to epistemic planning problems, that is, we have characterised solutions as formulas. [11] takes a semantic approach to strong solutions for epistemic planning problems. In their work plans are sequences of actions, requiring conditional choice of actions at different states to be encoded in the action structure itself. We represent choice explicitly, using a language of conditional plans. An alternative to our approach of translating conditional plans into formulas of DEL would be to translate plans directly into (complex) event models. This is the approach taken in [4], where they have a language of epistemic programs similar to our language of plans (modulo the omission of ontic actions). Using this approach in a planning setting, one could translate each possible plan π into the corresponding event model $\mathcal{E}(\pi)$, check its applicability, and check whether $\mathcal{M}_0 \otimes \mathcal{E}(\pi) \models \phi_g$ (the goal is satisfied in the product update of the initial state with the event model). However, even for a finite action library, there are infinitely many distinct plans, and thus infinitely many induced event models to consider when searching for a solution. To construct a terminating planning algorithm with this approach, one would still have to limit the plans considered (e.g. by using characterising formulas), and also develop a more involved loop-checking mechanism working at the level of plans. Furthermore, our approach more obviously generalises to algorithms for replanning, which is current work.

The meaningful plans of [16, chap. 2] are reminiscent of the work in this paper. Therein, plan verification is cast as validity of an EDL-consequence in a given system description. Like us, they consider single-agent scenarios, conditional plans, applicability and incomplete knowledge in the initial state. Unlike us, they consider only deterministic actions. In the multi-agent treatment [16, chap. 4], action laws are translated to a fragment of DEL with only public announcements and public assignments, making actions singleton event models. This means foregoing nondeterminism and therefore sensing actions.

Planning problems in [17] are solved by producing a sequence of pointed event models where an external variant of applicability (called *possible at*) is used. Using such a formulation means outcomes of actions are fully determined, making conditional plans and weak solutions superfluous. As noted by the authors, and unlike our framework, their approach does not consider factual change. We stress that [11,17,16] all consider the multi-agent setting which we have not treated here.

In our work so far, we haven't treated the problem of where domain formulations come from, assuming just that they are given. Standardised description languages are vital if modal logic-based planning is to gain wide acceptance in the planning community. Recent work worth noting in this area includes [7], which presents a specification language for the multi-agent belief case.

As suggested by our construction of planning trees, there are several connections between our approach and two-player imperfect information games. First, product updates imply perfect recall [10]. Second, when the game is at a node belonging to an information set, the agent knows a proposition only if it holds throughout the information set; corresponding to our use of information cells. Finally, the strong solutions we synthesise are very similar to mixed strategies. A strong solution caters to any information cell (contingency) it may bring about, by selecting exactly one sub-plan for each [3].

Our work naturally relates to [14], where the notions of strong and weak solutions are found. Their belief states are sets of states which may be partioned by observation variables. Our partition of epistemic models into information cells follows straight from the definition of product update. A clear advantage in our approach is that actions encode both nondetermism and partial observability. [18] shows that for conditional planning (prompted by nondeterministic actions) in partially observable domains the *plan existence problem* is 2-EXP-complete (plans must succeed with probability 1; i.e. be strong solutions). StrongPlan($\mathcal{P}$) implicitly answers the same question for $\mathcal{P}$ (it gives a strong solution if one exists). Reductions between the two decision problem variants would give a complexity measure of our approach, and also formally link conditional epistemic planning with the approaches used in automated planning.

We would like to do plan verification and synthesis in the multi-agent settings. We believe that generalising the notions introduced in this paper to multi-pointed epistemic and event models are key. Plan synthesis in the multi-agent setting is undecidable [11], but considering restricted classes of actions as is done in [17] seems a viable route for achieving decidable multi-agent planning. Another interesting area is to consider modalities such as plausibility and preferences. This would allow an agent to plan for (perhaps only) the most likely outcomes of its own actions and the preferred actions taken by other agents in the system. This could then be combined with the possibility of doing replanning, as mentioned above.

References

1. Andersen, M.B., Bolander, T., Jensen, M.H.: Conditional epistemic planning (long version). Tech. rep. (2012), `http://www2.imm.dtu.dk/~tb/cep-long.pdf`
2. Aucher, G.: An internal version of epistemic logic. Studia Logica 94(1), 1–22 (2010)
3. Aumann, R.J., Hart, S.: Handbook of Game Theory with Economic Applications. Elsevier (1992)
4. Baltag, A., Moss, L.S.: Logics for Epistemic Programs. Synthese 139, 165–224 (2004)
5. Baltag, A., Moss, L.S., Solecki, S.: The logic of public announcements and common knowledge and private suspicions. In: TARK 1998, pp. 43–56 (1998)
6. Baltag, A., Smets, S.: A qualitative theory of dynamic interactive belief revision. In: Bonanno, G., van der Hoek, W., Wooldridge, M. (eds.) Logic and the Foundations of Game and Decision Theory (LOFT7). Texts in Logic and Games, vol. 3, pp. 13–60. Amsterdam University Press (2008)
7. Baral, C., Gelfond, G., Pontelli, E., Son, T.C.: An action language for reasoning about beliefs in multi-agent domains. In: Proceedings of the 14th International Workshop on Non-Monotonic Reasoning (2012)
8. Barwise, J., Moss, L.: Vicious circles. CSLI Publications (1996)
9. van Benthem, J.: Dynamic odds and ends. Technical Report ML-1998-08, University of Amsterdam (1998)
10. van Benthem, J.: Games in dynamic-epistemic logic. Bulletin of Economic Research 53(4), 219–248 (2001)
11. Bolander, T., Andersen, M.B.: Epistemic planning for single- and multi-agent systems. Journal of Applied Non-Classical Logics 21, 9–34 (2011)
12. van Ditmarsch, H., Kooi, B.: Semantic results for ontic and epistemic change. In: LOFT 7, pp. 87–117. Amsterdam University Press (2008)
13. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Springer (2007)
14. Ghallab, M., Nau, D.S., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann (2004)
15. Horrocks, I., Hustadt, U., Sattler, U., Schmidt, R.: Computational modal logic. In: Handbook of Modal Logic. Elsevier (2006)
16. de Lima, T.: Optimal Methods for Reasoning about Actions and Plans in Multi-Agents Systems. Ph.D. thesis, IRIT, University of Toulouse 3, France (2007)
17. Löwe, B., Pacuit, E., Witzel, A.: DEL Planning and Some Tractable Cases. In: van Ditmarsch, H., Lang, J., Ju, S. (eds.) LORI 2011. LNCS, vol. 6953, pp. 179–192. Springer, Heidelberg (2011)
18. Rintanen, J.: Complexity of planning with partial observability. In: Zilberstein, S., Koehler, J., Koenig, S. (eds.) ICAPS, pp. 345–354. AAAI (2004)

PTL: A Propositional Typicality Logic*

Richard Booth[1], Thomas Meyer[2], and Ivan Varzinczak[2]

[1] University of Luxembourg
richard.booth@uni.lu
[2] Centre for Artificial Intelligence Research
CSIR Meraka Institute and UKZN, South Africa
{tommie.meyer,ivan.varzinczak}@meraka.org.za

Abstract. We introduce Propositional Typicality Logic (PTL), a logic for reasoning about typicality. We do so by enriching classical propositional logic with a typicality operator of which the intuition is to capture the most typical (or normal) situations in which a formula holds. The semantics is in terms of ranked models as studied in KLM-style preferential reasoning. This allows us to show that rational consequence relations can be embedded in our logic. Moreover we show that we can define consequence relations on the language of PTL itself, thereby moving beyond the propositional setting. Building on the existing link between propositional rational consequence and belief revision, we show that the same correspondence holds for rational consequence and belief revision on PTL. We investigate entailment for PTL, and propose two appropriate notions thereof.

Keywords: Nonmonotonic reasoning, typicality, belief revision, rationality.

1 Introduction and Motivation

The preferential and rational consequence relations first studied by Lehmann and colleagues in the 90's play a central role in nonmonotonic reasoning [13,14]. This has been the case due to at least three main reasons. Firstly, they are based on semantic constructions that are elegant and neat. Secondly, they provide the foundation for the determination of the important notion of entailment in this context. Finally they also offer an alternative perspective on belief change [9].

A curious aspect of such consequence relations (and corresponding belief revision constructions) is that they are crucially, albeit tacitly, based on a notion of *typicality*. However, in the corresponding underlying language it is not possible to refer directly to such a notion. In this paper, we fill this gap with the introduction of an explicit operator to talk about typicality. Intuitively, our new syntactic construction allows us to single out those most typical situations in which a formula holds. The result is a more expressive language allowing us, for instance, to make statements of the form "the most typical αs are most typical βs", which is not possible in the aforementioned frameworks.

* This work is based upon research supported by the National Research Foundation. Any opinion, findings and conclusions or recommendations expressed in this material are those of the author(s) and therefore the NRF do not accept any liability in regard thereto. This work was partially funded by Project number 247601, Net2: Network for Enabling Networked Knowledge, from the FP7-PEOPLE-2009-IRSES call.

The remainder of the paper is structured as follows: After some preliminaries (Section 2), we define and investigate PTL, a propositional typicality logic extending propositional logic (Section 3). The semantics of PTL is in terms of ranked models as studied in the literature on preferential reasoning. This allows us to embed propositional KLM-style consequence relations in our new language. In Section 4 we show that, although the addition of the typicality operator increases the expressivity of the logic, the nesting of the typicality does not. In Section 5 we investigate the link between AGM belief revision and PTL. We show that propositional AGM belief revision can be expressed in terms of typicality, and also that it can be lifted to a version of revision on PTL. We then move to an investigation of rational consequence relations in terms of PTL (Section 6). We show that propositional rational consequence can be expressed in PTL, that it can be extended to PTL in terms of PTL itself, and that the propositional connection between rational consequence and revision carries over to PTL. In Section 7 we raise the question of what an appropriate notion of entailment for PTL is, and propose at least two candidates. After a discussion of and comparison with related work (Section 8), we conclude with a summary of the contributions and directions for further investigation.

2 Preliminaries

We work in a propositional language over a finite set of *atoms* $\mathcal{P}$, denoted by $p, q, \ldots$ (In later sections we adopt a richer language.) Propositional formulas (and in later sections, formulas of the richer language) are denoted by $\alpha, \beta, \ldots$, and are recursively defined in the usual way: $\alpha ::= p \mid \neg\alpha \mid \alpha \wedge \alpha$. The other truth-functional connectives are defined in terms of $\neg$ and $\wedge$ in the usual way. We use $\top$ as an abbreviation for $p \vee \neg p$, and $\bot$ for $p \wedge \neg p$, for some $p \in \mathcal{P}$. With $\mathcal{L}$ we denote the set of all propositional formulas.

We denote by $\mathscr{U}$ the set of all *valuations* $v : \mathcal{P} \longrightarrow \{0, 1\}$. Satisfaction of $\alpha \in \mathcal{L}$ by $v \in \mathscr{U}$ is defined in the usual truth-functional way. With *Mod*(α) we denote the set of all valuations satisfying α. Given sentences α and β, $\alpha \models \beta$ (α entails β) means *Mod*$(\alpha) \subseteq$ *Mod*(β). We extend the notions of *Mod*$(\cdot)$ and entailment to knowledge bases in the usual way: for a finite $\mathcal{K} \subseteq \mathcal{L}$, *Mod*$(\mathcal{K})$ is the set of all valuations satisfying every formula in $\mathcal{K}$, and $\mathcal{K} \models \alpha$ if and only if *Mod*$(\mathcal{K}) \subseteq$ *Mod*(α).

A propositional defeasible consequence relation $\mathrel{|\!\sim}$ is defined as a binary relation on the formulas of the underlying (finitely generated) propositional logic. $\mathrel{|\!\sim}$ is said to be *preferential* if it satisfies the following set of properties [13]:

$$\text{(Ref) } \alpha \mathrel{|\!\sim} \alpha \qquad \text{(LLE) } \frac{\models \alpha \leftrightarrow \beta,\ \ \alpha \mathrel{|\!\sim} \gamma}{\beta \mathrel{|\!\sim} \gamma} \qquad \text{(And) } \frac{\alpha \mathrel{|\!\sim} \beta,\ \ \alpha \mathrel{|\!\sim} \gamma}{\alpha \mathrel{|\!\sim} \beta \wedge \gamma}$$

$$\text{(Or) } \frac{\alpha \mathrel{|\!\sim} \gamma,\ \ \beta \mathrel{|\!\sim} \gamma}{\alpha \vee \beta \mathrel{|\!\sim} \gamma} \qquad \text{(RW) } \frac{\alpha \mathrel{|\!\sim} \beta,\ \ \models \beta \rightarrow \gamma}{\alpha \mathrel{|\!\sim} \gamma} \qquad \text{(CM) } \frac{\alpha \mathrel{|\!\sim} \beta,\ \ \alpha \mathrel{|\!\sim} \gamma}{\alpha \wedge \beta \mathrel{|\!\sim} \gamma}$$

If, in addition to the properties of preferential consequence, $\mathrel{|\!\sim}$ also satisfies the following Rational Monotonicity property, it is said to be a *rational* consequence relation [14]:

$$\text{(RM) } \frac{\alpha \mathrel{|\!\sim} \beta,\ \ \alpha \mathrel{|\!\not\sim} \neg\gamma}{\alpha \wedge \gamma \mathrel{|\!\sim} \beta}$$

The semantics of (propositional) rational consequence is in terms of *ranked* models. These are partially ordered structures in which the ordering is *modular*.

Definition 1. *Given a set S, $\prec \subseteq S \times S$ is modular if and only if there is a ranking function $rk : S \longrightarrow \mathbb{N}$ such that for every $s, s' \in S$, $s \prec s'$ if and only if $rk(s) < rk(s')$.*

Definition 2. *A ranked model $\mathscr{R}$ is a pair $\langle \mathcal{V}, \prec \rangle$, where $\mathcal{V} \subseteq \mathscr{U}$ and $\prec \subseteq \mathcal{V} \times \mathcal{V}$ is a modular order over $\mathcal{V}$.*[1]

Definition 3. *Let $\alpha \in \mathcal{L}$ and let $\mathscr{R} = \langle \mathcal{V}, \prec \rangle$ be a ranked model. With $[\![\alpha]\!]$ we denote the set of valuations satisfying α in $\mathscr{R}$, defined as follows:*

$$[\![p]\!] := \{v \in \mathcal{V} \mid v(p) = 1\}, \ \ [\![\neg\alpha]\!] := \mathcal{V} \setminus [\![\alpha]\!], \ \ [\![\alpha \wedge \beta]\!] := [\![\alpha]\!] \cap [\![\beta]\!]$$

Given a ranked model $\mathscr{R}$, the intuition is that valuations lower down in the ordering are more preferred than those higher up. Hence, a pair (α, β) is in the consequence relation defined by $\mathscr{R}$ (denoted as $\alpha \mid\!\sim_{\mathscr{R}} \beta$) if and only if $\min_{\prec}[\![\alpha]\!] \subseteq [\![\beta]\!]$, i.e., the most preferred (with respect to $\prec$) α-valuations are also β-valuations.

Lehmann and Magidor provided a representation result for the propositional case, establishing that a defeasible consequence relation $\mid\!\sim$ on $\mathcal{L}$ is rational if and only if it is defined by some ranked model [14,9].

3 Propositional Typicality Logic

We introduce now a propositional typicality logic, called PTL, which extends propositional logic with a typicality operator $\overline{}$ (read 'bar'). The language of PTL, denoted by $\overline{\mathcal{L}}$, is recursively defined by: $\alpha \ ::= p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \overline{\alpha}$. (As before, the other connectives are defined in terms of $\neg$ and $\wedge$, and $\top$ and $\bot$ are abbreviations.) Intuitively, $\overline{\alpha}$ is understood to refer to the typical situations in which α holds. The semantics is in terms of ranked models and we extend the notion of satisfaction from Definition 3 as follows:

Definition 4. *Let $\alpha \in \overline{\mathcal{L}}$ and let $\mathscr{R} = \langle \mathcal{V}, \prec \rangle$. Then $[\![\overline{\alpha}]\!] := \min_{\prec}[\![\alpha]\!]$.*

Given $\alpha \in \overline{\mathcal{L}}$ and $\mathscr{R}$ a ranked model, we say that α is *true* in $\mathscr{R}$ (denoted as $\mathscr{R} \Vdash \alpha$) if $[\![\alpha]\!] = \mathcal{V}$. For $\mathcal{K} \subseteq \overline{\mathcal{L}}$, $\mathscr{R} \Vdash \mathcal{K}$ if $\mathscr{R} \Vdash \alpha$ for every $\alpha \in \mathcal{K}$. α is *valid*, denoted as $\models \alpha$, if $\mathscr{R} \Vdash \alpha$ for every ranked model $\mathscr{R}$.

Note that for every ranked model $\mathscr{R}$ and $\alpha \in \overline{\mathcal{L}}$, there is a $\beta \in \mathcal{L}$ (i.e., a propositional formula) such that $\mathscr{R} \Vdash \alpha \leftrightarrow \beta$. That is to say, given $\mathscr{R}$, every α can be expressed as a propositional formula (β) in $\mathscr{R}$. Of course, this does not mean that propositional logic is as expressive as PTL, since the formula β used to express α in the ranked model $\mathscr{R}$ depends on $\mathscr{R}$. Rather, the relationship between PTL and propositional logic is similar to the relationship between modal logic and propositional logic in the sense that both modal logic and PTL add to propositional logic an operator that is not truth-functional. (In Section 8 we discuss in more detail the relationship between PTL and modal logic.)

Next is a property which shows that if $\mathscr{R} \Vdash \overline{\alpha}$, then $\mathscr{R}$ consists of only α-worlds in which all worlds are incomparable (alias equally preferred) according to $\prec$.

[1] This is not Lehmann and Magidor's [14] original definition of ranked models but a characterization of rational consequence can be given in terms of ranked models as we present here [9].

Proposition 1. *Let $\mathscr{R} = \langle \mathcal{V}, \prec \rangle$. Then (1) $\prec = \emptyset$ iff there is an $\alpha \in \overline{\mathcal{L}}$ such that $\mathscr{R} \Vdash \overline{\alpha}$; (2) for every $\alpha \in \overline{\mathcal{L}}$, $\mathscr{R} \Vdash \overline{\alpha}$ iff for every $\beta \in \mathcal{L}$ such that $\mathscr{R} \Vdash \alpha \to \beta$, $\mathscr{R} \Vdash \overline{\beta}$.*

One of the consequences of this result is that if $\overline{\alpha}$ is true in a ranked model, then so is α (but the converse, of course, does not hold).

Another useful property of typicality is that it allows us to express (propositional) rational consequence, as defined in Section 2.

Proposition 2. *For $\alpha, \beta \in \mathcal{L}$, $\alpha \mid\!\sim_{\mathscr{R}} \beta$ if and only if $\mathscr{R} \Vdash \overline{\alpha} \to \beta$.*

Proposition 2 shows that the introduction of typicality into the object language allows us to express rational consequence. This forms part of our argument to show that our semantics for typicality is the correct one, but it does not provide a justification for introducing all the additional expressivity obtained from typicality. To provide such a justification we turn to the notion of *defeasible incompatibility*. Intuitively, α and β are said to be *incompatible* if they are contradictory. Therefore with an appropriate definition of defeasible incompatibility we should be able to capture the idea of α and β being defeasibly incompatible. There seems to be at least four different ways of expressing defeasible incompatibility, none of which are equivalent (with respect to ranked models), but all of which would be propositionally equivalent if the typicality operator were removed: (*i*) $(\overline{\alpha} \to \neg\beta) \wedge (\overline{\beta} \to \neg\alpha)$; (*ii*) $\overline{\top} \to \neg(\alpha \wedge \beta)$; (*iii*) $\overline{\neg(\alpha \wedge \beta)}$, and (*iv*) $\neg(\overline{\alpha} \wedge \overline{\beta})$. Observe that (*i*) can be expressed as two $\mid\!\sim$-statements ($\alpha \mid\!\sim \neg\beta$ and $\beta \mid\!\sim \neg\alpha$), that (*ii*) can be expressed in terms of $\mid\!\sim$, but that (*iii*) and (*iv*) cannot.

Furthermore, although it may be useful to be able to express all four of these options, our contention is that the notion of defeasible incompatibility is correctly captured by option (*iv*), one of the options that *cannot* be expressed in terms of $\mid\!\sim$. To see why, note firstly that option (*i*) is ruled out because it is too strong. It expressly forbids typical α-situations to be β-situations (and forbids typical β-situations to be α-situations). We could consider weakening it so that typical α-situations are only forbidden to be *typical* β-situations (and similarly with the roles of α and β reversed), i.e., to $(\overline{\alpha} \to \neg\overline{\beta}) \wedge (\overline{\beta} \to \neg\overline{\alpha})$. That looks reasonable indeed, but it is easy to see that this statement is equivalent to each of its two conjuncts $\overline{\alpha} \to \neg\overline{\beta}$ and $\overline{\beta} \to \neg\overline{\alpha}$, and also to option (*iv*). To see why option (*ii*) does not fit the bill either, it is best to consider its representation in terms of $\mid\!\sim$: $\top \mid\!\sim \neg(\alpha \wedge \beta)$. From this we do not always get $\gamma \mid\!\sim \neg(\alpha \wedge \beta)$. Thus, in a sense, option (*ii*) is too weak since it ignores, for the most part, the context in which defeasible incompatibility is supposed to hold. For option (*iii*), from Proposition 1 it follows that if $\overline{\neg(\alpha \wedge \beta)}$ holds, then so does $\neg(\alpha \wedge \beta)$, which is clearly too strong. Finally, observe that option (*iv*) is interpreted to mean that the most typical α-situations and the most typical β-situations are incompatible, which corresponds best to the informal notion of defeasible incompatibility. In summary then, it seems that to express defeasible incompatibility correctly, it is necessary to go beyond rational consequence, but sufficient to introduce typicality into the object language.

Finally, observe that Proposition 2 shows that rational consequence for *propositional logic* can be expressed in PTL. In Section 6 we shall see that it is also possible to express, in PTL itself, the extended notion of rational consequence *for the language $\overline{\mathcal{L}}$.*

4 Typicality Unraveled

In the previous section we have argued for the need to include typicality explicitly in the object language. The observant reader would have noticed that $\overline{\mathcal{L}}$ allows for the arbitrary (finite) nesting of the typicality operator. An important point to consider is whether this much expressivity is needed, and whether it is not perhaps sufficient to restrict the language to non-nested applications of typicality.

In this section we show that once typicality is added to the language, nesting does not increase the expressivity any further, provided that we are allowed to add new propositional atoms. We shall thus be working with languages in which the set of propositional atoms $\mathcal{P}$ may vary, and more specifically, with languages with respect to a given knowledge base. So, given a knowledge base $\mathcal{K}$, we denote by $\mathcal{P}^{\mathcal{K}}$ the set of propositional atoms occurring in $\mathcal{K}$. Furthermore, by a *ranked model on* $\mathcal{P}^{\mathcal{K}}$ we mean a ranked model built up using only the propositional atoms occurring in $\mathcal{P}^{\mathcal{K}}$.

Now, given any $\mathcal{K} \subseteq \overline{\mathcal{L}}$ we: (*i*) Show how to transform every $\beta \in \overline{\mathcal{L}}$ into a formula $\widehat{\beta}$ containing no nested instances of the bar operator (and therefore also how to transform $\mathcal{K}$ into a knowledge base $\widehat{\mathcal{K}}$, containing no nested instances of the bar operator); (*ii*) Show how to construct an auxiliary set of formulas $\widehat{E}$, containing no nested instances of the bar operator, regulating the behavior of the newly introduced propositional atoms, and (*iii*) Show how to transform every ranked model $\mathscr{R}$ on $\mathcal{P}^{\mathcal{K}}$ into its "appropriate representative" $\widehat{\mathscr{R}}$ on $\mathcal{P}^{\widehat{\mathcal{K}}}$ such that, for every $\beta \in \overline{\mathcal{L}}$, β is true in $\mathscr{R}$ if and only if $\widehat{\beta}$ is true in $\widehat{\mathscr{R}}$. Using these constructions we show that $\widehat{\mathcal{K}} \cup \widehat{E}$ is the non-nested version of $\mathcal{K}$ in the sense that the ranked models in which $\widehat{\mathcal{K}} \cup \widehat{E}$ are true are precisely the "appropriate representatives" of the ranked models in which $\mathcal{K}$ is true.

To be more precise, let $\mathcal{K} \subseteq \overline{\mathcal{L}}$, let $S^{\mathcal{K}}$ denote all subformulas of $\mathcal{K}$, and let $B^{\mathcal{K}} = \{\overline{\alpha} \in S^{\mathcal{K}} \mid \alpha \in \mathcal{L}\}$. So $B^{\mathcal{K}}$ contains all occurrences of subformulas in $\mathcal{K}$ containing a single bar. Informally, the idea is to substitute (all occurrences of) every element $\overline{\alpha}$ of $B^{\mathcal{K}}$ with a new atom $p^{\overline{\alpha}}$, and to require that $p^{\overline{\alpha}}$ be equivalent to $\overline{\alpha}$. In doing so we reduce the level of nesting in $\mathcal{K}$ by a factor of 1. Now, let $E^{\mathcal{K}} = \{p^{\overline{\alpha}} \leftrightarrow \overline{\alpha} \mid \overline{\alpha} \in B^{\mathcal{K}}\}$, and for every $\beta \in \overline{\mathcal{L}}$, let $\beta^{\mathcal{K}}$ be obtained from β by the simultaneous substitution in β of (every occurrence of) every $\overline{\alpha} \in B^{\mathcal{K}}$ by $p^{\overline{\alpha}}$ (observe that $\beta^{\mathcal{K}} = \beta$ if β is a propositional formula). We refer to $\beta^{\mathcal{K}}$ as the *$\mathcal{K}$-transform of β*. Also, let $\overline{\mathcal{K}} = \{\beta^{\mathcal{K}} \mid \beta \in \mathcal{K}\}$. The idea is that $\overline{\mathcal{K}} \cup E^{\mathcal{K}}$ is a version of $\mathcal{K}$ with one fewer level of nesting.

Example 1. Let $\mathcal{K} = \{\overline{p \land q} \to r, \overline{\overline{p} \lor r}, \overline{p \land \overline{q \lor \overline{r}}}\}$. Then $B^{\mathcal{K}} = \{\overline{p \land q}, \overline{p}, \overline{r}\}$ and $E^{\mathcal{K}} = \{p^{\overline{p \land q}} \leftrightarrow \overline{p \land q}, p^{\overline{p}} \leftrightarrow \overline{p}, p^{\overline{r}} \leftrightarrow \overline{r}\}$. Now $(\overline{p \land q} \to r)^{\mathcal{K}} = p^{\overline{p \land q}} \to r$, $(\overline{\overline{p} \lor r})^{\mathcal{K}} = \overline{p^{\overline{p}} \lor r}$, $(\overline{p \land \overline{q \lor \overline{r}}})^{\mathcal{K}} = \overline{p \land \overline{q \lor p^{\overline{r}}}}$. Hence $\overline{\mathcal{K}} = \{p^{\overline{p \land q}} \to r, \overline{p^{\overline{p}} \lor r}, \overline{p \land \overline{q \lor p^{\overline{r}}}}\}$. Observe that $\mathcal{K}$ has a nesting level of 3, while $\overline{\mathcal{K}}$ has a nesting level of 2.

Let $\mathscr{R} = \langle \mathcal{V}, \prec \rangle$ be a ranked model on $\mathcal{P}^{\mathcal{K}}$. We define $\overline{\mathscr{R}} = (\overline{\mathcal{V}}, \overline{\prec})$ on $\mathcal{P}^{\overline{\mathcal{K}}}$ as follows: for all $v \in \mathcal{V}$, let $\overline{v}$ be a valuation on $\mathcal{P}^{\overline{\mathcal{K}}}$ such that (*i*) $\overline{v}(p) = v(p)$ for every $p \in \mathcal{P}^{\mathcal{K}}$, and (*ii*) for every $p^{\overline{\alpha}} \in (\mathcal{P}^{\overline{\mathcal{K}}} \setminus \mathcal{P}^{\mathcal{K}})$, $\overline{v}(p^{\overline{\alpha}}) = 1$ if and only if $v \in [\![\overline{\alpha}]\!]$ in $\mathscr{R}$. And for all $\overline{v}, \overline{v'} \in \overline{\mathcal{V}}$, $\overline{v} \,\overline{\prec}\, \overline{v'}$ if and only if $v \prec v'$. So $\overline{\mathscr{R}}$ is an extended version of $\mathscr{R}$ with every valuation v in $\mathscr{R}$ replaced with an extended valuation $\overline{v}$ in which the truth values of atoms occurring in v remain unchanged, and the truth values of the new atoms are constrained by the requirement that every $p^{\overline{\alpha}}$ be equivalent to $\overline{\alpha}$ (for $\overline{\alpha} \in B^{\mathcal{K}}$). We refer to $\overline{\mathscr{R}}$ as the *$\mathcal{K}$-extended version of $\mathscr{R}$*. From this we obtain the following result.

Proposition 3. *For every ranked model $\mathscr{R}$ on $\mathcal{P}^{\mathcal{K}}$, $\overline{\mathscr{R}}$ satisfies $E^{\mathcal{K}}$. Conversely, a ranked model $\mathscr{R}'$ on $\mathcal{P}^{\overline{\mathcal{K}}}$ satisfies $E^{\mathcal{K}}$ if and only if there is a ranked model $\mathscr{R}$ on $\mathcal{P}^{\mathcal{K}}$ such that $\overline{\mathscr{R}} = \mathscr{R}'$. Furthermore, let $\mathscr{R}$ be a ranked model on $\mathcal{P}^{\mathcal{K}}$. Then $\mathscr{R}$ satisfies $\mathcal{K}$ if and only if $\overline{\mathscr{R}} \Vdash \overline{\mathcal{K}}$. Also, for all $\beta \in \overline{\mathcal{L}}$, $\mathscr{R} \Vdash \beta$ if and only if $\overline{\mathscr{R}} \Vdash \overline{\beta}$.*

The proposition above shows that the $\mathcal{K}$-extended version of a ranked model $\mathscr{R}$ is the only "appropriate representative" of $\mathscr{R}$ in the class of ranked models based on the extended language of $\overline{\mathcal{K}}$. Also, the $\mathcal{K}$-extended versions of the ranked models based on the language of $\mathcal{K}$ are the only ones satisfying $E^{\mathcal{K}}$.

As mentioned above, the move from $\mathcal{K}$ to $\overline{\mathcal{K}}$ ensures that we can reduce the level of nesting of $^{-}$ by a factor of 1. To arrive at a set $\widehat{\mathcal{K}}$ not containing any nested occurrences of $^{-}$ we just need to iterate the transform process a sufficient number of times. So, we define $\widehat{\mathcal{K}}$ as follows: Let $\mathcal{K}_0 = \mathcal{K}$, and for $i > 0$, let $B_i = B^{\mathcal{K}_{i-1}}$, $\mathcal{K}_i = \overline{\mathcal{K}_{i-1}}$, and let $n = \min_<\{i \mid B_{i+1} = \emptyset\}$. We then let $\widehat{\mathcal{K}} = \mathcal{K}_n$. So for every $i = 1, \ldots, n$, $\mathcal{K}_i$ has one fewer level of nesting of $^{-}$ than $\mathcal{K}_{i-1}$ until we get to $\mathcal{K}_n = \widehat{\mathcal{K}}$, which has no nested occurrences of $^{-}$. Similarly, for every $\beta \in \overline{\mathcal{L}}$, we define $\widehat{\beta}$ as follows: Let $\beta_0 = \beta$, for $i = 1, \ldots, n$, let $\beta_i = \beta^{\mathcal{K}_{i-1}}$, and let $\widehat{\beta} = \beta_n$. We refer to $\widehat{\beta}$ as the *full $\mathcal{K}$-transform of β*. In a similar vein, we let $\widehat{E} = \bigcup_{i=0}^{i=n-1} E^{\mathcal{K}_i}$.

Example 2. Continuing Example 1, let $\mathcal{K}_0 = \mathcal{K}$. Then $B_1 = B^{\mathcal{K}_0} = B^{\mathcal{K}}$, and $\mathcal{K}_1 = \overline{\mathcal{K}}$ with $E_0 = E^{\mathcal{K}}$; $B_2 = B^{\mathcal{K}_1} = \{\overline{q \vee p^{\overline{r}}}\}$, and $E_1 = \{p^{\overline{q \vee p^{\overline{r}}}} \leftrightarrow \overline{q \vee p^{\overline{r}}}\}$. Now $\mathcal{K}_2 = \overline{\mathcal{K}_1} = \overline{\overline{\mathcal{K}}} = \{p^{\overline{p \wedge q}} \rightarrow r, \overline{p^{\overline{p}} \vee r}, \overline{p \wedge p^{\overline{q \vee p^{\overline{r}}}}}\}$. In the 2nd iteration, $B_3 = B^{\mathcal{K}_2} = \{\overline{p^{\overline{p}} \vee r}, \overline{p \wedge p^{\overline{q \vee p^{\overline{r}}}}}\}$ with $E_2 = \{p^{\overline{p^{\overline{p}} \vee r}} \leftrightarrow \overline{p^{\overline{p}} \vee r}, p^{\overline{p \wedge p^{\overline{q \vee p^{\overline{r}}}}}} \leftrightarrow \overline{p \wedge p^{\overline{q \vee p^{\overline{r}}}}}\}$. Then $\mathcal{K}_3 = \overline{\mathcal{K}_2} = \{p^{\overline{p \wedge q}} \rightarrow r, p^{\overline{p^{\overline{p}} \vee r}}, p^{\overline{p \wedge p^{\overline{q \vee p^{\overline{r}}}}}}\}$. In the next iteration, $B_4 = \emptyset$. Hence $n = 3$, and

$$\widehat{\mathcal{K}} = \left\{ \begin{array}{c} p^{\overline{p \wedge q}} \rightarrow r, p^{\overline{p^{\overline{p}} \vee r}}, \\ p^{\overline{p \wedge p^{\overline{q \vee p^{\overline{r}}}}}} \end{array} \right\}, \widehat{E} = \left\{ \begin{array}{c} p^{\overline{p \wedge q}} \leftrightarrow \overline{p \wedge q}, p^{\overline{p}} \leftrightarrow \overline{p}, p^{\overline{r}} \leftrightarrow \overline{r}, p^{\overline{q \vee p^{\overline{r}}}} \leftrightarrow \overline{q \vee p^{\overline{r}}}, \\ p^{\overline{p^{\overline{p}} \vee r}} \leftrightarrow \overline{p^{\overline{p}} \vee r}, p^{\overline{p \wedge p^{\overline{q \vee p^{\overline{r}}}}}} \leftrightarrow \overline{p \wedge p^{\overline{q \vee p^{\overline{r}}}}} \end{array} \right\}$$

Finally, for any ranked model $\mathscr{R}$ on $\mathcal{P}^{\mathcal{K}}$, we define its *full $\mathcal{K}$-extended version* $\widehat{\mathscr{R}}$ as follows: Let $\mathscr{R}_0 = \mathscr{R}$, and for $i = 1, \ldots, n$, $\mathscr{R}_i = \overline{\mathscr{R}_{i-1}}$. Then we let $\widehat{\mathscr{R}} = \mathscr{R}_n$.

Using Proposition 3 we then obtain the result we require.

Theorem 1. *For every $\mathscr{R}$ on $\mathcal{P}^{\mathcal{K}}$, its full $\mathcal{K}$-extended version $\widehat{\mathscr{R}}$ satisfies $\widehat{E}$. Conversely, a ranked model $\mathscr{R}'$ on $\mathcal{P}^{\widehat{\mathcal{K}}}$ satisfies $\widehat{E}$ if and only if there is a ranked model $\mathscr{R}$ on $\mathcal{P}^{\mathcal{K}}$ such that $\mathscr{R}' = \widehat{\mathscr{R}}$. Furthermore, let $\mathscr{R}$ be a ranked model $\mathscr{R}$ on $\mathcal{P}^{\mathcal{K}}$. Then $\mathscr{R} \Vdash \mathcal{K}$ if and only if $\widehat{\mathscr{R}} \Vdash \widehat{\mathcal{K}}$. Also, for all $\beta \in \overline{\mathcal{L}}$, $\mathscr{R} \Vdash \beta$ if and only if $\widehat{\mathscr{R}} \Vdash \widehat{\beta}$.*

5 Belief Revision and Typicality

Given the well-known link between propositional rational consequence and AGM belief revision [1], as developed by Gärdenfors and Makinson [9], it is perhaps not surprising that propositional AGM belief revision can be expressed using the typicality operator. In this section we make this claim precise. The formal representation of propositional AGM revision we provide below is based on that of Katsuno and Mendelzon [12].

The starting point is to fix a non-empty subset $\mathcal{V}$ of $\mathscr{U}$ (as done by Kraus et al. [13]), and to assume that everything is done within the context of $\mathcal{V}$. In that sense, $\mathcal{V}$ becomes the set of *all* valuations available to us. This is slightly more general than the Katsuno-Mendelzon framework which assumes $\mathcal{V}$ to be equal to $\mathscr{U}$, but is a special case of the original AGM approach. To reflect this restriction, we use $\mathit{Mod}_{\mathcal{V}}(\alpha)$ to denote the set $\mathit{Mod}(\alpha) \cap \mathcal{V}$. In the same vein, in the postulates below, validity is understood to be modulo $\mathcal{V}$. That is, for $\alpha \in \mathcal{L}$ we let $\models \alpha$ if and only if $\mathit{Mod}_{\mathcal{V}}(\alpha) = \mathcal{V}$.

Next, we fix a knowledge base $\kappa \in \mathcal{L}$ (i.e., represented as a propositional formula) such that $\mathit{Mod}_{\mathcal{V}}(\kappa) \neq \emptyset$. A revision operator $\circ$ on $\mathcal{L}$ for κ is a function from $\mathcal{L}$ to $\mathcal{L}$. Intuitively, $\kappa \circ \alpha$ is the result of revising κ by α (clearly the models of $\kappa \circ \alpha$ should be in $\mathcal{V}$). An AGM revision operator $\circ$ on $\mathcal{L}$ for κ is a revision operator on $\mathcal{L}$ for κ which satisfies the following six properties:

(R1) $\models (\kappa \circ \alpha) \rightarrow \alpha$
(R2) If $\not\models \neg(\kappa \wedge \alpha)$, then $\models (\kappa \circ \alpha) \leftrightarrow (\kappa \wedge \alpha)$
(R3) If $\not\models \neg\alpha$, then $\not\models \neg(\kappa \circ \alpha)$
(R4) If $\models \kappa_1 \leftrightarrow \kappa_2$ and $\models \alpha_1 \leftrightarrow \alpha_2$, then $\models (\kappa_1 \circ \alpha_1) \leftrightarrow (\kappa_2 \circ \alpha_2)$
(R5) $\models ((\kappa \circ \alpha) \wedge \beta) \rightarrow (\kappa \circ (\alpha \wedge \beta))$
(R6) If $\not\models \neg(\kappa \circ \alpha) \wedge \beta$, then $\models (\kappa \circ (\alpha \wedge \beta)) \rightarrow ((\kappa \circ \alpha) \wedge \beta)$

A ranked model $\mathscr{R} = \langle \mathcal{V}, \prec \rangle$ is defined as *κ-faithful* if and only if $\min_{\prec} \mathcal{V} = \mathit{Mod}_{\mathcal{V}}(\kappa)$. A revision operator $\circ_{\mathscr{R}}$ (on $\mathcal{L}$) is *defined by a κ-faithful ranked model $\mathscr{R}$* if and only if $\mathit{Mod}_{\mathcal{V}}(\kappa \circ_{\mathscr{R}} \alpha) = \min_{\prec} \mathit{Mod}_{\mathcal{V}}(\alpha)$. Katsuno and Mendelzon [12] proved that for $\mathcal{V} = \mathscr{U}$, (*i*) every revision operator $\circ_{\mathscr{R}}$ defined by a κ-faithful ranked model $\mathscr{R}$ is an AGM revision operator (on $\mathcal{L}$), and (*ii*) for every AGM revision operator $\circ$ (on $\mathcal{L}$) for κ, there is a κ-faithful ranked model $\mathscr{R}$ such that $\mathit{Mod}_{\mathcal{V}}(\kappa \circ \alpha) = \mathit{Mod}_{\mathcal{V}}(\kappa \circ_{\mathscr{R}} \alpha)$.

We show that $\circ$ can be expressed in $\overline{\mathcal{L}}$ using typicality. The key insight is to identify the knowledge base κ to be revised with the formula $\overline{\top}$, while $\kappa \circ \alpha$ is identified with $\overline{\alpha}$.

Proposition 4. *Let $\mathscr{R} = \langle \mathcal{V}, \prec \rangle$ be any κ-faithful ranked model (with $\kappa \in \mathcal{L}$). Then $[\![\kappa \circ_{\mathscr{R}} \alpha]\!] = [\![\overline{\alpha}]\!]$ (for every $\alpha \in \mathcal{L}$). Conversely, let $\circ$ be any AGM revision operator (on $\mathcal{L}$) for κ. Then there is a κ-faithful ranked model $\mathscr{R}$ such that $\mathit{Mod}_{\mathcal{V}}(\kappa \circ \alpha) = [\![\overline{\alpha}]\!]$.*

This result shows that propositional AGM revision can be embedded in $\overline{\mathcal{L}}$. But we can take this a step further and extend revision to apply to the language $\overline{\mathcal{L}}$ as well. So, with $\mathcal{V}$ still fixed, we let $\mathcal{R}_{\mathcal{V}} = \{\mathscr{R} \mid \mathscr{R} = \langle \mathcal{V}, \prec \rangle\}$ and we fix a $\kappa \in \overline{\mathcal{L}}$ such that $\mathscr{R} \not\Vdash \neg\kappa$ for some $\mathscr{R} \in \mathcal{R}_{\mathcal{V}}$. The definition of a revision operator $\circ$ is then the same as above, except that it is now with respect to $\overline{\mathcal{L}}$. And the definition of an AGM revision operator on $\overline{\mathcal{L}}$ is then one which satisfies (R1)–(R6), but with validity in the postulates understood to be modulo $\mathcal{R}_{\mathcal{V}}$ (that is, for $\alpha \in \overline{\mathcal{L}}$, $\models \alpha$ if and only if $\mathscr{R} \Vdash \alpha$ for every $\mathscr{R} \in \mathcal{R}_{\mathcal{V}}$.) This gives us a representation result similar to that of Katsuno and Mendelzon, but with the revision operator defined on $\overline{\mathcal{L}}$.

Theorem 2. *Let $\mathscr{R}$ be a κ-faithful ranked model. Then $\circ_{\mathscr{R}}$ is an AGM revision operator on $\overline{\mathcal{L}}$ for κ. Conversely, for every AGM revision operator $\circ$ on $\overline{\mathcal{L}}$ for κ there is a κ-faithful ranked model $\mathscr{R}$ such that $\mathit{Mod}_{\mathcal{V}}(\kappa \circ \alpha) = \mathit{Mod}_{\mathcal{V}}(\kappa \circ_{\mathscr{R}} \alpha)$.*

6 Rational Consequence on $\overline{\mathcal{L}}$

We have seen in Section 5 that typicality can be used to express propositional AGM belief revision, as well as AGM belief revision defined for PTL. From Proposition 2 we know that rational consequence for propositional logic can be expressed in PTL. In this section we complete the picture by showing that (*i*) rational consequence for PTL can be expressed in PTL itself, a result analogous to Theorem 2, and (*ii*) that the expected connection between rational consequence and AGM revision for PTL does indeed hold.

As in Section 5, we start by fixing a set $\mathcal{V} \subseteq \mathscr{U}$. In this case, however, $\mathcal{V}$ is allowed to be empty as well. Then we let $\mid\!\sim$ be a binary relation on $\overline{\mathcal{L}}$. We say that $\mid\!\sim$ is a *rational consequence relation on* $\overline{\mathcal{L}}$ if and only if it satisfies the seven rationality properties from Section 2. In this case (as in Section 5) $\models$ is understood to be validity modulo $\mathcal{R}_{\mathcal{V}}$: $\models \alpha$ if and only if for every $\mathscr{R} \in \mathcal{R}_{\mathcal{V}}$, $\mathscr{R} \Vdash \alpha$. As was done in Section 2, given a ranked model $\mathscr{R}$, a pair (α, β) is in the consequence relation defined by $\mathscr{R}$ (denoted as $\alpha \mid\!\sim_{\mathscr{R}} \beta$) if and only if $\min_{\prec}[\![\alpha]\!] \subseteq [\![\beta]\!]$. In this case, however, α and β are taken to be elements of $\overline{\mathcal{L}}$ and not just of $\mathcal{L}$. From this we get the following:

Theorem 3. *Every $\mid\!\sim_{\mathscr{R}}$ defined by some $\mathscr{R}$ is a rational consequence relation on $\overline{\mathcal{L}}$. Conversely, for every rational consequence relation $\mid\!\sim$ on $\overline{\mathcal{L}}$ there exists a ranked model $\mathscr{R}$ such that $\mid\!\sim_{\mathscr{R}} = \mid\!\sim$.*

It is worth mentioning that the proof of Theorem 3 makes use of Theorem 2, as well as the connection between AGM revision and rational consequence for PTL in the style of Gärdenfors and Makinson [9], which we now proceed to describe. First we consider the following additional property on defeasible consequence relations:

$$\text{(Cons)}\quad \top \not\mid\!\sim \bot$$

It is easy to see that for a ranked model $\mathscr{R} = \langle \mathcal{V}, \prec \rangle$, $\top \mid\!\sim_{\mathscr{R}} \bot$ if and only if $\mathcal{V} = \emptyset$. By insisting that (Cons) holds, we are restricting ourselves to ranked models in which $\mathcal{V} \neq \emptyset$, a restriction that is necessary to comply with postulate (R3) for AGM belief revision. So, we consider only the case where the (fixed) set $\mathcal{V}$ is non-empty.[2]

Intuitively, given a rational consequence relation $\mid\!\sim$ and a belief revision operator $\circ$ for a knowledge base κ, the idea is to (*i*) associate κ with all βs such that $\top \mid\!\sim \beta$ holds and (*ii*) to associate the consequences of $\kappa \circ \alpha$ with all the βs such that $\alpha \mid\!\sim \beta$ holds.

For a rational $\mid\!\sim$ on $\overline{\mathcal{L}}$, let $C^{\mid\sim} = \{\alpha \in \overline{\mathcal{L}} \mid \top \mid\!\sim \alpha\}$ and let $\mathcal{K}^{\mid\sim}$ be the set of logically strongest formulas (modulo $\mathcal{R}_{\mathcal{V}}$) to be defeasibly concluded from $\top$. That is,

$$\mathcal{K}^{\mid\sim} = \{\alpha \in C^{\mid\sim} \mid \text{ for all } \beta \in C^{\mid\sim},\ \text{if } \models \beta \rightarrow \alpha,\ \text{then } \models \alpha \rightarrow \beta\}.^{3}$$

Theorem 4. *Let $\mid\!\sim$ be a rational consequence relation on $\overline{\mathcal{L}}$ also satisfying (Cons), and let $\kappa \in \mathcal{K}^{\mid\sim}$. There is an AGM revision operator $\circ$ on $\overline{\mathcal{L}}$ for κ such that $\alpha \mid\!\sim \beta$ if and only if $\models (\kappa \circ \alpha) \rightarrow \beta$. Conversely, let κ be any element of $\overline{\mathcal{L}}$ such that $\not\models \neg\kappa$, and let $\circ$ be an AGM revision operator on $\overline{\mathcal{L}}$ for κ. Then there is a rational consequence relation $\mid\!\sim$ on $\overline{\mathcal{L}}$ also satisfying (Cons) such that $\alpha \mid\!\sim \beta$ if and only if $\models (\kappa \circ \alpha) \rightarrow \beta$.*

[2] It is easy to see that if $\mathcal{V} = \emptyset$, then $\overline{\mathcal{L}} \times \overline{\mathcal{L}}$ is the only rational consequence relation satisfying all seven rationality properties, that $\mathscr{R} = \langle \emptyset, \emptyset \rangle$ is the only ranked model, and that $\mid\!\sim_{\mathscr{R}} = \overline{\mathcal{L}} \times \overline{\mathcal{L}}$.

[3] Where $\models$ is understood to mean validity modulo $\mathcal{R}_{\mathcal{V}}$.

7 Entailment for PTL

In this section we focus on what is perhaps the central question concerning PTL from the perspective of knowledge representation and reasoning: What does it mean for a PTL formula to be *entailed* by a (finite) knowledge base $\mathcal{K}$? Formally, we view an entailment relation as a binary relation $\models^*$ from the power set of the language under consideration (in this case $\overline{\mathcal{L}}$) to the language itself. Its associated *consequence* relation is defined as: $Cn^*(\mathcal{K}) = \{\alpha \mid \mathcal{K} \models^* \alpha\}$. Before looking at specific candidates, we propose some desired properties for such an entailment relation. The obvious place to start is to consider the properties for Tarskian consequence below.

(Inclusion) $\mathcal{K} \subseteq Cn^*(\mathcal{K})$
(Idempotency) $Cn^*(\mathcal{K}) = Cn^*(Cn^*(\mathcal{K}))$
(Monotonicity) If $\mathcal{K}_1 \subseteq \mathcal{K}_2$, then $Cn^*(\mathcal{K}_1) \subseteq Cn^*(\mathcal{K}_2)$

Inclusion and Idempotency are both properties we want to have satisfied, but Monotonicity is not. To see why not, it is enough to refer to the classic example in nonmonotonic reasoning: Let $\mathcal{K}_1 = \{p \to b, \overline{b} \to f\}$ (penguins are birds, and birds typically fly), and let $\mathcal{K}_2 = \mathcal{K}_1 \cup \{\overline{p} \to \neg f\}$ (add to $\mathcal{K}_1$ that penguins typically do not fly). We want $\overline{p} \to f \in Cn^*(\mathcal{K}_1)$ (penguins typically fly as a consequence of $\mathcal{K}_1$), but we want $\overline{p} \to f \notin Cn^*(\mathcal{K}_2)$ (penguins typically fly *not* as a consequence of $\mathcal{K}_2$), thereby invalidating Monotonicity.

In addition to Inclusion and Idempotency we require $\models^*$ to behave classically when presented with propositional information only (below $\models$ denotes classical entailment):

(Classic) If $\mathcal{K} \subseteq \mathcal{L}$, then for every $\alpha \in \mathcal{L}$, $\mathcal{K} \models^* \alpha$ iff $\mathcal{K} \models \alpha$

Therefore, we also require that the classical consequences of a knowledge base expressed in $\overline{\mathcal{L}}$ be classically closed (below $Cn(\cdot)$ refers to classical consequence of $\mathcal{L}$):

(Classic Closure) $Cn^*(\mathcal{K}) \cap \mathcal{L} = Cn(Cn^*(\mathcal{K}) \cap \mathcal{L})$

We now consider an obvious candidate for entailment: the standard Tarskian notion of entailment applied to the semantics of PTL:

$\mathcal{K} \models^T \alpha$ iff every ranked model $\mathscr{R}$ satisfying $\mathcal{K}$ also satisfies α

It is easy to show that $\models^T$ satisfies Inclusion, Idempotency, Classic, and Classic Closure. However, it also satisfies Monotonicity, which eliminates it from contention as a viable form of entailment. Moreover, there is an additional argument against the use of $\models^T$ as well, one that is based on an adaptation of a result obtained by Lehmann and Magidor in the propositional case [14]. To make the argument, we first present a result showing that all formulas of $\overline{\mathcal{L}}$ can be rewritten as statements of rational consequence:

Lemma 1. *For every $\mathscr{R}$ and $\alpha \in \overline{\mathcal{L}}$, $\mathscr{R} \Vdash \alpha$ if and only if $\mathscr{R} \Vdash \overline{\neg\alpha} \to \bot$ if and only if $\neg\alpha \mid\!\sim_{\mathscr{R}} \bot$. Conversely for every $\mathscr{R}$ and $\alpha, \beta \in \overline{\mathcal{L}}$, $\alpha \mid\!\sim_{\mathscr{R}} \beta$ if and only if $\mathscr{R} \Vdash \overline{\alpha} \to \beta$.*

We can therefore think of $\overline{\mathcal{L}}$ as a language for expressing defeasible consequence on $\overline{\mathcal{L}}$ with $\mid\!\sim$ viewed as the only main connective. More precisely, let $\overline{\mathcal{L}}_{\mid\sim} = \{\alpha \mid\!\sim \beta \mid \alpha, \beta \in \overline{\mathcal{L}}\}$, and for any ranked model $\mathscr{R}$, let $\mathscr{R} \Vdash \alpha \mid\!\sim \beta$ if and only if $\alpha \mid\!\sim_{\mathscr{R}} \beta$. The next result shows that the languages $\overline{\mathcal{L}}$ and $\overline{\mathcal{L}}_{\mid\sim}$ are equally expressive.

Proposition 5. *For every $\mathscr{R}$ and $\alpha \mid\!\sim \beta \in \overline{\mathcal{L}}_{\mid\sim}$, $\mathscr{R} \Vdash \alpha \mid\!\sim \beta$ if and only if $\mathscr{R} \Vdash \overline{\alpha} \rightarrow \beta$. Conversely, for every $\mathscr{R}$ and $\alpha \in \overline{\mathcal{L}}$, $\mathscr{R} \Vdash \alpha$ if and only if $\mathscr{R} \Vdash \neg\alpha \mid\!\sim \bot$.*

$\overline{\mathcal{L}}_{\mid\sim}$ is similar to the language for conditional knowledge bases studied by Lehmann and Magidor, but with the propositional component replaced by $\overline{\mathcal{L}}$ (i.e., $\mid\!\sim \subseteq \overline{\mathcal{L}} \times \overline{\mathcal{L}}$).

Based on this we restate entailment in terms of the language $\overline{\mathcal{L}}_{\mid\sim}$, and propose an additional property that any appropriate notion of entailment should satisfy. Let $\mathcal{K}$ be a (finite) subset of $\overline{\mathcal{L}}_{\mid\sim}$, let $\models^*$ be a (potential) entailment relation from $\mathscr{P}(\overline{\mathcal{L}}_{\mid\sim})$ to $\overline{\mathcal{L}}_{\mid\sim}$, and let $\mid\!\sim^*_{\mathcal{K}}$ be a defeasible consequence relation on $\overline{\mathcal{L}}$ obtained from $\models^*$ as follows: $\alpha \mid\!\sim^*_{\mathcal{K}} \beta$ if and only if $\mathcal{K} \models^* \alpha \mid\!\sim \beta$.

(Rationality) For every finite $\mathcal{K} \subseteq \overline{\mathcal{L}}_{\mid\sim}$, the consequence relation $\mid\!\sim^*_{\mathcal{K}}$ obtained from $\models^*$ should be a *rational*

Rationality is essentially the property for the entailment of propositional conditional knowledge bases proposed by Lehmann and Magidor [14], but applied to $\overline{\mathcal{L}}_{\mid\sim}$. Based on their results, it follows that $\models^T$ (defined on $\overline{\mathcal{L}}_{\mid\sim}$) does not satisfy Rationality. In fact, analogous to one of their results, we have the following result.

Proposition 6. *For finite $\mathcal{K} \subseteq \overline{\mathcal{L}}_{\mid\sim}$, let $\mid\!\sim^{\mathcal{K}} = \{(\alpha, \beta) \mid \alpha \mid\!\sim \beta \in \mathcal{K}\}$, and let $\mid\!\sim^P$ be the intersection of all preferential consequence relations on $\overline{\mathcal{L}}$ containing $\mid\!\sim^{\mathcal{K}}$. For the consequence relation $\mid\!\sim^T_{\mathcal{K}}$ obtained from $\models^T$, it follows that $\mid\!\sim^*_{\mathcal{K}} = \mid\!\sim^P$ is a preferential consequence relation, but not necessarily a rational consequence relation.*

Since $\overline{\mathcal{L}}$ and $\overline{\mathcal{L}}_{\mid\sim}$ are equally expressive, Proposition 6 provides additional evidence that $\models^T$ is not an appropriate form of entailment.

7.1 Rational Closure for PTL

Having shown that $\models^T$ is *not* an appropriate form of entailment for PTL, we now turn our attention to a proposal for an appropriate version of entailment. It is the notion of the *rational closure* of a conditional knowledge base, proposed by Lehmann and Magidor for the propositional case, applied to $\overline{\mathcal{L}}_{\mid\sim}$.

Definition 5. *Let $\mid\!\sim_0$ and $\mid\!\sim_1$ be rational consequence relations. $\mid\!\sim_0$ is preferable to $\mid\!\sim_1$ (written $\mid\!\sim_0 \ll \mid\!\sim_1$) if and only if*

- *there is an $\alpha \mid\!\sim \beta \in \mid\!\sim_1 \setminus \mid\!\sim_0$ s.t. for all γ s.t. $\gamma \vee \alpha \mid\!\sim_0 \neg\alpha$ and for all δ s.t. $\gamma \mid\!\sim_0 \delta$, we also have $\gamma \mid\!\sim_1 \delta$;*
- *for every $\gamma, \delta \in \mathcal{L}$, if $\gamma \mid\!\sim \delta$ is in $\mid\!\sim_0 \setminus \mid\!\sim_1$, then there is an assertion $\rho \mid\!\sim \nu$ in $\mid\!\sim_1 \setminus \mid\!\sim_0$ s.t. $\rho \vee \gamma \mid\!\sim_1 \neg\gamma$.*

The motivation for $\ll$ here is essentially that for the same ordering for the propositional case provided by Lehmann and Magidor [14]. Given $\mathcal{K} \subseteq \overline{\mathcal{L}}$, the idea is now to define the rational closure as the most preferred (with respect to $\ll$) of all those rational consequence relations which include $\mathcal{K}$.

Lemma 2. *Let $\mathcal{K}$ be a finite subset of $\overline{\mathcal{L}}_{\mid\sim}$ and let $\mid\!\sim^{\mathcal{K}} = \{(\alpha, \beta) \mid \alpha \mid\!\sim \beta \in \mathcal{K}\}$. There is a unique rational consequence relation containing $\mid\!\sim^{\mathcal{K}}$ which is preferable (with respect to $\ll$) to all other rational consequence relations containing $\mid\!\sim^{\mathcal{K}}$.*

This allows us to define the rational closure $\models^{rc}$ of a knowledge base on $\overline{\mathcal{L}}_{\mid\sim}$.

Definition 6. *For finite $\mathcal{K} \subseteq \overline{\mathcal{L}}_{\mid\sim}$, let $\mid\sim^{\mathcal{K}} = \{(\alpha,\beta) \mid \alpha \mid\sim \beta \in \mathcal{K}\}$, and let $\mid\sim^{rc}$ be the (unique) rational consequence relation containing $\mid\sim^{\mathcal{K}}$ which is preferable (with respect to $\ll$) to all other rational consequence relations containing $\mid\sim^{\mathcal{K}}$. Then $\alpha \mid\sim \beta$ is in the rational closure of $\mathcal{K}$ (written as $\mathcal{K} \models^{rc} \alpha \mid\sim \beta$) if and only if $\alpha \mid\sim^{rc} \beta$.*

Definition 6 gives us a notion of rational closure for $\overline{\mathcal{L}}_{\mid\sim}$. Since $\overline{\mathcal{L}}$ and $\overline{\mathcal{L}}_{\mid\sim}$ are equally expressive, we can use Definition 6 to define rational closure for $\overline{\mathcal{L}}$ as well:

Definition 7. *Let $\mathcal{K} \subseteq \overline{\mathcal{L}}$, $\alpha \in \overline{\mathcal{L}}$, and let $\mathcal{K}^{\mid\sim} = \{\neg\beta \mid\sim \bot \mid \beta \in \mathcal{K}\}$. α is in the rational closure of $\mathcal{K}$ (written as $\mathcal{K} \models^{rc} \alpha$) if and only if $\neg\alpha \mid\sim \bot$ is in the rational closure of $\mathcal{K}^{\mid\sim}$.*

It is not hard to show that rational closure satisfies Inclusion, Idempotency, Classic, Classic Closure, and Rationality, but not Monotonicity. It is therefore a reasonable candidate for entailment for PTL.

7.2 Minimum Entailment for PTL

In this section we turn our attention to another proposal for entailment for $\overline{\mathcal{L}}$ based on a semantic construction. It is inspired by a proposal by Giordano et al. [10]. The idea is to define a partial order on a certain subclass of ranked models satisfying a knowledge base $\mathcal{K} \subseteq \overline{\mathcal{L}}$, with models lower down in the ordering being viewed as more 'conservative', in the sense that one can draw fewer conclusions from them, and therefore being more preferred. For $\mathcal{K} \subseteq \overline{\mathcal{L}}$, let $\mathcal{V}^{\mathcal{K}}$ be the elements of $\mathscr{U}$ permitted by $\mathcal{K}$: $\mathcal{V}^{\mathcal{K}} = \{v \mid v \in \mathcal{V}$ for some $\mathscr{R} = \langle \mathcal{V}, \prec\rangle$ s.t. $\mathscr{R} \Vdash \mathcal{K}\}$. And let $\mathcal{R}^{\mathcal{K}} = \{\mathscr{R} = \langle \mathcal{V}^{\mathcal{K}}, \prec\rangle \mid \mathscr{R} \Vdash \mathcal{K}\}$. Now, for any $\mathscr{R} = \langle \mathcal{V}^{\mathcal{K}}, \prec\rangle \in \mathcal{R}^{\mathcal{K}}$, let $\mathcal{V}_0^{\mathscr{R}} = \min_\prec \mathcal{V}^{\mathcal{K}}$, and for $i > 0$ let $\mathcal{V}_i^{\mathscr{R}} = \min_\prec \left(\mathcal{V}^{\mathcal{K}} \setminus (\cup_{j=0}^{j=i-1} \mathcal{V}_j^{\mathscr{R}})\right)$. So $\mathcal{V}_0^{\mathscr{R}}$ contains the elements of $\mathcal{V}^{\mathcal{K}}$ lowest down w.r.t. $\prec$, $\mathcal{V}_1^{\mathscr{R}}$ contains the elements of $\mathcal{V}^{\mathcal{K}}$ just above $\mathcal{V}_0^{\mathscr{R}}$ w.r.t. $\prec$, etc. Next, for every $v \in \mathcal{V}^{\mathcal{K}}$ we define the *height* of v in $\mathscr{R}$ as $h^{\mathscr{R}}(v) = i$ if and only if $v \in \mathcal{V}_i^{\mathscr{R}}$. And based on that, we define the partial order $\preceq$ on $\mathcal{R}^{\mathcal{K}}$ as follows: $\mathscr{R}_1 \preceq \mathscr{R}_2$ if and only if for every $v \in \mathcal{V}^{\mathcal{K}}$, $h^{\mathscr{R}_1}(v) \leq h^{\mathscr{R}_2}(v)$. From this we get:

Proposition 7. *For every $\mathcal{K} \subseteq \overline{\mathcal{L}}$, the partial order $\preceq$ on the elements of $\mathcal{R}^{\mathcal{K}}$ has a unique minimum element.*

This allows us to provide a definition for the *minimum entailment* of a knowledge base.

Definition 8. *Let $\mathcal{K} \in \overline{\mathcal{L}}$, $\alpha \in \overline{\mathcal{L}}$, and $\mathscr{R}^{\mathcal{K}}$ be the (unique) minimum element of $\mathcal{R}^{\mathcal{K}}$ w.r.t. the partial order $\preceq$ on $\mathcal{R}^{\mathcal{K}}$. Then α is in the minimum entailment of $\mathcal{K}$ ($\mathcal{K} \models^{\min} \alpha$) if and only if $\mathscr{R}^{\mathcal{K}} \Vdash \alpha$.*

It can be shown that minimum entailment satisfies Inclusion, Idempotency, Classic, Classic Closure, and Rationality, but not Monotonicity. As for rational closure, it is a reasonable candidate for entailment for $\overline{\mathcal{L}}$. In fact, the connection between rational closure and minimal entailment may even be closer than that. There is strong evidence to support the conjecture that they actually coincide.

8 Related Work

To the best of our knowledge, the first attempt to formalize a notion of typicality in the context of defeasible reasoning was that by Delgrande [8]. Given the relationship between our constructions and those by Kraus et al., most of the remarks in the comparison made by Lehmann and Magidor [14, Section 3.7] are applicable in comparing Delgrande's approach to ours and we do not repeat them here.

Crocco and Lamarre [7] as well as Boutilier [2] have also explored the links between defeasible consequence relations and notions of normality similar to ours. In particular, Boutilier showed that nonmonotonic consequence can be embedded in conditional logics via a binary modality $\Rightarrow$. Here we have considered a unary operator. The links between our $\overline{}$ and the conditional $\Rightarrow$ remain to be explored, though. Our conjecture at the moment is that they provide for the same expressivity.

In a description logic setting, Giordano et al. [11] also study notions of typicality. Semantically, they do so by placing an (absolute) ordering on *objects* in first-order domains in order to define versions of defeasible subsumption relations in the description logic $\mathcal{ALC}$. The authors moreover extend the language of $\mathcal{ALC}$ with an explicit typicality operator $\mathbf{T}$ of which the intended meaning is to single out instances of a concept that are deemed as 'typical'. That is, given an $\mathcal{ALC}$ concept C, $\mathbf{T}(C)$ denotes the most typical individuals having the property of being C in a particular DL interpretation.

Giordano et al.'s approach defines rational versions of the DL subsumption relation $\sqsubseteq$. Nevertheless, they do not provide representation results and do not address the question of entailment either. In a recent paper [5] we have addressed precisely these issues in DLs. Even though here we have investigated typicality in a propositional setting, we expect that our representation result and constructions for the rational closure (as well as the links with belief revision) can be lifted to the DL case.

Britz et al. [3] investigate another embedding of propositional preferential reasoning in modal logic. In their setting, the modular ordering is an accessibility relation on possible worlds, axiomatized via a modal operator $\Box$. Our typicality operator can be defined in terms of their modality as $\overline{\alpha} \equiv_{\text{def}} \Box\neg\alpha \wedge \alpha$. The modal sentence $\Box\neg\alpha \wedge \alpha$ says that the worlds satisfying it are α-worlds and whatever world is more preferable than these is a $\neg\alpha$-world. In other words, these are the minimal α-worlds. The general case of defining Britz et al.'s modality in terms of our typicality operator is not possible, but in a finitely generated language as we consider here, the logics become identical.

Britz and Varzinczak [6] investigate another aspect of defeasibility by introducing (non-standard) modal operators allowing us to talk about relative normality in accessible worlds. With their defeasible versions of modalities, it is possible to make statements of the form "α holds in all of the normal (typical) accessible worlds", thereby capturing defeasibility of what is 'expected' in target worlds. This allows for the definition of a family of modal logics in which defeasible modes of inference can be expressed, and which can be integrated with existing $\mid\!\sim$-based modal logics [4].

9 Concluding Remarks

The contributions of this work are as follows: (*i*) We present the logic PTL which provides a formal account of typicality with which to capture the most typical situations in which a given formula holds; (*ii*) We show that we can embed the (propositional) KLM framework within PTL, and we also define rational consequence on PTL itself; (*iii*) We establish a connection between rational consequence and belief revision, both on PTL, and (*iv*) We investigate appropriate notions of entailment for PTL and propose two candidates.

For future work we are interested in algorithms for computing the appropriate forms of entailment for PTL, specifically algorithms that can be reduced to validity checking for PTL. It follows indirectly from results by Lehmann and Magidor [14] that this type of entailment has the same worst-case complexity of validity checking for PTL. Given the links with modal logic, we know that this is at least in PSPACE. Finally we plan to extend PTL to more expressive logics such as description logics and modal logics.

References

1. Alchourrón, C., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. Journal of Symbolic Logic 50, 510–530 (1985)
2. Boutilier, C.: Conditional logics of normality: A modal approach. Artificial Intelligence 68(1), 87–154 (1994)
3. Britz, K., Heidema, J., Labuschagne, W.: Semantics for dual preferential entailment. Journal of Phil. Logic 38, 433–446 (2009)
4. Britz, K., Meyer, T., Varzinczak, I.: Preferential reasoning for modal logics. Electronic Notes in Theoretical Computer Science 278, 55–69 (2011)
5. Britz, K., Meyer, T., Varzinczak, I.: Semantic Foundation for Preferential Description Logics. In: Wang, D., Reynolds, M. (eds.) AI 2011. LNCS (LNAI), vol. 7106, pp. 491–500. Springer, Heidelberg (2011)
6. Britz, K., Varzinczak, I.: Defeasible modes of inference: A preferential perspective. In: 14th International Workshop on Nonmonotonic Reasoning, NMR (2012)
7. Crocco, G., Lamarre, P.: On the connections between nonmonotonic inference systems and conditional logics. In: Proc. KR, pp. 565–571. Morgan Kaufmann (1992)
8. Delgrande, J.P.: A first-order logic for prototypical properties. Art. Intel. 33, 105–130 (1987)
9. Gärdenfors, P., Makinson, D.: Nonmonotonic inference based on expectations. Artificial Intelligence 65(2), 197–245 (1994)
10. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A minimal model semantics for rational closure. In: 14th International Workshop on Nonmonotonic Reasoning, NMR (2012)
11. Giordano, L., Olivetti, N., Gliozzi, V., Pozzato, G.L.: $\mathcal{ALC} + T$: a preferential extension of description logics. Fundamenta Informaticae 96(3), 341–372 (2009)
12. Katsuno, H., Mendelzon, A.: Propositional knowledge base revision and minimal change. Artificial Intelligence 3(52), 263–294 (1991)
13. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44, 167–207 (1990)
14. Lehmann, D., Magidor, M.: What does a conditional knowledge base entail? Artificial Intelligence 55, 1–60 (1992)

The Complexity of One-Agent Refinement Modal Logic

Laura Bozzelli[1], Hans van Ditmarsch[2], and Sophie Pinchinat[3]

[1] Technical University of Madrid (UPM), Madrid, Spain
laura.bozzelli@fi.upm.es
[2] Logic, University of Sevilla, Spain & IMSc, Chennai, India
hvd@us.es
[3] IRISA/INRIA, University of Rennes
Sophie.Pinchinat@irisa.fr

Abstract. We investigate the complexity of satisfiability for one-agent Refinement Modal Logic (RML), a known extension of basic modal logic (ML) obtained by adding refinement quantifiers on structures. It is known that RML has the same expressiveness as ML, but the translation of RML into ML is of non-elementary complexity, and RML is at least *doubly* exponentially more succinct than ML. In this paper, we show that RML-satisfiability is 'only' *singly* exponentially harder than ML-satisfiability, the latter being a well-known PSPACE-complete problem. More precisely, we establish that RML-satisfiability is complete for the complexity class $\mathsf{AEXP}_{\mathsf{pol}}$, i.e., the class of problems solvable by alternating Turing machines running in single exponential time but only with a polynomial number of alternations (note that $\mathsf{NEXPTIME} \subseteq \mathsf{AEXP}_{\mathsf{pol}} \subseteq \mathsf{EXPSPACE}$).

1 Introduction

From propositional to refinement quantification in modal logics. Modal logics with propositional quantifiers have been investigated since Fine's seminal paper [7]. Fine distinguishes different propositional quantifications, which allow different kinds of model transformations, not all of which are, in our modern terms, bisimulation preserving. However, in the general case, propositional quantification can easily lead to undecidable logics [7,8]. This has motivated, more recently, the introduction of bisimulation quantified logics [19,10,8,15]: in this framework, the quantification is over the models which are bisimilar to the current model except for a propositional variable p (note that this includes model restriction). A novel way of quantification in rather dynamic modal logics is quantifying over all modally definable submodels of a given model [2]. The setting for these logics is how to quantify over information change; for example, in the logic APAL of [2], an expression that we might write as $\exists_r \varphi$ for our purposes stands for "there is a formula ψ such that after model restriction with relativization to ψ, the formula φ holds". *Refinement modal logic* (see [17,18] and the unpublished manuscript [4]) is a generalization of this perspective to more complex model transformations than mere model restrictions: this is achieved by existential and universal quantifiers which range over the *refinements* of the current structure (model). From the *atoms/forth/back* requirements of bisimulation, a refinement of a modal structure need only satisfy *atoms* and *back*. It is the dual of a simulation that need only satisfy *atoms* and *forth*. Refinement is more general than model restriction, since it is equivalent to bisimulation followed by model restriction.

Moreover, refinement quantification corresponds to implicit quantification over propositional variables (i.e., quantification over variables not occurring in the formula bound by the quantifier), just as in bisimulation quantified logics we have explicit quantification over propositional variables; in fact, it is an abstraction of a bisimulation quantification followed by a relativization [4]. As amply illustrated in [4], refinement quantification has applications in many settings: in logics for games [1,15], it may correspond to a player discarding some moves; for program logics [9], it may correspond to operational refinement; and for logics for spatial reasoning, it may correspond to sub-space projections [14].

Our Contribution. We focus on complexity issues for (one-agent) Refinement Modal Logic (RML) [17,18,4], the extension of (one-agent) basic modal logic (ML) obtained by adding the existential and universal refinement quantifiers $\exists_r$ and $\forall_r$. It is known [18,4] that RML has the same expressivity as ML, but the translation of RML into ML is of non-elementary complexity and no elementary upper bound is known for its satisfiability problem [4]. In fact, an upper bound in 2EXPTIME has been claimed in [18] by a tableaux-based procedure: the authors later concluded that the procedure is sound but not complete [4]. In this paper, our aim is to close that gap. We also investigate the complexity of satisfiability for some equi-expressive fragments of RML. In particular, we associate with each RML formula φ a parameter $\Upsilon_w(\varphi)$ corresponding to a slight variant of the classical quantifier alternation depth (measured w.r.t. $\exists_r$ and $\forall_r$), and for each $k \geq 1$, we consider the fragment RML^k consisting of the RML formulas φ such that $\Upsilon_w(\varphi) \leq k$. Moreover, we consider the existential (resp., universal) fragment $\mathsf{RML}^{\exists}$ (resp., $\mathsf{RML}^{\forall}$) obtained by disallowing the universal (resp., existential) refinement quantifier.

In order to present our results, first, we recall some computational complexity classes. We assume familiarity with the standard notions of complexity theory [11,13]. We will make use of the levels Σ_k^{EXP} ($k \geq 1$) of the exponential-time hierarchy EH, which are defined similarly to the levels Σ_k^{P} of the polynomial-time hierarchy PH, but with NP replaced with NEXPTIME. In particular, Σ_k^{EXP} corresponds to the class of problems decided by single exponential-time bounded Alternating Turing Machines (ATM, for short) with at most $k-1$ alternations and where the initial state is existential [11]. Note that Σ_1^{EXP} = NEXPTIME. Recall that EH $\subseteq$ EXPSPACE and EXPSPACE corresponds to the class of problems decided by single exponential-time bounded ATM (with no constraint on the number of alternations) [5]. We are also interested in an intermediate class between EH and EXPSPACE, here denoted by $\mathsf{AEXP}_{\mathsf{pol}}$, that captures the precise complexity of some relevant problems [6,11,16] such as the first-order theory of real addition with order [6,11]. Formally, $\mathsf{AEXP}_{\mathsf{pol}}$ is the class of problems solvable by single exponential-time bounded ATM with a polynomial-bounded number of alternations.

Our complexity results are summarized in Figure 1 where we also recall the well-known complexity of ML-satisfiability. For the upper bounds, the (technically non-trivial) main step in the proposed approach exploits a "small" size model property: we establish that like basic modal logic ML, RML enjoys a single exponential size model property.[1]

We conclude this section by observing that our results are surprising for the following reason. While our results essentially indicate that satisfiability of RML is "only" *singly* exponentially harder than satisfiability of ML, it is known [4] that RML is *doubly* exponentially more succinct than ML.

[1] Omitted proofs can be found in the online extended version with the same title.

ML	$\mathsf{RML}^{\exists} = \mathsf{RML}^{1}$	$\mathsf{RML}^{\forall} \subseteq \mathsf{RML}^{2}$	RML^{k+1} $(k \geq 1)$	RML
PSPACE-complete	$\in$ NEXPTIME PSPACE-hard	$\in \Sigma_2^{\mathsf{EXP}}$ NEXPTIME-hard	$\in \Sigma_{k+1}^{\mathsf{EXP}}$ Σ_k^{EXP}-hard	$\mathsf{AEXP}_{\mathsf{pol}}$-complete

Fig. 1. Complexity results for satisfiability of RML and RML-fragments

2 Preliminaries

In the rest of this section, we fix a finite set P of atomic propositions.

Structures, Tree Structures, and Refinement Preorder. A (*one-agent Kripke*) *structure* (*over* P) is a tuple $M = \langle S, E, V \rangle$, where S is a set of states (or worlds), $E \subseteq S \times S$ is a transition (or accessibility) relation, and $V : S \mapsto 2^P$ is a P-*valuation* assigning to each state s the set of propositions in P which hold at s. For states s and t of M such that $(s,t) \in E$, we say that t is a *successor* of s. A *pointed* structure is a pair (M,s) consisting of a structure M and a designated initial state s of M.

A tree T is a prefix-closed subset of $\mathbb{N}^*$, where $\mathbb{N}$ is the set of natural numbers. The elements of T are called *nodes* and the empty word ε is the *root* of T. For $x \in T$, the set of children (or successors) of x is $\{x \cdot i \in T \mid i \in \mathbb{N}\}$. The *size* $|T|$ of T is the number of T-nodes. A (*rooted*) *tree structure* (over P) is a pair $\langle T, V \rangle$ such that T is a tree and $V : T \mapsto 2^P$ is a P-valuation over T. For $x \in T$, the *tree substructure of* $\langle T, V \rangle$ *rooted at* x is the tree structure $\langle T_x, V_x \rangle$, also denoted by $\langle T, V \rangle_x$, where $T_x = \{y \in \mathbb{N}^* \mid x \cdot y \in T\}$ and $V_x(y) = V(x \cdot y)$ for all $y \in T_x$. Note that a tree structure $\langle T, V \rangle$ corresponds to the pointed structure $(\langle T, E, V \rangle, \varepsilon)$, where $(x,y) \in E$ iff y is a child of x. Moreover, we can associate with any pointed structure (M,s) a tree structure, denoted by $Unw(M,s)$, obtained by unwinding M from s in the usual way.

For two structures $M = \langle S, E, V \rangle$ and $M' = \langle S', E', V' \rangle$, a *refinement from* M *to* M' is a relation $\mathfrak{R} \subseteq S \times S$ such that for all $(s,s') \in \mathfrak{R}$: (i) $V(s) = V'(s')$, and (ii) if $(s',t') \in E'$ for some $t' \in S'$, then there is some state $t \in S$ such that $(s,t) \in E$ and $(t,t') \in \mathfrak{R}$. If, additionally, the inverse of $\mathfrak{R}$ is a refinement from M' to M, then $\mathfrak{R}$ is a *bisimulation* from M to M'. For states $s \in S$ and $s' \in S'$, (M',s') is a *refinement* of (M,s) (resp., (M,s) and (M',s') are *bisimilar*), written $(M,s) \succcurlyeq (M',s')$ (resp., $(M,s) \approx (M',s')$), if there is a refinement (resp., bisimulation) $\mathfrak{R}$ from M to M' s.t. $(s,s') \in \mathfrak{R}$. Note that $\succcurlyeq$ is a preorder (i.e., reflexive and transitive) and $\approx$ is an equivalence relation over pointed structures.

Remark 1. For each pointed structure (M,s), $(M,s) \approx Unw(M,s)$.

Refinement Modal Logic. We recall the syntax and semantics of one-agent *refinement modal logic* (RML) [18,4], an equally expressive extension of basic modal logic [3] obtained by adding the *existential and universal refinement quantifiers*. For technical convenience, the syntax of RML formulas φ over P is given in *positive form* as:

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond \varphi \mid \Box \varphi \mid \exists_r \varphi \mid \forall_r \varphi$$

where $p \in P$, $\Diamond\varphi$ reads as "possibly φ", $\Box\varphi$ reads as "necessarily φ", and $\exists_r$ and $\forall_r$ are the existential and universal refinement quantifiers. The *dual* $\widetilde{\varphi}$ of a RML formula φ is inductively defined as: $\widetilde{p} = \neg p$, $\widetilde{\neg p} = p$, $\widetilde{\varphi \vee \psi} = \widetilde{\varphi} \wedge \widetilde{\psi}$, $\widetilde{\Diamond\varphi} = \Box\widetilde{\varphi}$, $\widetilde{\Box\varphi} = \Diamond\widetilde{\varphi}$, $\widetilde{\exists_r\varphi} = \forall_r\widetilde{\varphi}$, and $\widetilde{\forall_r\varphi} = \exists_r\widetilde{\varphi}$. We assume a standard DAG representation of a formula, hence, the size $|\varphi|$ of a formula φ is the number of vertices in the associated DAG (corresponding to the number of distinct subformulas of φ). For example $|p \vee p| = 2$ and $|p \vee q| = 3$. RML is interpreted over pointed structures (M,s). The satisfaction relation $(M,s) \models \varphi$ is inductively defined as follows (we omit the clauses for boolean connectives):

$$\begin{array}{lll}
(M,s) \models p & \texttt{iff} & p \in V(s) \text{ where } M = \langle S,E,V\rangle \\
(M,s) \models \Diamond\varphi & \texttt{iff} & \text{for some successor } t \text{ of } s \text{ in } M, (M,t) \models \varphi \\
(M,s) \models \Box\varphi & \texttt{iff} & \text{for all successors } t \text{ of } s \text{ in } M, (M,t) \models \varphi \\
(M,s) \models \exists_r\varphi & \texttt{iff} & \text{for some refinement } (M',s') \text{ of } (M,s), (M',s') \models \varphi \\
(M,s) \models \forall_r\varphi & \texttt{iff} & \text{for all refinements } (M',s') \text{ of } (M,s), (M',s') \models \varphi
\end{array}$$

Note that $(M,s) \models \varphi$ iff $(M,s) \not\models \widetilde{\varphi}$. If $(M,s) \models \varphi$, we say that (M,s) *satisfies* φ, or also that (M,s) is a *model* of φ. A RML formula φ is *satisfiable* if φ admits some model.

Fragments of RML. Let ML be the fragment of RML obtained by disallowing the refinement quantifiers, which corresponds to basic modal logic [3], and $\mathsf{RML}^{\forall}$ and $\mathsf{RML}^{\exists}$ be the fragments of RML obtained by disallowing the existential refinement quantifier and the universal refinement quantifier, respectively. Moreover, we introduce a family $\{\mathsf{RML}^k\}_{k\geq 1}$ of RML-fragments, where RML^k consists of the RML formulas whose *weak refinement quantifier alternation depth* (see Definition 1 below) is at most k.

Definition 1 (Weak Refinement Quantifier Alternation Depth). We first define the *weak alternation length* $\ell(\chi)$ of finite sequences $\chi \in \{\exists_r,\forall_r\}^*$ of refinement quantifiers: $\ell(\varepsilon) = 0$, $\ell(Q) = 1$ for every $Q \in \{\exists_r,\forall_r\}$, and $\ell(QQ'\chi)$ is $\ell(Q'\chi)$ if $Q = Q'$, and $\ell(Q'\chi)+1$ otherwise. For a RML formula φ, let $T(\varphi)$ be the standard tree encoding of φ, where each node is labeled by either a modality, or a boolean connective, or an atomic proposition. The *weak refinement quantifier alternation depth* $\Upsilon_w(\varphi)$ of a RML formula φ is the maximum of the alternation lengths $\ell(\chi)$ where χ is the sequence of refinement quantifiers along a path of $T(\exists_r\varphi)$ (note that we consider $T(\exists_r\varphi)$ and not $T(\varphi)$).

As an example, for $\varphi = (\forall_r\exists_r p) \vee \Box(\exists_r(p \wedge \forall_r q))$, $\Upsilon_w(\varphi) = 3$. Note that $\mathsf{RML}^{\exists} = \mathsf{RML}^1$ and $\mathsf{RML}^{\forall} \subseteq \mathsf{RML}^2$. Moreover, for each RML formula φ, $\Upsilon_w(\forall_r\varphi) = \Upsilon_w(\widetilde{\forall_r\varphi}) + 1$. The following example illustrates the succinctness of $\mathsf{RML}^{\exists}$ w.r.t. ML.

Example 1. For $n \geq 1$, an n-block is a sequence $b_1,\ldots,b_{n+1}$ of $n+1$ bits. The following $\mathsf{RML}^{\exists}$ formula φ_n is satisfied by a tree structure iff there are two paths from the root encoding two n-blocks of the form $b_1,\ldots,b_n,b_{n+1}$ and $b_1,\ldots,b_n,b'_{n+1}$ s.t. $b_{n+1} \neq b'_{n+1}$:

$$\varphi_n := \exists_r\big(\Diamond^{n+1}(0 \wedge \neg 1) \wedge \Diamond^{n+1}(1 \wedge \neg 0) \wedge \bigwedge_{i=1}^{n} \bigvee_{b\in\{0,1\}} \Box^i(b \wedge \neg(1-b))\big)$$

By using the approach in Section 6.2 of [4], one can easily show that any ML formula which is equivalent to φ_n has size singly exponential in n.

Investigated Problems. For each RML-fragment $\mathfrak{F}$, let $\mathsf{SAT}(\mathfrak{F})$ be the set of satisfiable $\mathfrak{F}$ formulas. In this paper, we investigate the complexity of $\mathsf{SAT}(\mathfrak{F})$ for any $\mathfrak{F} \in \{\mathsf{RML},\mathsf{RML}^{\exists},\mathsf{RML}^{\forall},\mathsf{RML}^2,\ldots\}$. Figure 1 depicts our complexity results.

Assumption. Since RML is bisimulation invariant [18,4], by Remark 1, w.l.o.g. we can assume that the semantics of RML is restricted to tree structures.

Since RML and ML are equi-expressive [18,4], we easily obtain the following.

Proposition 1 (Finite Model Property). *Let* φ *be a* RML *formula and* $\langle T,V\rangle$ *be a tree structure satisfying* φ*. Then, there is a* finite refinement $\langle T_r,V_r\rangle$ *of* $\langle T,V\rangle$ *satisfying* φ.

3 Upper Bounds

In this section, we provide the upper bounds illustrated in Figure 1. Our approach consists of two steps. First, in Section 3.1, we show that RML enjoys a singly exponential size model property. Then, by using this result, we show in Section 3.2 that SAT(RML) can be decided by a singly exponential-time bounded ATM whose number of alternations on an input φ is at most $\Upsilon_w(\varphi)-1$ and whose initial state is existential. We fix a finite set P of atomic propositions and consider RML formulas and tree structures over P.

3.1 Exponential Size Model Property

In this section, we prove the following result.

Theorem 1 (Exponential Size Model Property). *For all satisfiable* RML *formulas* φ *and tree structures* $\langle T,V\rangle$ *such that* $\langle T,V\rangle$ *satisfies* φ*, the following holds: there exists a* finite refinement $\langle T',V'\rangle$ *of* $\langle T,V\rangle$ *such that* $\langle T',V'\rangle$ *satisfies* φ *and* $|T'|\leq|\varphi|^{3|\varphi|^2}$.

First, we summarize the main steps in the proof of Theorem 1. Given a RML formula φ, we associate with φ tableaux-based *finite* objects called *constraints systems for* φ (Definition 2). Essentially, a constraint system $\mathcal{S}$ for φ is a tuple of hierarchically ordered finite tree structures which intuitively represents an *extended model* of φ: (1) each node x in a tree structure of $\mathcal{S}$ is additionally labeled by a set of subformulas of φ which hold at the tree substructure rooted at node x, and, in particular, the first tree structure, called *main structure*, represents a model of φ, and (2) the other tree structures of $\mathcal{S}$ are used to manage the $\exists_r$-subformulas of φ. In fact, in order to be an *extended model* of φ, $\mathcal{S}$ has to satisfy additional structural requirements which capture the semantics of the boolean connectives and all the modalities except the universal refinement quantifier $\forall_r$, the latter being only semantically captured. Let $\mathcal{C}(\varphi)$ be the set of these constraints systems for φ, which are said to be *well-formed, saturated, and semantically* $\forall_r$*-consistent.* We individuate a subclass $\mathcal{C}_{min}(\varphi)$ of $\mathcal{C}(\varphi)$ consisting of '*minimal*' constraints systems for φ whose sizes are *singly exponential* in the size of φ, and which can be obtained from φ by applying structural *completion rules* (Definitions 3 and 4). Furthermore, we introduce a notion of 'refinement' between constraint systems for φ (Definition 5) which preserves the semantic $\forall_r$-consistency requirement. Then, given a *finite* tree structure $\langle T,V\rangle$ satisfying φ, we show that: (1) there is a constraint system $\mathcal{S}\in\mathcal{C}(\varphi)$ whose main structure is $\langle T,V\rangle$ (Lemma 1), and (2) starting from $\mathcal{S}$, it is possible to construct a minimal constraint system $\mathcal{S}_{min}\in\mathcal{C}_{min}(\varphi)$ which is a *refinement* of $\mathcal{S}$ (Lemma 3). This entails that the main structure of $\mathcal{S}_{min}$ is a refinement of $\langle T,V\rangle$ satisfying φ and having a single exponential size. Hence, by Proposition 1, Theorem 1 follows. Now, we proceed with the details of the proof of Theorem 1.

We denote by $\overline{P}$ the set of negations of propositions in P, i.e. $\overline{P}=\{\neg p\mid p\in P\}$. A set χ of RML formulas is *complete* if for each $p\in P$, either $p\in\chi$ or $\neg p\in\chi$. In the following, we fix a RML formula φ. The *closure* $\mathsf{cl}(\varphi)$ of φ is the set containing all the subformulas of φ and the formulas in $P\cup\overline{P}$. Moreover, $\mathsf{d}_{\Diamond,\Box}(\varphi)$ denotes the nesting depth of modalities $\Diamond$ and $\Box$ (in φ), and $\mathsf{d}_{\exists}(\varphi)$ denotes the nesting depth of modality $\exists_r$.

Definition 2. A *constraint system* for φ is a tuple $\mathcal{S} = \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$, where for all $1 \leq i \leq n$, T_i is a *finite* tree and L_i and $\leftarrow_i$ are two T_i-labelings such that:

1. for each $x \in T_i$, $L_i(x)$ is a *complete* subset of $\mathsf{cl}(\varphi)$; moreover, $\varphi \in L_i(\varepsilon)$ if $i = 1$;
2. $\leftarrow_i : T_i \mapsto \{\perp\}$ if $i = 1$ ($\perp$ is for undefined), and $\leftarrow_i : T_i \mapsto \{j\} \times T_j$ for some $1 \leq j < i$ otherwise (note that $j < i$); moreover, for $i > 1$ and $x, x' \in T_i$ with $\leftarrow_i(x) = \langle j, y\rangle$ and $\leftarrow_i(x') = \langle j, y'\rangle$, if x' is a successor of x in T_i, then y' is a successor of y in T_j.

We denote by $\widetilde{L}_i$ the P-valuation over T_i defined as $\widetilde{L}_i(x) := L_i(x) \cap P$ for all $x \in T_i$, by $\mathcal{S}(i)$ the ith component of $\mathcal{S}$, i.e. $\langle T_i, L_i, \leftarrow_i\rangle$, and by $\mathsf{dim}(\mathcal{S})$ the number of $\mathcal{S}$ components, i.e., n. The (tree) structure $\langle T_1, \widetilde{L}_1\rangle$ is called the *main structure* of $\mathcal{S}$.

Let $1 \leq i, j \leq n$, $x \in T_i$, $y \in T_j$ and $\psi \in \mathsf{cl}(\varphi)$. We write $\langle j, y\rangle \leftarrow_{\mathcal{S}} \langle i, x\rangle$ to mean that $\leftarrow_i(x) = \langle j, y\rangle$, and $\mathcal{S} \vdash \langle i, x, \psi\rangle$ (resp., $\mathcal{S} \not\vdash \langle i, x, \psi\rangle$) to mean that $\psi \in L_i(x)$ (resp., $\psi \notin L_i(x)$). If $\mathcal{S} \vdash \langle i, x, \psi\rangle$, we say that $\langle i, x, \psi\rangle$ is a *$\mathcal{S}$-constraint*. $\mathcal{S}$ contains a *clash* if $\mathcal{S} \vdash \langle i, x, p\rangle$ and $\mathcal{S} \vdash \langle i, x, \neg p\rangle$ for some $1 \leq i \leq n$, $x \in T_i$, and $p \in P$. Otherwise, $\mathcal{S}$ is called *clash-free*. Moreover, $\mathcal{S}$ is said to be *well-formed* if $\mathcal{S}$ is clash-free and whenever $\langle j, y\rangle \leftarrow_{\mathcal{S}} \langle i, x\rangle$, then $\widetilde{L}_j(y) = \widetilde{L}_i(x)$. Furthermore, $\mathcal{S}$ is said to be *semantically $\forall_r$-consistent* if whenever $\mathcal{S} \vdash (i, x, \forall_r\psi)$ then the tree structure $\langle T_i, \widetilde{L}_i\rangle_x$ satisfies $\forall_r\psi$.

If $\mathcal{S}$ is well-formed, then the labelings $\leftarrow_i$ induce a refinement hierarchy:

Remark 2. Let $\mathcal{S} = \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$ be a *well-formed* constraint system for φ. Then, $\langle j, y\rangle \leftarrow_{\mathcal{S}} \langle i, x\rangle$ implies that $\langle T_i, \widetilde{L}_i\rangle_x$ is a refinement of $\langle T_j, \widetilde{L}_j\rangle_y$.

Definition 3 (Saturated Constraint Systems). A constraint system $\mathcal{S}$ for φ is *saturated* if none of the following *completion rules* are applicable to $\mathcal{S}$.

```
∧-rule:  if $\mathcal{S} \vdash \langle i, x, \psi_1 \wedge \psi_2\rangle$, $\mathcal{S}(i) = \langle T, L, \leftarrow\rangle$, and $\{\psi_1, \psi_2\} \not\subseteq L(x)$
         then update $L(x) := L(x) \cup \{\psi_1, \psi_2\}$
∨-rule:  if $\mathcal{S} \vdash \langle i, x, \psi_1 \vee \psi_2\rangle$, $\mathcal{S}(i) = \langle T, L, \leftarrow\rangle$, and $\{\psi_1, \psi_2\} \cap L(x) = \emptyset$
         then update $L(x) := L(x) \cup \{\psi_k\}$ for some $k \in \{1, 2\}$
∃_r-rule: if $\mathcal{S} \vdash \langle i, x, \exists_r\psi\rangle$, $\mathcal{S} := \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_n, L_n, \leftarrow_n\rangle\rangle$, and
              $\mathcal{S} \not\vdash \langle h, \varepsilon, \psi\rangle$ for each $h \leq \mathsf{dim}(\mathcal{S})$ such that $\leftarrow_h(\varepsilon) = \langle i, x\rangle$
          then update $\mathcal{S} := \langle\langle T_1, L_1, \leftarrow_1\rangle, \ldots, \langle T_{n+1}, L_{n+1}, \leftarrow_{n+1}\rangle\rangle$, where
              $T_{n+1} := \{\varepsilon\}$, $L_{n+1}(\varepsilon) := \{\psi\} \cup (L_i(x) \cap (P \cup \overline{P}))$, and $\leftarrow_{n+1}(\varepsilon) := \langle i, x\rangle$
□-rule:  if $\mathcal{S} \vdash \langle i, x, \Box\psi\rangle$ and $\mathcal{S} \not\vdash \langle i, x', \psi\rangle$ for some successor $x'$ of $x$ in $\mathcal{S}(i)$
         then let $\mathcal{S}(i) = \langle T, L, \leftarrow\rangle$
              update $L(x') := L(x') \cup \{\psi\}$ for each successor $x'$ of $x$ in $T$
◇-rule:  if $\mathcal{S} \vdash \langle i, x, \Diamond\psi\rangle$ and $\mathcal{S} \not\vdash \langle i, x', \psi\rangle$ for each successor $x'$ of $x$ in $\mathcal{S}(i)$
         then let $\langle i_0, x_0\rangle \leftarrow_{\mathcal{S}} \ldots \leftarrow_{\mathcal{S}} \langle i_k, x_k\rangle$ with $i_0 = 1$ and $\langle i_k, x_k\rangle = \langle i, x\rangle$
              guess some complete set $\chi \subseteq P \cup \overline{P}$,
              for each $q = k, k-1, \ldots, 0$ with $\mathcal{S}(i_q) = \langle T_q, L_q, \leftarrow_q\rangle$ do
                  update $T_q := T_q \cup \{x_q \cdot h_q\}$ for some $h_q \in \mathbb{N}$ such that $x_q \cdot h_q \notin T_q$
                  if $q < k$ then $L_q(x_q \cdot h_q) := \chi$ and $\leftarrow_{i_{q+1}}(x_{q+1} \cdot h_{q+1}) := \langle i_q, x_q \cdot h_q\rangle$
                               else $L_q(x_q \cdot h_q) := \{\psi\} \cup \chi$
```

Remark 3. Let $\mathcal{S}$ be a constraint system for φ. Then, applying any rule of Definition 3 to $\mathcal{S}$ yields a constraint system for φ.

The $\vee$-rule, the $\wedge$-rule, and the $\Box$-rule of Definition 3 are standard. The $\exists_r$-rule and the $\Diamond$-rule are the unique rules which add new nodes to the given constraint system $\mathcal{S}$ for φ. The $\exists_r$-rule is applicable to a $\mathcal{S}$-constraint $\xi = \langle i, x, \exists_r \psi \rangle$ if there are no $\mathcal{S}$-constraints (ξ-witnesses) of the form $\langle h, \varepsilon, \psi \rangle$ such that $\langle i, x \rangle \leftarrow_{\mathcal{S}} \langle h, \varepsilon \rangle$. The rule then adds a ξ-witness $\langle n+1, \varepsilon, \psi \rangle$ to $\mathcal{S}$ by extending $\mathcal{S}$ with a new component containing a single node (the root) whose label is propositionally consistent with the label of x. The $\Diamond$-rule is applicable to a $\mathcal{S}$-constraint $\xi = \langle i, x, \Diamond \psi \rangle$ if there are no $\mathcal{S}$-constraints (ξ-witnesses) of the form $\langle i, x', \psi \rangle$ where x' is a successor of x. Let $\langle i_0, x_0 \rangle \leftarrow_{\mathcal{S}} \ldots \leftarrow_{\mathcal{S}} \langle i_k, x_k \rangle$ be the maximal chain of 'backward links' from $\langle i_k, x_k \rangle = \langle i, x \rangle$. The rule then adds a ξ-witness $\langle i_k, x'_k, \psi \rangle$ to $\mathcal{S}$ (x'_k being a new successor of $x_k = x$), a complete set $\chi \subseteq P \cup \overline{P}$ is guessed, and the hierarchical structure of $\mathcal{S}$ is restored as follows: the rule adds the new constraints $\langle i_0, x'_0, \chi \rangle, \ldots, \langle i_k, x'_k, \chi \rangle$, where $x'_0, \ldots, x'_{k-1}$ are new successors of $x_0, \ldots, x_{k-1}$ respectively, and the new chain of 'backward links' $\langle i_0, x'_0 \rangle \leftarrow_{\mathcal{S}} \ldots \leftarrow_{\mathcal{S}} \langle i_k, x'_k \rangle$.

Lemma 1 (Soundness & Completeness). *Let $\langle T, V \rangle$ be a finite tree structure. Then, $\langle T, V \rangle$ satisfies φ* if and only if *there is a well-formed, saturated, and semantically $\forall_r$-consistent constraint system $\mathcal{S}$ for φ whose main structure is $\langle T, V \rangle$.*

Definition 4 (Minimal Constraint Systems). A constraint system $\mathcal{S}$ for φ is *initial* if $\mathcal{S} = \langle \langle \{\varepsilon\}, L, \leftarrow \rangle \rangle$, and for all $\psi \in L(\varepsilon)$, either $\psi = \varphi$ or $\psi \in P \cup \overline{P}$. A *minimal* constraint system $\mathcal{S}$ for φ is a constraint system for φ which can be obtained from some *initial* constraint system for φ by a sequence of applications of the rules of Definition 3.

The following lemma shows that every *minimal* constraint system for φ has a 'size' singly exponential in the size of φ.

Lemma 2. *Each* minimal *constraint system $\mathcal{S}$ for* φ *satisfies the following invariant: (i) each tree in $\mathcal{S}$ has height at most $\mathsf{d}_{\Diamond,\Box}(\varphi)$ and branching degree at most $|\varphi|^{(2\mathsf{d}_{\exists}(\varphi)+1)}$, and (ii) $\mathsf{dim}(\mathcal{S})$ is at most $|\varphi|^{4\cdot(\mathsf{d}_{\exists}(\varphi))^2\cdot(\mathsf{d}_{\Diamond,\Box}(\varphi)+1)}$. Moreover, any sequence of applications of the rules of Definition 3 starting from an* initial *constraint system for φ is finite.*

We introduce a notion of 'refinement' over constraint systems for φ, which generalizes the refinement preorder over finite structures. Moreover, this notion crucially preserves both well-formedness and the semantic $\forall_r$-consistency requirement (Lemma 3).

Definition 5 (Refinement for constraint systems). *Let $\mathcal{S} = \langle \langle T_1, L_1, \leftarrow_1 \rangle, \ldots, \langle T_n, L_n, \leftarrow_n \rangle \rangle$ and $\mathcal{S}' = \langle \langle T'_1, L'_1, \leftarrow'_1 \rangle, \ldots, \langle T'_m, L'_m, \leftarrow'_m \rangle \rangle$ be constraint systems for φ. $\mathcal{S}$ is a* refinement *of $\mathcal{S}'$ if there is a tuple $\mathcal{T} = \langle \uparrow_1, \ldots, \uparrow_n \rangle$ such that for all $1 \leq i \leq n$, there is $1 \leq j \leq m$ so that $\uparrow_i \colon T_i \mapsto T'_j$ and for all $x \in T_i$ with $\uparrow_i(x) = y$:*

1. *$L_i(x) \subseteq L'_j(y)$, $j = 1$ iff $i = 1$, and $y = \varepsilon$ iff $x = \varepsilon$;*
2. *for each successor x' of x in T_i, $\uparrow_i(x') = y'$ where y' is a successor of y in T'_j;*
3. *if $\leftarrow_i(x) = \langle i', x' \rangle$, then $\leftarrow'_j(y) = \langle j', y' \rangle$ and $\uparrow_{i'}(x') = y'$ (in particular, $\uparrow_{i'} \colon T_{i'} \mapsto T'_{j'}$).*

Lemma 3 (Minimalization). *Let $\mathcal{S}'$ be a constraint system for φ which is well-formed and semantically $\forall_r$-consistent. Then, any constraint system $\mathcal{S}$ for φ which is a refinement of $\mathcal{S}'$ is well-formed and semantically $\forall_r$-consistent too, and the main structure of $\mathcal{S}$ is a refinement of the main structure of $\mathcal{S}'$. Moreover, if $\mathcal{S}'$ is additionally saturated, there is a* minimal *and saturated constraint system $\mathcal{S}$ for φ which is a refinement of $\mathcal{S}'$.*

Sketched Proof. The first part of Lemma 3 follows from Definition 5 and the following crucial observation: if $\langle T_r, V_r\rangle$ is a refinement of a tree structure $\langle T, V\rangle$, then for each $\forall_r$-formula $\forall_r\psi$, $\langle T, V\rangle \models \forall_r\psi$ implies $\langle T_r, V_r\rangle \models \forall_r\psi$. For the second part of Lemma 3, let $\mathcal{S}'$ be a well-formed, saturated, and semantically $\forall_r$-consistent constraint system for φ. Since $\mathcal{S}'$ is well-formed, there is a unique *initial* constraint system $\mathcal{S}_0$ for φ s.t. $\mathcal{S}_0$ is a refinement of $\mathcal{S}'$. For each rule of Definition 3, we define an extension of such a rule which has the same *precondition* and the same *effect* (w.r.t. a given constraint system $\mathcal{S}$) with the difference that the nondeterministic choices are guided by $\mathcal{S}'$. By Lemma 2, it follows that any sequence of applications of the *new* rules starting from $\mathcal{S}_0$ is *finite*. Moreover, the application of these new rules preserves the property of a constraint system to be a refinement of $\mathcal{S}'$. Hence, we deduce that there is a *minimal* and saturated constraint system $\mathcal{S}$ for φ which is a refinement of $\mathcal{S}'$. $\square$

Proof of Theorem 1. Let $\langle T, V\rangle$ be a tree structure satisfying φ. By Proposition 1, there is a finite refinement $\langle T_r, V_r\rangle$ of $\langle T, V\rangle$ satisfying φ. By Lemma 1, there is a well-formed, saturated, and semantically $\forall_r$-consistent constraint system $\mathcal{S}$ for φ whose main structure is $\langle T_r, V_r\rangle$. Thus, by Lemma 3, there is a *minimal*, well-formed, saturated, and semantically $\forall_r$-consistent constraint system $\mathcal{S}_{min}$ for φ whose main structure $\langle T_{min}, V_{min}\rangle$ is a refinement of $\langle T_r, V_r\rangle$. Hence, $\langle T_{min}, V_{min}\rangle$ is a refinement of $\langle T, V\rangle$ as well, and by Lemmata 1 and 2, $\langle T_{min}, V_{min}\rangle$ satisfies φ and $|T_{min}| \leq |\varphi|^{3|\varphi|^2}$. Hence, the result follows.

3.2 Checking Satisfiability

For a RML formula, the set $\mathsf{FL}(\varphi)$ of *first-level* subformulas of φ is defined as follows: if $\varphi = \varphi_1 \vee \varphi_2$ or $\varphi = \varphi_1 \wedge \varphi_2$, then $\mathsf{FL}(\varphi) = \mathsf{FL}(\varphi_1) \cup \mathsf{FL}(\varphi_2)$; otherwise $\mathsf{FL}(\varphi) = \{\varphi\}$.

Theorem 2. *$SAT(RML) \in AEXP_{pol}$ and $SAT(RML^k) \in \Sigma_k^{EXP}$ for each $k \geq 1$.*

Proof. For a RML formula φ, a *certificate* of φ is a finite tree structure $\langle T, V\rangle$ such that $|T| \leq |\varphi|^{3\cdot|\varphi|^2}$. Define:

$$\widehat{\mathsf{SAT}}(\mathsf{RML}) := \{(\varphi, \langle T, V\rangle) \mid \varphi \in \mathsf{RML} \textit{ and } \langle T, V\rangle \text{ is a certificate of } \varphi \text{ satisfying } \varphi\}$$

By Theorem 1, $\varphi \in \mathsf{SAT}(\mathsf{RML})$ iff $(\varphi, \langle T, V\rangle) \in \widehat{\mathsf{SAT}}(\mathsf{RML})$ for some certificate $\langle T, V\rangle$ of φ. Since $\langle T, V\rangle$ has size singly exponential in the size of φ, it suffices to show that $\widehat{\mathsf{SAT}}(\mathsf{RML})$ can be decided by a *polynomial-time* bounded ATM whose number of alternations on an input $(\varphi, \langle T, V\rangle)$ is at most $\Upsilon_w(\varphi) - 1$ and whose initial state is existential. For this, in turn, we show that $\widehat{\mathsf{SAT}}(\mathsf{RML})$ can be decided by a *nondeterministic polynomial-time bounded* procedure "*check*" that given an input $(\varphi, \langle T, V\rangle)$, uses in case $\Upsilon_w(\varphi) > 1$ as an *oracle* the same language $\widehat{\mathsf{SAT}}(\mathsf{RML})$ but with input queries of the form $(\psi, \langle T', V'\rangle)$, where $\Upsilon_w(\psi) < \Upsilon_w(\varphi)$ and $\psi \in \mathsf{cl}(\varphi)$. Hence, by standard arguments in complexity theory [11,13], the result follows. Procedure *check* is defined as follows.

```
check($\varphi, \langle T, V\rangle$)        /** $\varphi \in$ RML and $\langle T, V\rangle$ is a certificate of $\varphi$ **/
  $\mathcal{K} \leftarrow \{(\varphi, \langle T, V\rangle)\}$;
  while $\mathcal{K} \neq \emptyset$ do
    select $(\psi, \langle T', V'\rangle) \in \mathcal{K}$; update $\mathcal{K} \leftarrow \mathcal{K} \setminus \{(\psi, \langle T', V'\rangle)\}$;
    guess $\mathcal{F} \subseteq$ FL($\psi$) and let $\psi_{\mathcal{F}}$ be the boolean formula obtained from $\psi$ by replacing
```

```
        each first-level subformula $\theta$ of $\psi$ with true if $\theta \in \mathcal{F}$, and false otherwise;
        if $\psi_{\mathcal{F}}$ evaluates to false then *reject the input*;
        for each $\theta \in \mathcal{F}$ do
            case $\theta = p$ with $p \in P$: if $p \notin V'(\varepsilon)$ then *reject the input*;
            case $\theta = \neg p$ with $p \in P$: if $p \in V'(\varepsilon)$ then *reject the input*;
            case $\theta = \Diamond\theta'$: guess a child $x$ of the $T'$-root, update $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\theta', \langle T', V'\rangle_x)\}$;
            case $\theta = \Box\theta'$: update $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\theta', \langle T', V'\rangle_{x_1}), \ldots, (\theta', \langle T', V'\rangle_{x_k})\}$
                                where $x_1, \ldots, x_k$ are the children of the root of $T'$;
            case $\theta = \exists_r\theta'$: guess a *certificate* $\langle T'', V''\rangle$ of $\theta'$ which is a refinement of
                                $\langle T', V'\rangle$ and update $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\theta', \langle T'', V''\rangle)\}$;[2]
            case $\theta = \forall_r\theta'$: query the oracle for $\widehat{\mathsf{SAT}}(\mathsf{RML})$ with input $(\widetilde{\forall_r\theta'}, \langle T', V'\rangle)$;
                                 /** note that $\Upsilon_w(\widetilde{\forall_r\theta'}) < \Upsilon_w(\forall_r\theta') \leq \Upsilon_w(\varphi)$ **/
                                 if the oracle answers *YES* then *reject the input*;
        end for
    end while
    *accept the input.*
```

Correctness of the procedure *check* easily follows from Theorem 1. □

4 Lower Bounds

In this section, we provide the lower bounds illustrated in Figure 1. The main contribution is $\mathsf{AEXP}_{\mathsf{pol}}$-hardness of $\mathsf{SAT}(\mathsf{RML})$, which is proved by a polynomial-time reduction from a suitable $\mathsf{AEXP}_{\mathsf{pol}}$-complete problem. First, we define this problem.

Let $k \geq 1$. A *k-ary deterministic Turing Machine* is a deterministic Turing machine $\mathcal{M} = \langle k, I, A, Q, \{q_{acc}, q_{rej}\}, q_0, \delta\rangle$ operating on k ordered semi-infinite tapes and having just one read/write head, where: I (resp., $A \supset I$) is the input (resp., work) alphabet, A contains the blank symbol #, Q is the set of states, q_{acc} (resp., q_{rej}) is the terminal accepting (resp., rejecting) state, q_0 is the initial state, and $\delta : (Q \setminus \{q_{acc}, q_{rej}\}) \times A \rightarrow (Q \times A \times \{-1, +1\}) \cup \{1, \ldots, k\}$ is the transition function. In each non-terminal step, if the read/write head scans a cell of the ℓth tape ($1 \leq \ell \leq k$) and $(q, a) \in (Q \setminus \{q_{acc}, q_{rej}\}) \times A$ is the current pair state/ scanned cell content, the following occurs:

- $\delta(q, a) \in Q \times A \times \{-1, +1\}$ (*ordinary moves*): $\mathcal{M}$ overwrites the tape cell being scanned, there is a change of state, and the read/write head moves one position to the left (-1) or right (+1) in accordance with $\delta(q, a)$.
- $\delta(q, a) = h \in \{1, \ldots, k\}$ (*jump moves*): the read/write head jumps to the left-most cell of the hth tape and the state remains unchanged.

$\mathcal{M}$ *accepts* a k-ary input $(w_1, \ldots, w_k) \in (I^*)^k$, written $\mathcal{M}(w_1, \ldots, w_k)$, if the computation of $\mathcal{M}$ from $(w_1, \ldots, w_k)$ (initially, the ℓth tape contains the word w_ℓ, and the head points to the left-most cell of the first tape) is accepting. We consider the following problem.

[2]By Theorem 1, if there is a refinement of $\langle T', V'\rangle$ which satisfies θ', there is also a refinement of $\langle T', V'\rangle$ satisfying θ' which is a certificate of θ'. Moreover, checking for two given *finite* tree structures $\langle T, V\rangle$ and $\langle T', V'\rangle$, whether $\langle T, V\rangle$ is a refinement of $\langle T', V'\rangle$ (or, equivalently, $\langle T', V'\rangle$ is a simulation of $\langle T, V\rangle$) can be done in polynomial time (see, e.g., [12]).

Alternation Problem. An instance of the problem is a triple $(k,n,\mathcal{M})$, where $k \geq 1$, $n > 1$, and $\mathcal{M}$ is a *polynomial-time bounded* k-ary deterministic Turing Machine with input alphabet I. The instance $(k,n,\mathcal{M})$ is *positive* iff the following holds, where $\mathsf{Q}_\ell = \exists$ if ℓ is odd, and $\mathsf{Q}_\ell = \forall$ otherwise (for all $1 \leq \ell \leq k$),

$$\mathsf{Q}_1 x_1 \in I^{2^n}.\mathsf{Q}_2 x_2 \in I^{2^n}.\ldots \mathsf{Q}_k x_k \in I^{2^n}.\,\mathcal{M}(x_1,\ldots,x_k)$$

Note that the quantifications Q_i are restricted to words over I of length 2^n.

For $k \geq 1$, the *k-Alternation Problem* is the Alternation Problem restricted to instances of the form $(k,n,\mathcal{M})$ (i.e., the first input parameter is fixed to k), and the *Linear Alternation Problem* is the Alternation Problem restricted to instances of the form $(n,n,\mathcal{M})$.

Proposition 2. *The Linear Alternation Problem is $\mathsf{AEXP_{pol}}$-complete and for all $k \geq 1$, the k-Alternation Problem is Σ_k^{EXP}-complete.*

The proof of Proposition 2 is standard. Fix an instance $(k,n,\mathcal{M})$ of the Alternation Problem with $\mathcal{M} = \langle k,I,A,Q,\{q_{acc},q_{rej}\},q_0,\delta\rangle$. Since $\mathcal{M}$ is polynomial-time bounded, there is an integer constant $c \geq 1$ such that when started on a k-ary input $(w_1,\ldots,w_k)$, $\mathcal{M}$ reaches a terminal configuration in at most $(|w_1|+\ldots+|w_k|)^c$ steps. A (k,n)-input is a k-ary input $(w_1,\ldots,w_k)$ such that $w_i \in I^{2^n}$ for all $1 \leq i \leq k$. Let $\mathsf{c}(k,n) := c\cdot(n+\lceil\log k\rceil)$, where $\lceil\log k\rceil$ denotes the smallest $i \in \mathbb{N}$ such that $i \geq \log k$. Note that a configuration of $\mathcal{M}$ reachable from a (k,n)-input, called (k,n)*-configuration*, can be described as a tuple $\overrightarrow{C} = (C_1,\ldots,C_k)$ of k words $C_1,\ldots,C_k$ over $A \cup (Q \times A)$ of length exactly $2^{\mathsf{c}(k,n)}$ such that for some $1 \leq \ell \leq k$, C_ℓ is of the form $w\cdot(q,a)\cdot w' \in A^* \times (Q\times A)\times A^*$, and for $i \neq \ell$, $C_i \in A^{2^{\mathsf{c}(k,n)}}$. For a (k,n)-input $(a\cdot w_1,\ldots,w_k)$, the associated *initial* (k,n)*-configuration* is $((q_0,a)\cdot w_1\cdot \#^{2^{\mathsf{c}(k,n)}-2^n},\ldots,w_k\cdot\#^{2^{\mathsf{c}(k,n)}-2^n})$. Thus, the computations of $\mathcal{M}$ from (k,n)-inputs, called (k,n)*-computations*, can be described by sequences π of at most $2^{\mathsf{c}(k,n)}$ (k,n)-configurations. In fact, w.l.o.g., we can assume that π has length *exactly* $2^{\mathsf{c}(k,n)}$. In the rest of this section, we prove the following result.

Theorem 3. *One can construct a RML^{k+1} formula φ in time polynomial in n, k, and the size of the TM $\mathcal{M}$ such that (i) φ is a $\mathsf{RML}^\forall$ formula if $k = 1$, and (ii) φ is satisfiable* if and only if *the instance $(k,n,\mathcal{M})$ of the Alternation Problem is positive.*

By Proposition 2 and Theorem 3, we obtain the following.

Corollary 1. *$\mathsf{SAT}(\mathsf{RML})$ is $\mathsf{AEXP_{pol}}$-hard, $\mathsf{SAT}(\mathsf{RML}^\forall)$ is $\mathsf{NEXPTIME}$-hard, and for all $k \geq 1$, $\mathsf{SAT}(\mathsf{RML}^{k+1})$ is Σ_k^{EXP}-hard.*

Tree Encoding of (k,n)-Computations. In order to prove Theorem 3, first, we define an encoding of (k,n)-computations by suitable tree structures over P, where P is given by

$$P = \{0,1,arg_1,\ldots,arg_k\} \cup \Lambda$$

and Λ is the set of triples (u_-,u,u_+) s.t. $u \in A \cup (Q \times A)$ and $u_-,u_+ \in A \cup (Q\times A) \cup \{\bot\}$ ($\bot$ is for undefined). An *extended TM block ext_bl* is a word over 2^P of length $2\mathsf{c}(k,n)+2$ of the form $ext\_bl = \{bit_1\}\cdot\ldots\cdot\{bit_{\mathsf{c}(k,n)}\}\cdot bl$, where bl, called *TM block*, is of the form $bl = \{bit'_1\}\cdot\ldots\cdot\{bit'_{\mathsf{c}(k,n)}\}\cdot\{arg_\ell\}\cdot\{t\}$ with $1 \leq \ell \leq k$ and $t \in \Lambda$. The *content* $\mathsf{CON}(ext\_bl)$ (resp., $\mathsf{CON}(bl)$) of *ext_bl* (resp., *bl*) is *bl* (resp., *t*), the *component number* of *ext_bl* and *bl* is ℓ, and the *position number* of *ext_bl* (resp., *bl*) is the integer in

$[0, 2^{\mathsf{c}(k,n)} - 1]$ whose binary code is $bit_1, \ldots, bit_{\mathsf{c}(k,n)}$ (resp., $bit'_1, \ldots, bit'_{\mathsf{c}(k,n)}$). Intuitively, *ext_bl* encodes the triple $t = (C_\ell(i-1), C_\ell(i), C_\ell(i))$ with $i = \mathsf{ID}(bl)$ (where $C_\ell(i-1) = \perp$ if $i = 0$, and $C_\ell(i+1) = \perp$ if $i = 2^{\mathsf{c}(k,n)} - 1$) of the ℓth component C_ℓ of some (k,n)-configuration, the latter being the $(\mathsf{ID}(ext\_bl))$-th (k,n)-configuration of some (k,n)-computation. For a sequence $\pi = \overrightarrow{C}_0, \ldots, \overrightarrow{C}_{2^{\mathsf{c}(k,n)}-1}$ of $2^{\mathsf{c}(k,n)}$ (k,n)-configurations, we can encode π by the set $S_{ext\_bl}(\pi)$ of extended blocks defined as: $ext\_bl \in S_{ext\_bl}(\pi)$ iff there are $0 \leq i, j \leq 2^{\mathsf{c}(k,n)} - 1$ and $0 \leq \ell \leq k$ such that $\mathsf{ID}(ext\_bl) = i$, $\mathsf{CON}(ext\_bl) = bl$, $\mathsf{ID}(bl) = j$ and bl is the TM block associated with the jth symbol of the ℓth component of $\overrightarrow{C}_i$. The tree representation of the set $S_{ext\_bl}(\pi)$ is defined as follows.

Definition 6. A (k,n)*-computation tree code is* a tree structure $\langle T, V \rangle$ over P such that:

1. Each path of $\langle T, V \rangle$ from the root has length $2\mathsf{c}(k,n) + 2$, and each node is labeled exactly by a proposition in P. Moreover, $\langle T, V \rangle$ satisfies the ML-formula
$$\bigwedge_{i=1}^{2\mathsf{c}(k,n)} \Box^{i-1}((\Diamond 0 \wedge \Diamond 1) \wedge \Box(0 \vee 1)) \wedge \Box^{2\mathsf{c}(k,n)}(\bigwedge_{\ell=1}^{k} \Diamond arg_\ell \wedge \Box \bigvee_{\ell=1}^{k} arg_\ell) \wedge \Box^{2\mathsf{c}(k,n)+2} \bigvee_{t \in \Lambda} t$$
This requirement implies, in particular, that each path ν of $\langle T, V \rangle$ from the root is labeled by a word of the form $V(\varepsilon) \cdot ext\_bl$, where *ext_bl* is an extended TM block.
2. There is a sequence $\pi = \overrightarrow{C}_0, \ldots, \overrightarrow{C}_{2^{\mathsf{c}(k,n)}-1}$ of $2^{\mathsf{c}(k,n)}$ (k,n)-configurations such that the set of extended TM blocks of $\langle T, V \rangle$ corresponds to the set $S_{ext\_bl}(\pi)$.

We also need to encode existential and universal quantification on the different components of a (k,n)-input of the TM $\mathcal{M}$. This leads to the following definition.

Definition 7 (Initialized full (k,n)-computation tree codes). Let $1 \leq \ell \leq k$. A ℓ*-initialized full* (k,n)*-computation tree code* is a tree structure $\langle T, V \rangle$ over P such that:

1. *Fullness requirement.* $\langle T, V \rangle$ satisfies Property 1 of Definition 6. Moreover, let $\nu = z_0, \ldots, z_{2c(n)+1}, z_{2c(n)+2}$ be a path of $\langle T, V \rangle$ (from the root) encoding an extended TM block *ext_bl* with component number h such that either $h > \ell$ or $\mathsf{ID}(ext\_bl) > 0$. Then, for each $t \in \Lambda$, there is a child z of $z_{2c(n)+1}$ which is labeled by $\{t\}$.
2. ℓ*-initialization requirement.* There are $w_1, \ldots, w_\ell \in I^{2^n}$ s.t. for each extended block *ext_bl* of $\langle T, V \rangle$ with component number $1 \leq h \leq \ell$ and position number 0, $bl = \mathsf{CON}(ext\_bl)$ encodes the $\mathsf{ID}(bl)$th symbol of the hth component of any *initial* (k,n)-configuration associated with a (k,n)-input of the form $(w_1, \ldots, w_\ell, w'_{\ell+1}, \ldots, w'_k)$ for some $w'_{\ell+1}, \ldots, w'_k \in I^{2^n}$. We say that $w_1, \ldots, w_\ell \in I^{2^n}$ is the ℓ-ary input (which is uniquely determined) associated with $\langle T, V \rangle$ and we write $\langle T, V \rangle(w_1, \ldots, w_\ell)$.

Intuitively, a ℓ-initialized full (k,n)-computation tree code $\langle T, V \rangle$ associated with a ℓ-ary input $w_1, \ldots, w_\ell \in I^{2^n}$ encodes all the possible (k,n)-computations from (k,n)-inputs of the form $(w_1, \ldots, w_\ell, w'_{\ell+1}, \ldots, w'_k)$ for arbitrary words $w'_{\ell+1}, \ldots, w'_k \in I^{2^n}$. More precisely, by construction, the following holds.

Proposition 3. *Let* $1 \leq \ell \leq k$, $w_1, \ldots, w_\ell \in I^{2^n}$, *and* $\langle T, V \rangle$ *be a* ℓ*-initialized full* (k,n)*-computation tree code such that* $\langle T, V \rangle(w_1, \ldots, w_\ell)$ *holds. Then, the following holds:*

1. *case* $\ell < k$*: for each* $w \in I^{2^n}$*, there is a refinement* $\langle T_r, V_r \rangle$ *of* $\langle T, V \rangle$ *which is a* $\ell + 1$*-initialized full* (k,n)*-computation tree code satisfying* $\langle T_r, V_r \rangle(w_1, \ldots, w_\ell, w)$*. Moreover, for each refinement* $\langle T_r, V_r \rangle$ *of* $\langle T, V \rangle$ *which is a* $\ell + 1$*-initialized full* (k,n)*-computation tree code, there is* $w \in I^{2^n}$ *such that* $\langle T_r, V_r \rangle(w_1, \ldots, w_\ell, w)$ *holds.*

2. *case $\ell = k$: the set of refinements of $\langle T, V\rangle$ which are (k,n)-computation tree codes encoding (k,n)-computations is non-empty, and each of such refinements encodes the (k,n)-computation from the (k,n)-input $(w_1,\dots,w_k)$.*

Lemma 4. *One can construct in time polynomial in n, k, and the size of the TM $\mathcal{M}$,*

1. *a $\mathsf{RML}^{\forall}$ formula φ^1_{init} over P such that given a tree structure $\langle T, V\rangle$, $\langle T, V\rangle$ satisfies φ^1_{init} if and only if $\langle T, V\rangle$ is a 1-initialized full (k,n)-computation tree code;*
2. *a $\mathsf{RML}^{\forall}$ formula φ^{ℓ}_{init} over P (for each $2 \le \ell \le k$) such that given a* refinement *$\langle T_r, V_r\rangle$ of a $\ell - 1$-initialized full (k,n)-computation tree code, $\langle T_r, V_r\rangle$ satisfies φ^{ℓ}_{init}* if and only if *$\langle T_r, V_r\rangle$ is a ℓ-initialized full (k,n)-computation tree code;*
3. *a $\mathsf{RML}^{\forall}$ formula φ_{comp} over P such that given a* refinement *$\langle T_r, V_r\rangle$ of a k-initialized full (k,n)-computation tree code, $\langle T_r, V_r\rangle$ satisfies φ_{comp}* iff *$\langle T_r, V_r\rangle$ is a (k,n)-computation tree code encoding a (k,n)-computation;*
4. *a ML formula φ_{acc} over P such that given a (k,n)-computation tree code $\langle T, V\rangle$, $\langle T, V\rangle$ satisfies φ_{acc}* iff *the (k,n)-configuration with position number $2^{\mathsf{c}(k,n)-1}$ encoded by $\langle T, V\rangle$ is accepting.*

Theorem 3 directly follows from the following two results (Theorems 4 and 5).

Theorem 4. *One can construct a RML^{k+1} formula φ in time polynomial in n, k, and the size of the TM $\mathcal{M}$ such that φ is satisfiable* if and only if

$$\mathsf{Q}_1 x_1 \in I^{2^n}.\mathsf{Q}_2 x_2 \in I^{2^n}.\dots\mathsf{Q}_k x_k \in I^{2^n}.\mathcal{M}(x_1,\dots,x_k)$$

where $\mathsf{Q}_\ell = \exists$ if ℓ is odd, and $\mathsf{Q}_\ell = \forall$ otherwise (for all $1 \le \ell \le k$).

Proof. Let $\varphi^1_{init},\dots,\varphi^k_{init}$, and φ_{comp} be the $\mathsf{RML}^{\forall}$ formulas satisfying Properties 1–3 of Lemma 4, and φ_{acc} be the ML formula satisfying Property 4 of Lemma 4. Then, the RML^{k+1} formula φ is defined as follows, where $\widetilde{\mathsf{Q}}_\ell = \exists_r$ and $\mathsf{op}_\ell = \wedge$ if ℓ is odd, and $\widetilde{\mathsf{Q}}_\ell = \forall_r$ and $\mathsf{op}_\ell = \rightarrow$ otherwise (for all $2 \le \ell \le k$):[3]

$$\varphi := \varphi^1_{init} \wedge \widetilde{\mathsf{Q}}_2(\varphi^2_{init}\ \mathsf{op}_2\ \widetilde{\mathsf{Q}}_3(\varphi^3_{init}\ \mathsf{op}_3\ \dots\ \mathsf{op}_{k-1}\ \widetilde{\mathsf{Q}}_k(\varphi^k_{init}\ \mathsf{op}_k\ \widetilde{\mathsf{Q}}_k(\varphi_{comp}\ \mathsf{op}_k\ \varphi_{acc}))\dots))$$

By construction and Lemma 4, it easily follows that φ is RML^{k+1} formula which can be constructed in time polynomial in n, k, and the size of the TM $\mathcal{M}$. Let $\varphi_1 := \varphi$, $\varphi_{k+1} := \widetilde{\mathsf{Q}}_k(\varphi_{comp}\ \mathsf{op}_k\ \varphi_{acc})$, and for each $2 \le \ell \le k$,

$$\varphi_\ell := \widetilde{\mathsf{Q}}_\ell(\varphi^\ell_{init}\ \mathsf{op}_\ell\ \widetilde{\mathsf{Q}}_{\ell+1}(\varphi^{\ell+1}_{init}\ \mathsf{op}_{\ell+1}\ \dots\ \mathsf{op}_{k-1}\ \widetilde{\mathsf{Q}}_k(\varphi^k_{init}\ \mathsf{op}_k\ \widetilde{\mathsf{Q}}_k(\varphi_{comp}\ \mathsf{op}_k\ \varphi_{acc}))\dots))$$

Correctness of the construction directly follows from the following claim, where a 0-initialized full (k,n)-computation tree code is an arbitrary tree structure.

Claim: let $0 \le \ell \le k, w_1,\dots,w_\ell \in I^{2^n}$, and $\langle T, V\rangle$ be a ℓ-initialized full (k,n)-computation tree code such that $\langle T, V\rangle(w_1,\dots,w_\ell)$ holds. Then, $\langle T, V\rangle$ satisfies $\varphi_{\ell+1}$ if and only if

$$\mathsf{Q}_{\ell+1} x_{\ell+1} \in I^{2^n}.\dots\mathsf{Q}_k x_k \in I^{2^n}.\mathcal{M}(w_1,\dots,w_\ell,x_{\ell+1},\dots,x_k)$$

The claim follows from Proposition 3 and Lemma 4. □

Theorem 5. *Let $k = 1$. Then, one can construct a $\mathsf{RML}^{\forall}$ formula $\varphi^{\forall}$ in time polynomial in n and the size of the TM $\mathcal{M}$ such that $\varphi^{\forall}$ is satisfiable* if and only if $\exists x \in I^{2^n}.\mathcal{M}(x)$.

[3] For RML formulas φ and ψ, $\varphi \rightarrow \psi$ is an abbreviation for $\widetilde{\varphi} \vee \psi$.

5 Concluding Remarks

An intriguing question left open is the complexity of satisfiability for *multi-agent* RML [18,4]. Our approach does not seem to scale to the multi-agent case. Indeed, in this more general setting, refinement is defined according to a designated agent so that refinement restricts (modulo bisimulation) the accessibility relation of the designated agent, but preserves (modulo bisimulation) those of all the other agents. From a technical point of view, this means that a generalization of the minimalization lemma for the multi-agent framework (Lemma 3) which preserves the crucial semantic $\forall_r$-consistency requirement does not seem possible, and a different more sophisticated approach may be required. Another interesting direction is to investigate the exact complexity of the fragments $\mathsf{RML}^{\exists}$, $\mathsf{RML}^{\forall}$, and RML^{k}, and the succinctness gap between RML^{k} and RML^{k+1} for each $k \geq 1$. Furthermore, since the modal μ-calculus extended with refinement quantifiers (RML^{μ}, for short) is non-elementarily decidable [4], it would be interesting to individuate weak forms of interactions between fixed-points and refinement quantifiers, which may lead to elementarily decidable and interesting RML^{μ}-fragments.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM 49(5), 672–713 (2002)
2. Balbiani, P., Baltag, A., van Ditmarsch, H., Herzig, A., Hoshi, T., De Lima, T.: ‘Knowable’ as ‘known after an announcement’. Review of Symbolic Logic 1(3), 305–334 (2008)
3. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science 53. Cambridge University Press, Cambridge (2001)
4. Bozzelli, L., van Ditmarsch, H., French, T., Hales, J., Pinchinat, S.: Refinement modal logic (2012), `http://arxiv.org/abs/1202.3538`
5. Chandra, A.K., Kozen, D.C., Stockmeyer, L.J.: Alternation. Journal of the ACM 28(1), 114–133 (1981)
6. Ferrante, J., Rackoff, C.: A decision procedure for the first order theory of real addition with order. SIAM Journal on Computing 4(1), 69–76 (1975)
7. Fine, K.: Propositional quantifiers in modal logic. Theoria 36(3), 336–346 (1970)
8. French, T.: Bisimulation quantifiers for modal logic. PhD thesis, University of Western Australia (2006)
9. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press (2000)
10. Hollenberg, M.: Logic and bisimulation. PhD thesis, University of Utrecht (1998)
11. Johnson, D.S.: A catalog of complexity classes. In: Handbook of Theoretical Computer Science, pp. A:67–A:161. MIT Press (1990)
12. Kupferman, O., Vardi, M.Y.: Verification of Fair Transisiton Systems. In: Alur, R., Henzinger, T.A. (eds.) CAV 1996. LNCS, vol. 1102, pp. 372–382. Springer, Heidelberg (1996)
13. Papadimitriou, C.H.: Computational Complexity. Addison Wesley (1994)
14. Parikh, R., Moss, L., Steinsvold, C.: Topology and epistemic logic. In: Handbook of Spatial Logics, pp. 299–341. Springer (2007)
15. Pinchinat, S.: A Generic Constructive Solution for Concurrent Games with Expressive Constraints on Strategies. In: Namjoshi, K.S., Yoneda, T., Higashino, T., Okamura, Y. (eds.) ATVA 2007. LNCS, vol. 4762, pp. 253–267. Springer, Heidelberg (2007)

16. Rybina, T., Voronkov, A.: Upper Bounds for a Theory of Queues. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 714–724. Springer, Heidelberg (2003)
17. van Ditmarsch, H., French, T.: Simulation and Information: Quantifying over Epistemic Events. In: Meyer, J.-J.C., Broersen, J. (eds.) KRAMAS 2008. LNCS, vol. 5605, pp. 51–65. Springer, Heidelberg (2009)
18. van Ditmarsch, H., French, T., Pinchinat, S.: Future event logic - axioms and complexity. In: Proc. 7th Advances in Modal Logic, vol. 8, pp. 77–99. College Publications (2010)
19. Visser, A.: Bisimulations, model descriptions and propositional quantifiers. Logic Group Preprint Series 161, Department of Philosophy, Utrecht University (1996)

The View-Update Problem for Indefinite Databases

Luciano Caroprese[1], Irina Trubitsyna[1], Mirosław Truszczyński[2], and Ester Zumpano[1]

[1] DEIS, Università della Calabria, 87030 Rende, Italy
{caroprese,irina,zumpano}@deis.unical.it
[2] Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA
mirek@cs.uky.edu

Abstract. This paper introduces and studies a declarative framework for updating views over *indefinite databases*. An indefinite database is a database with null values that are represented, following the standard database approach, by a single *null* constant. The paper formalizes views over such databases as indefinite *deductive* databases, and defines for them several classes of database repairs that realize view-update requests. Most notable is the class of *constrained repairs*. Constrained repairs change the database "minimally" and avoid making arbitrary commitments. They narrow down the space of alternative ways to fulfill the view-update request to those that are grounded, in a certain strong sense, in the database, the view and the view-update request.

1 Introduction

A typical database system is large and complex. Users and applications rarely can access the entire system directly. Instead, it is more common that access is granted in terms of a *view*, a *virtual* database consisting of relations defined by a query to the *stored and maintained* database. Querying a view does not present a conceptual problem. In contrast, another key task, *view updating*, poses major challenges.

Example 1. Let $D = \{q(a,b)\}$ be a database over relation symbols q and r, where the relation r has arity three and is currently empty. Let us consider the view over D given by the Datalog program $P = \{p(X) \leftarrow q(X,Y), r(X,Y,Z)\}$. That view consists of a single unary relation p. Given the present state of D, the view is empty.

To satisfy the request that $p(a)$ holds in the view (as it is now, it does not), one needs to update the database D. Such update consists of executing update actions that specify facts to insert to and to delete from D. These update actions (in a simplified setting that we consider for now) are "signed" facts $+F$ and $-G$, where $+F$ stands for "insert F" and $-G$ stands for "delete G." In our case, the set of update actions $\{-q(a,b), +q(a,a), +r(a,a,a)\}$ is a correct update to D. Executing it on D results in the database $D' = \{q(a,a), r(a,a,a)\}$, which has the desired property that $p(a)$ holds in the view determined by P. There are also other ways to satisfy the user's request, for instance: $\{+r(a,b,a)\}$ and $\{+q(c,d), +r(c,d,d)\}$, where c and d are any elements of the domain of the database. □

As this example suggests, view updating consists of translating a *view-update request*, that is, an update request against the view, into an *update*, a set of update actions against the stored (extensional) database. The example highlights the basic problem of view updating. It may be (in fact, it is common), that a view-update request can be fulfilled by any of a large number of database updates. One of them has to be committed to and executed. Thus, developing methods to automate the selection process or, at least, aid the user in making the choice is essential and has been one of the central problems of view updating [9,2,13,11]. That problem is also the focus of our paper.[1]

To restrict the space of possible database updates to select from, it is common to consider only those that accomplish the view-update request and are in some sense minimal. That reduces the space of updates. For instance, the update $\{-q(a,b), +q(a,a), +r(a,a,a)\}$ in Example 1 is not minimal. We can remove $-q(a,b)$ from it and what remains is still an update that once executed ensures that $p(a)$ holds in the view. Minimal updates fulfilling the view-update request are commonly called *repairs*.

Even imposing minimality may still leave a large number of candidate repairs. In Example 1, updates $\{+r(a,b,\psi)\}$, where ψ is any domain element, are repairs, and there are still more. As long as we insist on the completeness of the repaired database, there is little one can do at this point but ask the user to *select* one.

The situation changes if we are willing to allow indefiniteness (incomplete information) in databases. Given the "regular" structure of the family of repairs above, one could represent it by a single *indefinite* repair, $\{+r(a,b,\psi)\}$ regarding ψ as a distinct *null* value standing for some unspecified domain elements. The choice facing the user substantially simplifies as possibly infinitely many repairs is reduced to only one.

This approach was studied by Farré et al. [5], where the terminology of Skolem constants rather than null values was used. However, while seemingly offering a plausible solution to the problem of multiple repairs, the approach suffers from two drawbacks. First, the use of multiple Skolem constants (essentially, multiple null values) violates the SQL standard [15]. Second, the approach assumes that the initial database is complete (does not contain null values). Thus, while the approach might be applied once, iterating it fails as soon as an indefinite database is produced, since no guidance on how to proceed in such case is given.

We consider the view-update problem in the setting in which *both* the original and the updated databases are indefinite (may contain occurrences of null values) and the database domain is possibly infinite. To be compliant with the SQL standard, we allow a single null value only. We denote it by $\perp$ and interpret it as expressing the existence of unknown values for each attribute (with possibly infinite domain) it is used for [10]. Typically, a database is represented as a set of facts. We propose to represent indefinite databases by *two* sets of facts that we interpret by means of a two-level closed-world assumption tailored to the case of indefinite information. We interpret indefinite databases in terms of their *possible worlds*. We extend the possible-world semantics to the setting of views, which we formalize in terms of *indefinite deductive databases*, and apply the resulting formalism to state and study the view-updating problem.

We then turn attention to the problem of multiple repairs. In general, using null values to encode multiple repairs is still not enough to eliminate them entirely (cf. our

[1] In some cases no update satisfies the view-update request. The question whether the lack of an appropriate update is caused by errors in the design of the view, in the extensional database, or in the view-update request is interesting and deserves attention, but is outside the scope of the present work.

discussion above), and some level of the user's involvement may be necessary. Therefore, it is important to identify principled ways to narrow down the space of repairs for the user to consider. We propose a concept of minimality tailored precisely to the new setting. The primary concern is to minimize the set of new constants introduced by an update. The secondary concern is to minimize the degree of the semantic change. The resulting notion of minimality yields the notion of a *repair*.

Our concept of minimality leads us also to the concept of a *relevant* repair, an update that introduces no new constants and minimizes the degree of change. In Example 1, $\{+r(a,b,\perp)\}$, $\{+r(a,a,\perp),+q(a,a)\}$ and $\{+r(a,c,\perp),+q(a,c)\}$, where c is an element of the database domain other than a and b, are all repairs. The first two are obviously relevant, the third one is not.

Some occurrences of non-nullary constants in a relevant repair may still be "ungrounded" or "arbitrary," that is, replacing them with another constant results in a repair. For instance, replacing in the relevant repair $\{+r(a,a,\perp),+q(a,a)\}$ the second and the forth occurrences of a with a fresh constant c yields $\{+r(a,c,\perp),+q(a,c)\}$, an update that is a repair. Intuitively, "arbitrary" occurrences of constants, being replaceable, are not forced by the information present in the view (deductive database) and in the view-update request. By restricting relevant repairs to those *without* arbitrary occurrences of constants we arrive at the class of *constrained* repairs. In the view-update problem considered in Example 1, there is only *one* constrained repair, $\{+r(a,b,\perp)\}$.

Finally, we study the complexity of the problems of the existence of repairs, relevant repairs and constrained repairs. We obtain precise results for the first two classes and an upper bound on the complexity for the last class.

To summarize, our main contributions are as follows. We propose a two-set representation of indefinite database that is more expressive than the standard one. We define the semantics and the operation of updating indefinite databases (Section 2). We define views over indefinite databases (indefinite deductive databases), and generalize the semantics of indefinite databases to views (Section 3). We state and study the view-update problem in the general setting when the initial and the repaired databases are indefinite. We propose a notion of minimality of an update and use it to define the concept of a repair. We address the problem of multiple repairs by defining relevant and constrained repairs (Section 4). We study the complexity of problems of existence of repairs, relevant repairs and constrained repairs (Section 5).

2 Indefinite Databases

We consider a finite set Π of relation symbols and a set *Dom* of constants that includes a designated element $\perp$, called the *null value*. We define $Dom_d = Dom \setminus \{\perp\}$. Normally, we assume that *Dom* is an infinite countable set. However, for the sake of simplicity, in several of the examples the set *Dom* is finite.

Some predicates in Π are designated as *base* (or *extensional*) predicates and all the remaining ones are understood as *derived* (or *intensional*) predicates. A *term* is a constant from *Dom* or a variable. An *atom* is an expression of the form $p(t_1,\ldots,t_k)$, where $p \in \Pi$ is a *predicate symbol* of arity k and t_i's are terms. An atom is *ground* if it does not contain variables. We refer to ground atoms as *facts*. We denote the set of all facts by At. We call facts defined in terms of base and derived predicates *base facts* and *derived facts* respectively. A fact is *definite* if it does not contain occurrences of $\perp$. Otherwise, it is *indefinite*. Given a set S of atoms, we define $Dom(S)$ (resp. $Dom_d(S)$) as the set of constants in Dom (resp. Dom_d) occurring in S. For every two tuples of terms $t = (t_1,\ldots,t_k)$ and $t' = (t'_1,\ldots,t'_k)$ and every k-ary predicate symbol $p \in \Pi$, we write $t \preceq t'$ and $p(t) \preceq p(t')$ if for every i, $1 \le i \le k$, $t_i = t'_i$ or $t_i = \perp$. We say in such case that t' and $p(t')$ are *at least as informative* as t and $p(t)$, respectively. If, in addition, $t \neq t'$, we write $t \prec t'$ and $p(t) \prec p(t')$, and say that t' and $p(t')$ are *more informative* than t and $p(t)$. Sometimes, we say "at most as informative" and "less informative," with the obvious understanding of the intended meaning. We also define t and t' (respectively, $p(t)$ and $p(t')$) to be *compatible*, denoted by $t \approx t'$ (respectively, $p(t) \approx p(t')$), if for some k-tuple s of terms, $t \preceq s$ and $t' \preceq s$. Finally, for a set $D \subseteq At$, we define

$$D^{\Downarrow} = \{a \mid \text{there is } b \in D \text{ s.t. } a \preceq b\} \quad D^{\Uparrow} = \{a \mid \text{there is } b \in D \text{ s.t. } b \preceq a\}$$
$$D^{\approx} = \{a \mid \text{there is } b \in D \text{ s.t. } b \approx a\} \quad D^{\sim} = D^{\approx} \setminus D^{\Downarrow}.$$

To illustrate, let q be a binary relation symbol and $Dom = \{\perp, 1, 2\}$. Then:

$$\{q(1,\perp)\}^{\Downarrow} = \{q(1,\perp), q(\perp,\perp)\} \qquad \{q(1,\perp)\}^{\Uparrow} = \{q(1,\perp), q(1,1), q(1,2)\}$$
$$\{q(1,\perp)\}^{\approx} = \{q(1,\perp), q(1,1), q(1,2), q(\perp,1), q(\perp,2), q(\perp,\perp)\}$$
$$\{q(1,\perp)\}^{\sim} = \{q(1,1), q(1,2), q(\perp,1), q(\perp,2)\}.$$

We note also note that $D^{\approx} = (D^{\Uparrow})^{\Downarrow}$.

In the most common case, databases are finite subsets of At that contain *definite* facts only. The semantics of such databases is given by the *closed-world assumption* or CWA [12]: a definite fact q is *true* in a database D if $q \in D$. Otherwise, q is *false* in D. We are interested in databases that may contain indefinite facts, too. Generalizing, we will *for now* assume that an indefinite database is a finite set of possibly indefinite atoms. The key question is that of the semantics of indefinite databases.

Let D be an indefinite database. Clearly, all facts in D are true in D. In addition, any fact that is less informative than a fact in D is also true in D. Indeed, each such fact represents an existential statement, whose truth is established by the presence of a more informative fact in D (for instance, the meaning of $p(\perp)$ is that there is an element c in the domain of the database such that $p(c)$ holds; if $p(1) \in D$, that statement is true). Summarizing, every fact in $D^{\Downarrow}$ is true in D.

By CWA adapted for the setting of indefinite databases [10], facts that are not in $D^{\Downarrow}$ are *not* true in D, as D contains no evidence to support their truth. Those facts among them that are compatible with facts in D (in our notation, facts in $D^{\sim}$), might actually be true, but the database just does not know that. Of course, they may also be false, the database does not exclude that possibility either. They are regarded as *unknown*. By CWA again, the facts that are not compatible with any fact in D are false in D, as D provides no explicit evidence otherwise.

The simple notion of an indefinite database, while intuitive and having a clear semantics, has a drawback. It has a limited expressive power. For instance, there is no database D to represent our knowledge that $p(1)$ is false and that there is some definite c such that $p(c)$ holds (clearly, this c is not 1). To handle such cases, we introduce a more general concept of an indefinite database, still using CWA to specify its meaning.

Definition 1. *An* indefinite database *(a* database*, for short) is a pair $\mathscr{I} = \langle D, E\rangle$, where D and E are finite sets of (possibly indefinite) facts.* □

The intended role of D is to represent all facts that are true in the database $\langle D, E\rangle$, while E is meant to represent *exceptions*, those facts that normally would be unknown, but are in fact false (and the database knows it). More formally, the semantics of indefinite databases is presented in the following definition.

Definition 2. *Let $\langle D, E\rangle$ be a database and let $q \in At$ be a fact. Then: (1) q is* true *in $\langle D, E\rangle$, written $\langle D, E\rangle \models q$, if $q \in D^{\Downarrow}$; (2) q is* unknown *in $\langle D, E\rangle$, if $q \in (D^{\approx} \setminus D^{\Downarrow}) \setminus E^{\Uparrow}$ $(= D^{\sim} \setminus E^{\Uparrow})$; (3) q is* false *in $\langle D, E\rangle$, denoted $\langle D, E\rangle \models \neg q$, in all other cases, that is, if $q \notin D^{\approx}$ or if $q \in D^{\sim} \cap E^{\Uparrow}$.* □

The use of $E^{\Uparrow}$ in the definition (items (2) and (3)) reflects the property that if an atom a is false then every atom b at least as informative as a must be false, too.

We denote the sets of all facts that are true, unknown and false in a database $\mathscr{I} = \langle D, E\rangle$ by $\mathscr{I}_t$, $\mathscr{I}_u$ and $\mathscr{I}_f$, respectively. Restating the definition we have:

$$\mathscr{I}_t = D^{\Downarrow}, \quad \mathscr{I}_u = D^{\sim} \setminus E^{\Uparrow}, \text{ and } \quad \mathscr{I}_f = (At \setminus D^{\approx}) \cup (D^{\sim} \cap E^{\Uparrow}).$$

Example 2. The knowledge that $p(c)$ holds for some constant c and that $p(1)$ is false can be captured by the database $\langle \{p(\bot)\}, \{p(1)\}\rangle$. The database $\langle \{q(\bot,\bot), q(1,1)\}, \{q(1,\bot)\}\rangle$ specifies that the atoms $q(1,1)$, $q(1,\bot)$, $q(\bot,1)$ and $q(\bot,\bot)$ are true, that all definite atoms $q(1,d)$, with $d \neq 1$, are false, and that all other atoms $q(c,d)$ are unknown. While the fact that $q(\bot,\bot)$ is true follows from the fact that $q(1,1)$ is true, the presence of the former in the database is not redundant, that is, the database without it would have a different meaning. Namely, $q(\bot,\bot)$ makes all atoms $q(a,b)$ potentially unknown (with some of them true or false due to other elements of the database). □

Definition 3. *A set W of* definite *facts is a* possible world *for a database $\mathscr{I} = \langle D, E\rangle$ if $\mathscr{I}_t \subseteq W^{\Downarrow}$ (W "explains" all facts that are true in $\mathscr{I}$), $W \subseteq \mathscr{I}_t \cup \mathscr{I}_u$ (only definite facts that are true or unknown appear in a possible world).* □

Databases represent sets of possible worlds. For a database $\mathscr{I}$, we write $\mathscr{W}(\mathscr{I})$ to denote the family of all possible worlds for $\mathscr{I}$. Due to the absence of indefinite facts in $W \in \mathscr{W}(\mathscr{I})$, every fact in At is either true (if it belongs to $W^{\Downarrow}$) or false (otherwise) w.r.t. W. Extending the notation we introduced earlier, for a possible world W and $a \in At$ we write $W \models a$ if $a \in W^{\Downarrow}$ and $W \models \neg a$, otherwise. The following proposition shows that the semantics of a database can be stated in terms of its possible worlds.

Proposition 1. *Let $\mathscr{I}$ be a database and q a fact. Then $q \in \mathscr{I}_t$ if and only if $W \models q$, for every $W \in \mathscr{W}(\mathscr{I})$, and $q \in \mathscr{I}_f$ if and only if $W \models \neg q$, for every $W \in \mathscr{W}(\mathscr{I})$.* □

Updating a database $\langle D, E\rangle$ consists of executing on it *update actions*: inserting a base fact a into D or E, and deleting a base fact a from D or E. We denote them by $+a^D$, $+a^E$, $-a^D$ and $-a^E$, respectively. For a set U of update actions, we define $U_+^D = \{a \mid +a^D \in U\}$, $U_+^E = \{a \mid +a^E \in U\}$, $U_-^D = \{a \mid -a^D \in U\}$, and $U_-^E = \{a \mid -a^E \in U\}$. To be executable, a set U of update actions must not contain *contradictory* update actions:

$+a^D$ and $-a^D$, or $+a^E$ and $-a^E$. A contradiction-free set U of update actions is an *update*. We denote the set of all updates (in the fixed language we consider) by $\mathscr{U}$.

We now define the operation to update a database, that is, to apply an update to it.

Definition 4. *Let $\mathscr{I} = \langle D, E\rangle$ be an indefinite database and U an update. We define $\mathscr{I} \circ U$ as the database $\langle D', E'\rangle$, where $D' = (D \cup U_+^D) \setminus U_-^D$ and $E' = (E \cup U_+^E) \setminus U_-^E$.* □

3 Indefinite Deductive Databases

Integrity constraints (ICs, for short) are first-order sentences in the language $\mathscr{L}$ determined by the set of predicates Π and by the set Dom_d of definite constants. A database with integrity constraints is a pair $\langle \mathscr{I}, \eta\rangle$, where $\mathscr{I}$ is a database and η is a set of ICs. Possible worlds can be regarded as interpretations of the language $\mathscr{L}$ with Dom_d as their domain. This observation and Proposition 1 suggest a definition of the semantics of databases with ICs.

Definition 5. *Let $\langle \mathscr{I}, \eta\rangle$ be a database with ICs. A possible world $W \in \mathscr{W}(\mathscr{I})$ is a* possible world for a database $\langle \mathscr{I}, \eta\rangle$ *if W satisfies every integrity constraint in η. We denote the set of possible worlds of $\langle \mathscr{I}, \eta\rangle$ by $\mathscr{W}(\mathscr{I}, \eta)$. A database with ICs, $\langle \mathscr{I}, \eta\rangle$, is* consistent *if $\mathscr{W}(\mathscr{I}, \eta) \neq \emptyset$. Otherwise, $\langle \mathscr{I}, \eta\rangle$ is* inconsistent.

A fact q is true in $\langle \mathscr{I}, \eta\rangle$ if $W \models q$, for every $W \in \mathscr{W}(\mathscr{I}, \eta)$; q is false in $\langle \mathscr{I}, \eta\rangle$ if $W \models \neg q$, for every $W \in \mathscr{W}(\mathscr{I}, \eta)$; otherwise, q is unknown in $\langle \mathscr{I}, \eta\rangle$. □

Example 3. Let us consider the database $\mathscr{I} = \langle\{p(1), p(2), q(\bot)\}, \emptyset\rangle$ (there are no exceptions) and let $\eta = \{\forall X(q(X) \rightarrow p(X))\}$. Possible worlds of $\mathscr{I}$ include $\{p(1), p(2), q(1)\}$, $\{p(1), p(2), q(2)\}$ and $\{p(1), p(2), q(3)\}$. The first two satisfy the integrity constraint, the third one does not. Thus, only the first two are possible worlds of $\langle \mathscr{I}, \eta\rangle$. Since $p(1)$ belongs to all possible worlds of $\langle \mathscr{I}, \eta\rangle$, $p(1)$ is true in $\langle \mathscr{I}, \eta\rangle$. Further, $p(3)$ is false in every possible world of $\mathscr{I}$ and so also in every possible world of $\langle \mathscr{I}, \eta\rangle$. Thus, $p(3)$ is false in $\langle \mathscr{I}, \eta\rangle$. Lastly, we note that $q(1)$ and $q(2)$ are unknown in $\langle \mathscr{I}, \eta\rangle$, while $q(3)$ is false (due to $p(3)$ being false and the integrity constraint). □

We note that the possible-world semantics can capture additional information contained in integrity constraints. In Example 3, the semantics derives that $q(3)$ is *false* in $\langle \mathscr{I}, \eta\rangle$ even though this knowledge is not present in the database $\mathscr{I}$.

The concepts of an update and of the operation to execute an update on a database extend literally to the case of databases with ICs.

Following Ullman [16], *views* are safe Datalog$^\neg$ programs. We use the standard terminology and talk about (Datalog$^\neg$) *rules*, and *bodies* and *heads* of rules. A rule is *safe* if each variable occurring in the head or in a negative literal in the body also occurs in a positive literal in the body. A Datalog$^\neg$ program is safe if each rule is *safe*. We assume that views do not contain occurrences of $\bot$. The semantics of Datalog$^\neg$programs is given in terms of *answer sets* [6,7]. A precise definition of that semantics is immaterial to our study and so we do not provide the details.

Definition 6. *An* indefinite deductive database *(from now, simply, a deductive database) is a tuple $\mathscr{D} = \langle \mathscr{I}, \eta, P\rangle$, where $\mathscr{I}$ is a database, η is a set of integrity constraints, and*

P is a safe Datalog$^\neg$ *program (the specification of a* view*) such that no predicate occurring in the head of a rule in P is a base predicate.* □

Clearly, a deductive database with the empty view is a database with ICs, and a deductive database with the empty view and no ICs is simply a database.

Definition 7. *A deductive database* $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$ *is* consistent *if* $\langle \mathscr{I}, \eta \rangle$ *is consistent and for every possible world* $W \in \mathscr{W}(\mathscr{I}, \eta)$*, the program* $W \cup P$ *has answer sets. We denote the family of all those answer sets by* $\mathscr{W}(\mathscr{D})$ *or* $\mathscr{W}(\mathscr{I}, \eta, P)$*. We call elements of* $\mathscr{W}(\mathscr{D})$ possible worlds *of* $\mathscr{D}$. □

There is an alternative to our concept of consistency. One could define a deductive database $\langle \mathscr{I}, \eta, P \rangle$ as consistent if for *at least one* world $W \in \mathscr{W}(\mathscr{I}, \eta)$, the program $W \cup P$ has an answer set. That concept of consistency would allow situations where for some possible worlds of $\langle \mathscr{I}, \eta \rangle$, one of which could be a description of the real world, the view P does not generate any meaningful virtual database. Our concept of consistency is more robust. It guarantees that the user can have a view of a database no matter how the real world looks like, that is, which of the possible worlds describes it.

Example 4. Let $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$ be a deductive database, where $\mathscr{I} = \langle \{p(\perp)\}, \emptyset \rangle$, $\eta = \emptyset$ and $P = \{t \leftarrow p(1), p(2), not\ t\}$, for some derived ground atom t. Every non-empty set $W \subseteq \{p(u) \mid u \in Dom_d\}$ is a possible world of $\langle \mathscr{I}, \eta \rangle$. In particular, the set $\{p(1), p(2)\}$ is a possible world of $\langle \mathscr{I}, \eta \rangle$. Since the program $P \cup \{p(1), p(2)\}$ has no answer sets, $\mathscr{D}$ is inconsistent (according to our definition). If $\mathscr{D}' = \langle \mathscr{I}, \{p(1) \wedge p(2) \rightarrow \perp\}, P \rangle$, then the integrity constraint in $\mathscr{D}'$ eliminates the offending possible world and one can check that for every possible world W of $\langle \mathscr{I}, \{p(1) \wedge p(2) \rightarrow \perp\} \rangle$, $P \cup W$ has an answer set. Thus, $\mathscr{D}'$ is consistent.[2] □

The concept of an update extends in a natural way to deductive databases. If U is an update, and $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$, we define the result of updating $\mathscr{D}$ by U by $\mathscr{D} \circ U = \langle \mathscr{I} \circ U, \eta, P \rangle$.

Next, we define the semantics of a deductive database $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$, again building on the characterization given by Proposition 1.

Definition 8. *A fact* $a \in At$ *is* true *in a deductive database* $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$*, denoted by* $\mathscr{D} \models a$*, if for every possible world* $W \in \mathscr{W}(\mathscr{D})$*,* $W \models a$*; a is* false *in* $\mathscr{D}$*, denoted by* $\mathscr{D} \models \neg a$*, if for every* $W \in \mathscr{W}(\mathscr{D})$*,* $W \models \neg a$*; a is* unknown *in* $\mathscr{D}$*, otherwise. We denote the truth value of a in* $\mathscr{D}$ *by* $v_{\mathscr{D}}(a)$. □

Example 5. Let $Dom = \{\perp, 1, 2, 3\}$ and $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$ be a deductive database, where $\mathscr{I} = \langle \{p(\perp)\}, \emptyset \rangle$, $\eta = \{p(2) \rightarrow \perp\}$ and $P = \{q(X) \leftarrow p(X)\}$.

We have $\mathscr{W}(\mathscr{I}, \eta) = \{\{p(1)\}, \{p(3)\}, \{p(1), p(3)\}\}$. Each of the possible worlds in $\mathscr{W}(\mathscr{I}, \eta)$, when extended with the view P, gives rise to a program that has answer sets. Thus, $\mathscr{D}$ is consistent. Moreover, the possible worlds for $\mathscr{D}$ are $\{p(1), q(1)\}$, $\{p(3), q(3)\}$ and $\{p(1), q(1), p(3), q(3)\}$ (in this case, one for each possible world in $\mathscr{W}(\mathscr{I}, \eta)$). It follows that $p(\perp)$ and $q(\perp)$ are true, $p(2)$ and $q(2)$ are false, and $p(1), q(1), p(3)$ and $q(3)$ are unknown in $\mathscr{D}$. □

[2] We point out that in the paper, we overload the notation $\perp$. We use it to denote both the single null value in the language and the falsity symbol in the first-order language used for integrity constraints. Since the meaning is always clear from the context, no ambiguity arises.

4 View Updating

In the *view update problem*, the user specifies a *request*, a list of facts the user learned (observed) to be true or false, and wants the stored database to be updated to reflect it.[3]

Definition 9. *A* request *over a deductive database $\mathscr{D}$ is a pair $\mathscr{S} = (\mathscr{S}_t, \mathscr{S}_f)$, where $\mathscr{S}_t$ and $\mathscr{S}_f$ are disjoint sets of facts requested to be true and false, respectively.* □

To fulfill a request we need an update which, when executed, yields a database such that the view it determines satisfies the request. We call such updates *weak repairs*.

Definition 10. *Let $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$ be a deductive database and $\mathscr{S}$ a request. An update U for $\mathscr{I}$ is a* weak repair *for $(\mathscr{D}, \mathscr{S})$ if U* fulfills *$\mathscr{S}$, that is, if for every $a \in \mathscr{S}_t$, $v_{\mathscr{D}\circ U}(a) = true$ and for every $a \in \mathscr{S}_f$, $v_{\mathscr{D}\circ U}(a) = false$.* □

We are primarily interested in updates that do not drastically change the database. One condition of being "non-drastic" is not to introduce new predicate or constant symbols. That leads us to the notion of a relevant weak repair.

Definition 11. *Let $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$ be a deductive database and $\mathscr{S}$ a request. A constant is* relevant with respect to $\mathscr{D}$ and $\mathscr{S}$ *if it occurs in $\mathscr{D}$, or $\mathscr{S}$, or if it is $\perp$. A predicate is* relevant with respect to $\mathscr{D}$ and $\mathscr{S}$ *if it occurs in $\mathscr{D}$ or in $\mathscr{S}$. A weak repair U for $(\mathscr{D}, \mathscr{S})$ is* relevant *if every constant and predicate occurring in U is* relevant. □

More generally, a weak repair is "non-drastic" if it minimizes the change it incurs [14]. There are two aspects to the minimality of change: (1) minimizing the set of new predicate symbols and constants introduced by an update to the database (in the extreme case, no new symbols must be introduced, and we used that requirement to define relevant weak repairs above); (2) minimizing the change in the truth values of facts with respect to the database. Following the *Ockham's Razor* principle to avoid introducing new entities unless necessary, we take the minimality of the set of new symbols as a primary consideration. To define the resulting notion of change minimality, we assume that the truth values are ordered *false* $\leq$ *unknown* $\leq$ *true*. Further, for a deductive database $\mathscr{D}$, a request set $\mathscr{S}$ and an update $U \in \mathscr{U}$ we define $NC(\mathscr{D}, \mathscr{S}, U)$ as the set of non nullary constants that occur in U and not in $\mathscr{D}$ and $\mathscr{S}$.

Definition 12. *Let $\mathscr{D} = \langle \mathscr{I}, \eta \rangle$ be a database with integrity constraints. For updates $V, U \in \mathscr{U}$, we define $U \sqsubseteq V$ if: $NC(\mathscr{D}, \mathscr{S}, U) \subset NC(\mathscr{D}, \mathscr{S}, V)$, or $NC(\mathscr{D}, \mathscr{S}, U) = NC(\mathscr{D}, \mathscr{S}, V)$ and for every* base atom *a*

1. *if $v_{\mathscr{D}}(a) = true$, then $v_{\mathscr{D}\circ U}(a) \geq v_{\mathscr{D}\circ V}(a)$*
2. *if $v_{\mathscr{D}}(a) = false$, then $v_{\mathscr{D}\circ V}(a) \geq v_{\mathscr{D}\circ U}(a)$*
3. *if $v_{\mathscr{D}}(a) = unknown$, then $v_{\mathscr{D}\circ U}(a) = unknown$ or $v_{\mathscr{D}\circ V}(a) = v_{\mathscr{D}\circ U}(a)$.*

We also define $U \sqsubset V$ if $U \sqsubseteq V$ and $V \not\sqsubseteq U$. □

[3] We do not allow requests that facts be *unknown*. That is, we only allow definite requests. While there may be situations when all the user learns about the fact is that it is unknown, they seem to be rather rare. In a typical situation, the user will learn the truth or falsity of a fact.

We now define the classes of *repairs* and *relevant repairs* as subclasses of the respective classes of weak repairs consisting of their $\sqsubseteq$-minimal elements.

Definition 13. *Let $\mathscr{D} = \langle \mathscr{I}, \eta, P\rangle$ be a deductive database and $\mathscr{S}$ a request. A (*relevant*)* repair *for $(\mathscr{D},\mathscr{S})$ is a $\sqsubseteq$-minimal (relevant) weak repair for $(\mathscr{D},\mathscr{S})$.* □

We note that the existence of (weak) repairs does not guarantee the existence of relevant (weak) repairs. The observation remains true even if the view is empty.

Example 6. Let $\mathscr{D} = \langle \mathscr{I}, \eta, P\rangle$, where $\mathscr{I} = \langle \emptyset, \emptyset\rangle$, $\eta = \emptyset$ and $P = \{t \leftarrow p(x), q(x)\}$. If the request is $(\{t\}, \emptyset)$, then each repair is of the form $\{+p(i)^D, +q(i)^D\}$, for some $i \in Dom_d$. None of them is relevant. (We note that $\{+p(\bot)^D, +q(\bot)^D\}$ is not a (weak) repair. The database resulting from the update would admit possible worlds of the form $\{p(i), q(j)\}$, where $i \neq j$. Clearly, the corresponding possible world of the view over any such database does not contain t and so the update does not fulfill the request.) □

Some relevant constants are not "forced" by the database and the request, that is, can be replaced by other constants. If such constants are present in a relevant (weak) repair, this repair is *arbitrary*. Otherwise, it is *constrained*. A formal definition follows.

Definition 14. *Let $\mathscr{D} = \langle \mathscr{I}, \eta, P\rangle$ be a deductive database and $\mathscr{S}$ a request. A relevant (weak) repair U for $(\mathscr{D},\mathscr{S})$ is* constrained *if there is no non-nullary constant a in U such that replacing some occurrences of a in U with a constant $b \neq a$ (b might be $\bot$), results in a weak repair for $(\mathscr{D},\mathscr{S})$.* □

Example 7. Let $\mathscr{D} = \langle \mathscr{I}, \eta, P\rangle$, where $\mathscr{I} = \langle \{p(1), h(2)\}, \emptyset\rangle$, $\eta = \emptyset$ and $P = \{t \leftarrow p(X), q(X);\ s \leftarrow r(X)\}$. Let us consider the request $\mathscr{S} = (\{s,t\}, \emptyset)$. The updates $\mathscr{R}_i = \{+q(1)^D, +r(i)^D\}$, $i \in \{\bot, 1, 2\}$, and $\mathscr{R}'_i = \{+q(2)^D, +p(2)^D, +r(i)^D\}$, $i \in \{\bot, 1, 2\}$, are relevant weak repairs. One of them, $\mathscr{R}_\bot$, is constrained. Indeed, replacing in $\mathscr{R}_\bot$ the unique occurrence of a non-nullary constant (in this case, 1) with any other constant does not yield a weak repair. On the other hand, $\mathscr{R}_i$, $i \in \{1,2\}$, and $\mathscr{R}'_i$, $i \in \{\bot, 1, 2\}$, are not constrained. Indeed, replacing with 3 the second occurrence of 1 in $\mathscr{R}_1$, or the occurrence of 2 in R_2, or both occurrences of 2 in $\mathscr{R}'_i$ in each case results in a weak repair. Also weak repairs $\mathscr{R}_i = \{+q(1)^D, +r(i)^D\}$ and $\mathscr{R}'_i = \{+q(2)^D, +p(2)^D, +r(i)^D\}$, $i \in \{3, \ldots\}$, are not constrained as they are not even relevant. □

We stress that, in order to test whether a relevant (weak) repair $\mathscr{R}$ is constrained, we need to consider every *subset of occurrences* of non-nullary constants in $\mathscr{R}$. For instance, in the case of the repair $\mathscr{R}_1 = \{q(1), r(1)\}$ from Example 7, the occurrence of the constant 1 in $q(1)$ is constrained by the presence of $p(1)$. Replacing that occurrence of 1 with 3 does not result in a weak repair. However, replacing the occurrence of 1 in r with 3 gives a weak repair and shows that $\mathscr{R}_1$ is not constrained.

Example 8. Let $Dom = \{\bot, 1, 2, \ldots\}$ and $\mathscr{D} = \langle \mathscr{I}, \eta, P\rangle$, where $\mathscr{I} = \langle \{q(1,2), s(1,2,3)\}, \emptyset\rangle$, $\eta = \emptyset$ and $P = \{p(X) \leftarrow q(X,Y), r(X,Y,Z);\ r(X,Y,Z) \leftarrow s(X,Y,Z), t(X,Y,Z)\}$. Let us consider the request $\mathscr{S} = (\{p(1)\}, \emptyset)$. In this case, our approach yields a unique constrained repair $\mathscr{R} = \{+t(1,2,3)^D\}$. It recognizes that thanks to $s(1,2,3)$ simply inserting $t(1,2,3)$ guarantees $r(1,2,3)$ to be true and, consequently, ensures the presence of $p(1)$ in the view. There are other repairs and other relevant repairs, but only the one listed above is constrained. □

We observe that every (relevant, constrained) weak repair contains a (relevant, constrained) repair.

Proposition 2. *Let $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$ be a deductive database and $\mathscr{S}$ a request. A (relevant, constrained) repair for $(\mathscr{D}, \mathscr{S})$ exists if and only if a (relevant, constrained) weak repair for $(\mathscr{D}, \mathscr{S})$ exists.* □

5 Complexity

Finally, we discuss the complexity of problems concerning the existence of (weak) repairs of types introduced above. The results we present here have proofs that are non-trivial despite rather strong assumptions we adopted.

We assume that the sets of base and derived predicate symbols, the set of integrity constraints η and the view P are fixed. The only varying parts in the problems are a database $\mathscr{I}$ and a request $\mathscr{S}$. That is, we consider the *data complexity* setting. Moreover, we assume that $Dom = \{\bot, 1, 2, \ldots\}$, and take $=$ and $\leq$, both with the standard interpretation on $\{1, 2, \ldots\}$, as the *only* built-in relations. We restrict integrity constraints to expressions of the form: $\exists X (\forall Y (A_1 \wedge \ldots \wedge A_k \to B_1 \vee \ldots \vee B_m))$, where A_i and B_i are atoms with no occurrences of $\bot$ constructed of base and built-in predicates, and where every variable occurring in the constraint belongs to $X \cup Y$, and occurs in some atom A_i built of a base predicate.

We start by stating the result on the complexity of deciding the consistency of an indefinite database with integrity constraints. While interesting in its own right, it is also relevant to problems concerning the existence of repairs, as one of the conditions for U to be a repair is that the database that results from executing U be consistent.

Theorem 1. *The problem to decide whether a database $\langle \mathscr{I}, \eta \rangle$ has a possible world (is consistent) is NP-complete.* □

We now turn attention to the problem of checking request satisfaction. Determining the complexity of that task is a key stepping stone to the results on the complexity of deciding whether updates are (weak) repairs that are necessary for our results on the complexity of the existence of (weak) repairs. However, checking request satisfaction turns out to be a challenge even for very simple classes of views. In this paper, we restrict attention to the case when P is a safe definite (no constraints) acyclic (no recursion) Horn program, although we obtained Proposition 3 in a more general form.

Proposition 3. *The problem to decide whether a ground atom t is true in a deductive database $\langle \mathscr{I}, \eta, P \rangle$, where $\langle \mathscr{I}, \eta \rangle$ is consistent and P is a safe Horn program, is in the class co-NP.* □

Next, we consider the problem to decide whether a ground atom t is false in a deductive database $\langle \mathscr{I}, \eta, P \rangle$. We state it separately from the previous one as our present proof of that result requires the assumption of acyclicity.

Proposition 4. *The problem to decide whether a ground atom t is false (ground literal $\neg t$ is true) in a deductive database $\langle \mathscr{I}, \eta, P \rangle$, where $\langle \mathscr{I}, \eta \rangle$ is consistent and P is an acyclic Horn program, is in the class co-NP.* □

With Propositions 3 and 4 in hand, we move on to study the complexity of the problems of the existence of weak repairs. First, we establish an upper bound on the complexity of checking whether and update is a (relevant) weak repair.

Proposition 5. *Let $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$, where η is a set of integrity constraints, P an acyclic Horn program, U an update and $\mathscr{S}$ a request set. The problem of checking whether an update U is a weak repair (relevant weak repair) for $(\mathscr{D}, \mathscr{S})$ is in Δ_2^P.* □

With the results above, we can address the question of the complexity of the existence of repairs. The first problem concerns weak repairs and stands apart from others. It turns out, that deciding the existence of a weak repair is NP-complete, which may seem at odds with Proposition 5 (an obvious non-deterministic algorithm guesses an update U and checks that it is a weak repair apparently performing a "Σ_2^P computation"). However, this low complexity of the problem is simply due to the fact that there are no relevance, constrainedness or minimality constraints are imposed on weak repairs. Thus, the question can be reduced to the question whether there is a "small" database $\mathscr{J}$, in which the request holds. The corresponding weak repair consists of deleting all elements from $\mathscr{I}$ and "repopulating" the resulting empty database so that to obtain $\mathscr{J}$.

Theorem 2. *Let $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$, where η is a set of integrity constraints, and P an acyclic Horn program, and let $\mathscr{S}$ be a request set. The problem of deciding whether there is a weak repair for $(\mathscr{D}, \mathscr{S})$ is NP-complete.* □

As noted, the case of the existence of weak repairs is an outlier and deciding the existence of (weak) repairs of other types is much harder (under common assumptions concerning the polynomial hierarchy).

Theorem 3. *Let $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$, where η is a set of integrity constraints, and P an acyclic Horn program, and let $\mathscr{S}$ be a request set. The problems of deciding whether there is a relevant weak repair and whether there is a relevant repair for $(\mathscr{D}, \mathscr{S})$ are Σ_2^P-complete.* □

The last result concerns constrained (weak) repairs. It provides an upper bound on the complexity of the problem of deciding the existence of constrained repairs. We conjecture that the upper bound is in fact tight but have not been able to prove it. We leave the problem for future work.

Theorem 4. *Let $\mathscr{D} = \langle \mathscr{I}, \eta, P \rangle$, where η is a set of integrity constraints, and P an acyclic Horn program, and let $\mathscr{S}$ be a request set. The problems of deciding whether there is a constrained weak repair and whether there is a constrained repair for $(\mathscr{D}, \mathscr{S})$ are in Σ_3^P.* □

6 Discussion and Conclusion

We presented a declarative framework for view updating and integrity constraint maintenance for indefinite databases. The framework is based on the notion of an indefinite deductive database. In our approach, the indefiniteness appears in the extensional database and is modeled by a single null value, consistent with the standards of database practice (a condition not followed by earlier works on the view-update problem over indefinite databases). We defined a precise semantics for indefinite deductive databases in terms of possible worlds. We used the framework to formulate and study the view-update problem. Exploiting the concept of minimality of change introduced by an update, we defined several classes of repairs, including relevant and constrained repairs, that translate an update request against a view into an update of the underlying database. Finally, we obtained several complexity results concerning the existence of repairs.

Our paper advances the theory of view updating in three main ways. First, it proposes and studies the setting where extensional databases are indefinite both before and after an update. While *introducing* indefiniteness to narrow down the class of potential repairs was considered before [5], the assumption there was that the initial extensional database was complete. That assumption substantially limits the applicability of the earlier results. Second, our paper proposes a more expressive model of an indefinite extensional database. In our model databases are determined by two sets of facts. The first set of facts specifies what is true and provides an upper bound to what might still be unknown. By CWA, everything else is false. The second set of facts lists exceptions to the "unknown range," that is, facts that according to the first set might be unknown but are actually false (exceptions). Third, our paper introduces two novel classes or repairs, relevant and constrained, that often substantially narrow down possible ways to fulfill an update request against a view. Relevant repairs do not introduce any new constants and minimize change. Constrained repairs in addition do not involve constants that are in some precise sense "replaceable" and, thus, not grounded in the problem specification.

We already discussed some earlier work on view updating in the introduction as a backdrop to our approach. Expanding on that discussion, we note that the view-update problem is closely related to abduction and is often considered from that perspective. Perhaps the first explicit connection between the two was made by Bry [1] who proposed to use deductive tools as a means for implementing the type of abductive reasoning required in updating views. That idea was pursued by others with modifications that depended on the class and the semantics of the views. For instance, Kakas and Mancarella [9] exploited in their work on view updates the abductive framework by Eshghi and Kowalski [4] and were the first to consider the stable-model semantics for views. Neither of the two works mentioned above was, however, concerned with the case of updates to views over indefinite databases. Console et al. [2] studied the case in which *requests* can involve variables. These variables are replaced by null values and, in this way, null values eventually end up in repaired databases. However, once there, they loose their null value status and are treated just as any other constants. Consequently, no reasoning over null values takes place, in particular, they have no special effect on the notion of minimality. None of the papers discussed studied the complexity of the view-update problem. Instead, the focus was on tailoring resolution-based deductive reasoning tools to handle abduction. Some results on the complexity of abduction for logic programs were obtained by Eiter, Gottlob and Leone [3]. However, again the setting they considered did not assume incompleteness in extensional databases.

Our paper leaves several questions for future work. First, we considered restricted classes of views. That suggests the problem to extend our complexity results to the full case of Horn programs and, later, stratified ones. Next, we considered a limited class of integrity constraints. Importantly, we disallowed tuple-generating constraints. However, once they are allowed, even a problem of repairing consistency in an extensional database becomes undecidable. A common solution in the database research is to impose syntactic restrictions on the constraints [8]. That suggests considering view-updating in the setting in which only restricted classes of constraints are allowed.

Acknowledgments. The third author was supported by the NSF grant IIS-0913459. We are grateful to Sergio Greco and Leopoldo Bertossi for helpful discussions.

References

1. Bry, F.: Intensional updates: Abduction via deduction. In: Proceedings of ICLP 1990, pp. 561–575. MIT Press, Cambridge (1990)
2. Console, L., Sapino, M.L., Dupré, D.T.: The role of abduction in database view updating. J. Intell. Inf. Syst. 4(3), 261–280 (1995)
3. Eiter, T., Gottlob, G., Leone, N.: Abduction from logic programs: Semantics and complexity. Theor. Comput. Sci. 189(1-2), 129–177 (1997)
4. Eshghi, K., Kowalski, R.A.: Abduction compared with negation by failure. In: Proceedings of ICLP 1989, pp. 234–254. MIT Press, Cambridge (1989)
5. Farré, C., Teniente, E., Urpí, T.: Handling Existential Derived Predicates in View Updating. In: Palamidessi, C. (ed.) ICLP 2003. LNCS, vol. 2916, pp. 148–162. Springer, Heidelberg (2003)
6. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of ICLP/SLP 1988, pp. 1070–1080 (1988)
7. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. 9(3/4), 365–386 (1991)
8. Greco, S., Spezzano, F., Trubitsyna, I.: Stratification criteria and rewriting techniques for checking chase termination. PVLDB 4(11), 1158–1168 (2011)
9. Kakas, A.C., Mancarella, P.: Database updates through abduction. In: Proceedings of the Sixteenth International Conference on Very Large Databases, pp. 650–661. Morgan Kaufmann Publishers Inc., San Francisco (1990)
10. Libkin, L.: A Semantics-Based Approach to Design of Query Languages for Partial Information. In: Thalheim, B. (ed.) Semantics in Databases 1995. LNCS, vol. 1358, pp. 170–208. Springer, Heidelberg (1998)
11. Mayol, E., Teniente, E.: Consistency preserving updates in deductive databases. IEEE TDKE 47(1), 61–103 (2003)
12. Reiter, R.: On closed world data bases. In: Gallaire, H., Minker, J. (eds.) Logic and Data Bases, pp. 55–76. Plenum Press (1978)
13. Teniente, E., Olivé, A.: Updating knowledge bases while maintaining their consistency. VLDB J. 4(2), 193–241 (1995)
14. Todd, S.: Automatic constraint maintenance and updating defined relations. In: IFIP Congress, pp. 145–148 (1977)
15. Türker, C., Gertz, M.: Semantic integrity support in sql: 1999 and commercial (object-)relational database management systems. VLDB J. 10(4), 241–269 (2001)
16. Ullman, J.D.: Principles of Database and Knowledge-Base Systems, vol. 1. Computer Science Press (1988)

Three-Valued Logics for Incomplete Information and Epistemic Logic

Davide Ciucci[1,2,⋆] and Didier Dubois[1]

[1] IRIT, Université Paul Sabatier
118 route de Narbonne, 31062 Toulouse cedex 9, France
[2] DISCo - Università di Milano – Bicocca
Viale Sarca 336 – U14, 20126 Milano, Italia

Abstract. There are several three-valued logical systems. They give the impression of a scattered landscape. The majority of the works on this subject gives the truth tables, sometimes an Hilbert style axiomatization in a basic propositional language and a completeness theorem with respect to those truth tables. We show that all the reasonable connectives in three-valued logics can be built starting from few of them. Nevertheless, the issue of the usefulness of each system in relation with the third truth value is often neglected. Here, we review the interpretations of the third truth value. Then, we focus on the *unknown* case, suggested by Kleene. We show that any formula in three-valued logics can be encoded as a fragment of an epistemic logic (formulae of modal depth 1, with modalities in front of literals), preserving all tautologies and inference rules. We study in particular, the translation of Kleene, Gödel, Łukasiewicz and Nelson logics. This work enables us to lay bare the limited expressive power of three-valued logics in uncertainty management.

1 Introduction

Classical Boolean logic has a remarkable advantage over many others: the definition of its connectives is not questionable, even if the truth values *true* (1) and *false* (0) can be interpreted in practice in different ways. Moreover, there is complete agreement on its model-based semantics. Its formal setting seems to ideally capture the "targeted reality". The situation is quite different with many-valued logics, where we replace the two truth values by an ordered set L with more than two truth values. The simplest case is three-valued logic where we add a single intermediate value denoted by $\frac{1}{2}$. Naively, we might think that three-valued logic should be as basic as Boolean logic: the set $\{0, \frac{1}{2}, 1\}$ is the most simple example of bipolar scale, isomorphic to the set of signs $\{-, 0, +\}$. However, there is quite a number of three-valued logics since the extension to three values of the Boolean connectives is not unique. Worse, there is no agreement on the interpretation of this third truth value in the literature. Several interpretations of such a third truth value have been proposed. Here is a (probably not exhaustive) list:

⋆ Supported by FP7-Marie Curie Action (IEF) n.276158.

1. *Possible*: the oldest interpretation due to Lukasiewicz [4]. Unfortunately, it seems to have introduced some confusion between modalities and truth values, that is still looming in some parts of the many-valued logic literature.
2. *half-true*: it is the natural understanding in fuzzy logic [13]: if it is true that a man whose height is 1.80 m. is tall and it is false that a man with height 1.60 m. is tall, we can think that it is half-true that a man whose height is 1.70 m. is tall. Then $\frac{1}{2}$ captures the idea of *borderline*.
3. *Undefined*: this vision is typical of the studies on recursive functions modelled by logical formulae and it can be found in Kleene works [16]. A formula is not defined if some of its arguments are out of its domain. So, in this case the third truth value has a contaminating effect.
4. *Unknown*: in the same book, Kleene suggests this alternative interpretation of the intermediate value. It is the most usual point of view outside the fuzzy set community. Unfortunately, it suffers from the confusion between truth value and epistemic state, which generates paradoxes [22,7], just like the Łukasiewicz proposal.
5. *Inconsistent*: in some sense, it is the dual of "unknown". Several works try to tame the contradiction by means of a truth value (Priest, Belnap, and some paraconsistent logics for instance). This standpoint has been criticized as also generating paradoxes [10,6].
6. *Irrelevant*: this point of view is similar to "undefined" but with the opposite effect: abstention. If a component of a formula has $\frac{1}{2}$ as truth value, the truth value of the whole formula is determined by the remaining components.

In the present work, we are interested in the fourth interpretation *unknown* of $\frac{1}{2}$ (so, also in the first one). The idea that *unknown* can be a truth value seems to originate from a common usage in natural language, creating a confusion between *true* and *certainly true* (or yet provable), *false* and *certainly false*. In the spoken language, saying "it is true" is often short for "I know that it is true". We mix up, in this way, the idea of truth *per se* with the assertion of truth. The latter reveals some information possessed by the speaker (its epistemic state). The value *unknown* is in conflict with *certainly true* and *certainly false*. It corresponds to an epistemic state where neither the truth nor the falsity of a Boolean proposition can be asserted. The truth values *true* (1) and *false* (0) are of ontological nature (which means that they are part of the definition of what we call *proposition*[1]), whereas *unknown* as well as *certainly true* and *certainly false* have an epistemic nature: they reflect a knowledge state. Mathematically, *certainly true*, *certainly false* and *unknown* are subsets of truth values, that is $\{1\}$, $\{0\}$ and $\{0, 1\}$. The convention used here is the following: a subset $T \subseteq L$ of truth values attached to a proposition p contains the truth values that are possible for p in the considered knowledge state (the values outside T are impossible)[2]. For instance, $\{1\}$ encodes *certainly true* since the only possible truth value is *true*. Mixing up *true* and *certainly true* is the same as confusing an element with a

[1] And not that they represent Platonist ideals.

[2] Belnap[2] follows another convention: T represents a conjunction of truth values. Then, $\{0, 1\}$ encodes contradiction and the empty set represents *unknown*.

singleton. The use of qualifiers such as *certainly* immediately suggests the use of modal logic, just as Łukasiewicz "truth value" *possible* does. Clearly, *unknown* means that true and false are *possible*[3]. Already in 1921, Tarski had the idea of translating the modalities *possible* and *necessary* into Łukasiewicz three-valued logic. The modal *Possible* is defined on $\{0, \frac{1}{2}, 1\}$ as $\Diamond x = \neg x \rightarrow_L x = \min\{2x, 1\}$ with Łukasiewicz negation and implication. In this translation, *possible* thus means that the truth value is at least $\frac{1}{2}$. So the question is: which of the two is the most expressive language? modal logic or three-valued logic?

To address this question, we adopt the opposite point of view: rather than trying to translate modal logic into a three-valued one, it seems more fruitful to do the converse. According to the discussion of the epistemic nature of the interpretation of $\frac{1}{2}$ here chosen, the framework of some epistemic logic looks convenient. In particular, and we will show this in the following, it is more expressive than all the three-valued logics of *unknown*, the interest of which proves marginal in practice.

The paper develops as follows: we recall an elementary variant of epistemic logic, sufficient for our translation. It is a fragment of the logic KD, where we can express only Boolean propositional formulae prefixed by a modality (nesting of modalities is not allowed). It has a simple semantics in term of subsets of interpretations. We show how it is possible to translate propositions of the form "the truth value of three-valued proposition ϕ is in $T \subseteq \{0, \frac{1}{2}, 1\}$" by a Boolean modal formula. In the following section, we explain that only very few connectives are required to generate all the other connectives known in three-valued logics (essentially the minimum on $\{0, \frac{1}{2}, 1\}$ and its residuated implication, as well as an involutive negation). Some three-valued logics like Łukasiewicz's can thus express all the others. In the remaining sections, we consider several three-valued logics and translate them into MEL. We show that the tautologies of one are theorems of the other and the converse. On the contrary, the converse translation is impossible: only a fragment of MEL can be translated into a three-valued logic, the one where modalities are placed only in front of literals. In particular, Tarski's translation from $\Diamond\phi$ into $\neg\phi \rightarrow_L \phi$ is valid only if ϕ is a literal.

2 Connectives in Three-Valued Logics

According to the discussion in the introduction, we must not use the same symbols for Boolean truth values and the ones of the three-valued logic as long as $\frac{1}{2}$ means *unknown*, since the latter should be seen as subsets of the former. We will use 0 and 1 in the Boolean case and $\mathbf{3} = \{\mathbf{0}, \mathbf{1}, \frac{1}{2}\}$ in three-valued logics. Since $\frac{1}{2}$ is interpreted as *unknown*, $\mathbf{0}, \mathbf{1}, \frac{1}{2}$ will be considered as epistemic truth-values and 0, 1 as ontic ones. Moreover, we equip $\mathbf{3}$ with a total order $\leq$: $\mathbf{0} < \frac{1}{2} < \mathbf{1}$, often referred to as the truth ordering [2].

Conjunction, implication and negation on the set of values $\mathbf{0}, \frac{1}{2}, \mathbf{1}$ can be defined by minimal intuitive properties.

[3] Actually, Lukasiewicz proposed this idea for the study of contingent futures: it is possible that the battle will be won or lost.

Definition 1. *A conjunction on* $\mathbf{3}$ *is a binary mapping* $*$: $\mathbf{3} \times \mathbf{3} \mapsto \mathbf{3}$ *such that*

(C1) If $x \leq y$ *then* $x * z \leq y * z$*;*
(C2) If $x \leq y$ *then* $z * x \leq z * y$*;*
(C3) $\mathbf{0} * \mathbf{0} = \mathbf{0} * \mathbf{1} = \mathbf{1} * \mathbf{0} = \mathbf{0}$ *and* $\mathbf{1} * \mathbf{1} = \mathbf{1}$.

We note that (C3) requires that $*$ be an extension of the connective AND in Boolean logic. Then, the monotonicity properties (C1-C2) imply $\frac{1}{2} * \mathbf{0} = \mathbf{0} * \frac{1}{2} = \mathbf{0}$. If we consider all the possible cases, there are 14 conjunctions satisfying definition 1. Among them, only six are commutative and only five associative. These five conjunctions are already known in literature and precisely, they have been studied in the following logics: Sette [19], Sobociński [20], Łukasiewicz [4], Kleene [16], Bochvar [3]. The idempotent and commutative Kleene conjunction and disjunction (the minimum, denoted by $\sqcap$ and the maximum denoted by $\sqcup$) are present in $\mathbf{3}$ due the total order assumption ($x \sqcap y = y \sqcap x = x$ if and only if $x \leq y$ if and only if $x \sqcup y = y \sqcup x = y$).

In the case of implication, we can give a general definition, which extends Boolean logic and supposes monotonicity (decreasing in the first argument, increasing in the second).

Definition 2. *An implication on* $\mathbf{3}$ *is a binary mapping* $\rightarrow$: $\mathbf{3} \times \mathbf{3} \mapsto \mathbf{3}$ *such that*

(I1) If $x \leq y$ *then* $y \rightarrow z \leq x \rightarrow z$*;*
(I2) If $x \leq y$ *then* $z \rightarrow x \leq z \rightarrow y$*;*
(I3) $\mathbf{0} \rightarrow \mathbf{0} = \mathbf{1} \rightarrow \mathbf{1} = \mathbf{1}$ *and* $\mathbf{1} \rightarrow \mathbf{0} = \mathbf{0}$.

From the above definition we derive $x \rightarrow \mathbf{1} = \mathbf{1}$, $\mathbf{0} \rightarrow \mathbf{1} = \mathbf{1}$ and $\frac{1}{2} \rightarrow \frac{1}{2} \geq \{\mathbf{1} \rightarrow \frac{1}{2}, \frac{1}{2} \rightarrow \mathbf{0}\}$. There are 14 implications satisfying this definition. Nine of them are known and have been studied. Besides those implications named after the five logics mentioned above, there are also those named after Jaśkowski [15], Gödel [12], Nelson [17], Gaines-Rescher [11]. Gödel implication is present in $\mathbf{3}$ due to the total order and using the residuation:

$$x \sqcap y \leq z \text{ if and only if } x \leq y \rightarrow_G z.$$

It is such that $y \rightarrow_G z = \mathbf{1}$ if $y \leq z$ and z otherwise.

Finally, there are only three possible negations that extend the Boolean negation, that is if $\mathbf{0}' = \mathbf{1}$ and $\mathbf{1}' = \mathbf{0}$:

1. $\sim\frac{1}{2} = \mathbf{0}$. It corresponds to an intuitionistic negation (it satisfies the law of contradiction, not the excluded middle).
2. $\neg\frac{1}{2} = \frac{1}{2}$. It is an involutive negation.
3. $-\frac{1}{2} = \mathbf{1}$. It corresponds to a paraconsistent negation (as it satisfies the law of excluded middle, not the one of contradiction).

The intuitionistic negation is definable by Gödel implication as $\sim x = x \rightarrow_G 0$. Finally, despite the existence of several known systems of three-valued logics, we can consider that in the above setting, there is only one encompassing three-valued structure. That is, all the connectives satisfying the above definitions, can be obtained from a structure equipped with few primitive connectives [5].

Proposition 1. *We denote by* $\mathbf{3}$ *the set of three elements without any structure and by* $\overline{3}$ *the same set equipped with the usual order* $\mathbf{0} < \frac{1}{2} < \mathbf{1}$ *or equivalently,* $\overline{3} = (\mathbf{3}, \sqcap, \rightarrow_G)$. *All 14 conjunctions and implications can be expressed in any of the following systems:*

- $(\overline{3}, \neg) = (\mathbf{3}, \sqcap, \rightarrow_G, \neg)$*;*
- $(\overline{3}, \rightarrow_K)$ *where* $\rightarrow_K$ *is Kleene implication (*$\max(1-x, y)$*);*
- $(\mathbf{3}, \rightarrow_L, \mathbf{0})$ *where* $\rightarrow_L$ *is Łukasiewicz implication (*$\min(1, 1-x+y)$*);*
- $(\mathbf{3}, \rightarrow_K, \sim, \mathbf{0})$ *where* $\rightarrow_K$ *is Kleene implication and* $\sim$ *the intuitionistic negation.*

So, in the first two cases, we assume a residuated chain, whereas in the other two, we can derive it from the other connectives. We remark also that the intuitionistic negation can be replaced by the paraconsistent negation in the last item. The above result differs from functional completeness, since it only deals with three-valued functions that coincide with a Boolean connective on $\{T, F\}$.

3 A Simple Information Logic

Admitting that the concept of "unknown" refers to a knowledge state rather than to an ontic truth value, we may keep the logic Boolean and add to its syntax the capability of stating that we ignore the truth value (1 or 0) of propositions. The natural framework to syntactically encode statements about knowledge states of propositional logic (PL) statements is modal logic, and in particular, the logic KD. Nevertheless, only a very limited fragment of this language is needed here: the language MEL [1].

Consider a set of propositional variables $\mathcal{V} = \{a, b, c, \dots, p, \dots\}$ and a standard propositional language $\mathcal{L}$ built on these symbols along with the Boolean connectives of conjunction and negation $(\wedge, ')$. As usual, disjunction $\alpha \vee \beta$ stands for $(\alpha' \wedge \beta')'$, implication $\alpha \rightarrow \beta$ stands for $\alpha' \vee \beta$, and tautology $\top$ for $\alpha \vee \alpha'$. Let us build another propositional language $\mathcal{L}_\Box$ whose set of propositional variables is of the form $\mathcal{V}_\Box = \{\Box\alpha : \alpha \in \mathcal{L}\}$ to which the classical connectives can be applied. It is endowed with a modality operator $\Box$ expressing certainty, that encapsulates formulas in $\mathcal{L}$. We denote by $\alpha, \beta, \dots$ the propositional formulae of $\mathcal{L}$, and $\phi, \psi, \dots$ the modal formulae of $\mathcal{L}_\Box$. In other words

$$\mathcal{L}_\Box = \Box\alpha : \alpha \in \mathcal{L} | \neg\phi | \phi \wedge \psi | \phi \vee \psi | \phi \rightarrow \psi.$$

The logic MEL [1] uses the language $\mathcal{L}_\Box$ with the following axioms:

1. $\phi \rightarrow (\psi \rightarrow \phi)$
2. $(\psi \rightarrow (\phi \rightarrow \mu)) \rightarrow ((\psi \rightarrow \phi) \rightarrow (\psi \rightarrow \mu))$
3. $(\phi' \rightarrow \psi') \rightarrow (\psi \rightarrow \phi)$

(RM) : $\Box\alpha \rightarrow \Box\beta$ if $\vdash \alpha \rightarrow \beta$ in PL.
(M) : $\Box(\alpha \wedge \beta) \rightarrow (\Box\alpha \wedge \Box\beta)$
(C) : $(\Box\alpha \wedge \Box\beta) \rightarrow \Box(\alpha \wedge \beta)$
(N) : $\Box\top$

(D) : $\Box\alpha \rightarrow \Diamond\alpha$

and the inference rule is modus ponens. As usual, the *possible* modality $\Diamond$ is defined as $\Diamond\alpha \equiv (\Box\alpha')'$. The first three axioms are those of PL and the other those of modal logic KD. (M) and (C) can be replaced by axiom (K):

$$(K) : \Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q).$$

MEL is the subjective fragment of KD (or S5) without modality nesting.

The MEL semantics is very simple [1]. Let Ω be the set of $\mathcal{L}$ interpretations: $\{\omega : \mathcal{V} \rightarrow \{0,1\}\}$. The set of models of α is $[\alpha] = \{\omega : \omega \models \alpha\}$. A (meta)-interpretation of $\mathcal{L}_\Box$ is a non-empty set $E \subseteq \Omega$ of interpretations of $\mathcal{L}$ interpreted as an epistemic state. We define satisfiability as follows:

- $E \models \Box\alpha$ if $E \subseteq [\alpha]$ (α is certainly true in the epistemic state E)
- $E \models \phi \wedge \psi$ if $E \models \phi$ and $E \models \psi$;
- $E \models \phi'$ if $E \models \phi$ is false.

MEL is sound and complete with respect to this semantics [1].

We remark that in this framework, uncertainty modeling is Boolean but possibilistic. The satisfiability $E \models \Box\alpha$ can be written as $N([\alpha]) = 1$ in the sense of a necessity measure computed with the possibility distribution given by the characteristic function of E. Axioms (M) and (C) lay bare the connection with possibility theory [7], as they state the equivalence between $(\Box\alpha \wedge \Box\beta)$ and $\Box(\alpha \wedge \beta)$. We can justify the choice of this minimal modal formalism. It is the most simple logic to reason on incomplete propositional information[4]. We only need to express that a proposition in PL is certainly true, certainly false or unknown as well as all the logical combinations of these assertions.

4 The Principles of the Translation

We denote by $v(a) \in \mathbf{3}$ the epistemic truth value of the variable $a \in \mathcal{L}$. The assertion $v(a) \in T \subseteq \mathbf{3}$ informs about the knowledge state of a Boolean variable which we also denote by a. Stricto sensu, we should not use the same notation for three-valued propositional variables and Boolean ones. However, we will do it for the sake of simplicity assuming that a possesses an epistemic truth value $v(a)$ and an ontic truth-value $t(a) \in \{0,1\}$. If we interpret the three epistemic truth-values $\mathbf{0}, \mathbf{1}, \frac{1}{2}$ as *certainly true*, *certainly false* and *unknown* respectively, we can encode the assignment of such truth-values to a propositional variable a in MEL as follows. We denote by $\mathcal{T}(v(a) \in T)$ the translation into MEL of

[4] We can debate whether MEL is an epistemic or a doxastic logic. This formalism does not take side, since axiom D is valid in both S5 and KD45 and the axiom T of knowledge ($\Box\alpha \rightarrow \alpha$) is not expressible in MEL, which is the subjective fragment of S5 as much as a KD45 fragment. We kept the term "epistemic" in reference to the idea of an information state, whether it is consistent with reality or not.

the statement $v(a) \in T$ and define it as follows, in agreement with the intended meaning of the epistemic truth-values:

$$\mathcal{T}(v(a) = \mathbf{1}) = \Box a \qquad \mathcal{T}(v(a) = \mathbf{0}) = \Box a' \qquad \mathcal{T}(v(a) = \tfrac{1}{2}) = \Diamond a \wedge \Diamond a'$$
$$\mathcal{T}(v(a) \geq \tfrac{1}{2}) = \Diamond a \qquad \mathcal{T}(v(a) \leq \tfrac{1}{2}) = \Diamond a'$$

We remark that these definitions shed light on the acceptability or not of the excluded middle law and the contradiction principle in the presence of the *unknown* value: a is always ontologically true or false, but $\Box a \vee \Box a'$ is not a tautology nor $\Diamond a \wedge \Diamond a'$ a contradiction in MEL.

In parallel we can express 3-valued valuations in the form of special epistemic states that serve as interpretations of the language $\mathcal{L}_\Box$ of MEL. Given a 3-valued valuation v, denote by E_v, the partial Boolean model defined by $t(a) = 1$ if and only if $v(a) = \mathbf{1}$ and $t(a) = 0$ if and only if $v(a) = \mathbf{0}$. Such an epistemic state E_v has a particular form: it is the set of propositional interpretations of a non-contradictory conjunction of literals $\wedge_{v(a)=\mathbf{1}} a \bigwedge \wedge_{v(a)=\mathbf{0}} a'$. Conversely, for any MEL epistemic state E (a disjunction of propositional interpretations) we can assign a single 3 valued interpretation v_E:

$$\forall a, v_E(a) = \begin{cases} \mathbf{1} & E \vDash \Box a \\ \mathbf{0} & E \vDash \Box a' \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

The map $E \mapsto v_E$ is not bijective, it defines an equivalence relation on epistemic states and $E_v = \cup\{E : v_E = v\}$ is the partial Boolean model induced by v. Let us now consider the fragment of MEL where we can put modalities only in front of literals, that is $\mathcal{L}^\ell_\Box = \Box a | \Box a' | \phi' | \phi \wedge \psi | \phi \vee \psi$.

Proposition 2. *Let α be a formula and Γ a set of formulae built on the language of the fragment $\mathcal{L}^\ell_\Box$. Then, $\Gamma \vdash \alpha$ if and only if $\forall v, E_v \models \Gamma$ implies $E_v \models \alpha$.*

In the following, we consider four known three-valued logics and show that, insofar as the third truth-value means *unknown*, they can be expressed in MEL: Kleene, Gödel three-valued intuitionistic, Łukasiewicz and Nelson logics. The two first ones can be viewed as fragments of the latter. Especially, we show that $\mathcal{L}^\ell_\Box$ exactly characterizes any of Łukasiewicz and Nelson logics as we will see in the next sections. We start with the simplest logic.

5 The Kleene Fragment of MEL

The best known and often used logic to represent uncertainty due to incomplete information is Kleene logic. The connectives are simply the min $\sqcap$, the max $\sqcup$, the involutive negation $\neg$. A material implication $a \rightarrow_K b := \neg a \sqcup b$ is then derived. The involutive negation preserves the De Morgan laws between $\sqcap$ and $\sqcup$. The syntax of Kleene logic is the same as the one of propositional logic (replacing $\wedge, \vee, '$ by $\sqcap, \sqcup, \neg$). Besides, it is known that Kleene logic does not have any tautology (there is no formula α such that $\forall v, v(\alpha) = \mathbf{1}$).

We first show that all the assignments of epistemic truth values to Kleene formulae, in the form $v(\alpha) \in T \subseteq \mathbf{3}$ can be translated into MEL. Using the translation of atoms in Section 4, the translation of other formulae is

$$\begin{aligned}
\mathcal{T}(v(\alpha \sqcap \beta) \geq i) &= \mathcal{T}(v(\alpha) \geq i) \wedge \mathcal{T}(v(\beta) \geq i), i \geq \tfrac{1}{2} \\
\mathcal{T}(v(\alpha \sqcup \beta) \geq i) &= \mathcal{T}(v(\alpha) \geq i) \vee \mathcal{T}(v(\beta) \geq i), i \geq \tfrac{1}{2} \\
\mathcal{T}(v(\alpha \sqcap \beta) \leq i) &= \mathcal{T}(v(\alpha) \leq i) \vee \mathcal{T}(v(\beta) \leq i), i \leq \tfrac{1}{2} \\
\mathcal{T}(v(\alpha \sqcup \beta) \leq i) &= \mathcal{T}(v(\alpha) \leq i) \wedge \mathcal{T}(v(\beta) \leq i), i \leq \tfrac{1}{2} \\
\mathcal{T}(v(\neg\alpha) = \mathbf{1}) &= \mathcal{T}(v(\alpha) = \mathbf{0}) = (\mathcal{T}(v(\alpha) \geq \tfrac{1}{2}))' \\
\mathcal{T}(v(\neg\alpha) \geq \tfrac{1}{2}) &= \mathcal{T}(v(\alpha) \leq \tfrac{1}{2}) = (\mathcal{T}(v(\alpha) = \mathbf{1}))'
\end{aligned}$$

The translation of Kleene implication can be obtained in this way; we can define it directly as follows using standard material implication $\to$.

$$\begin{aligned}
\mathcal{T}(v(\alpha \to_K \beta) = \mathbf{1}) &= \mathcal{T}(v(\alpha) \geq \tfrac{1}{2}) \to \mathcal{T}(v(\beta) = \mathbf{1}) \\
\mathcal{T}(v(\alpha \to_K \beta) \geq \tfrac{1}{2}) &= \mathcal{T}(v(\alpha) = \mathbf{1}) \to \mathcal{T}(v(\beta) \geq \tfrac{1}{2})
\end{aligned}$$

If $\alpha = a$, $\beta = b$ are atoms, we obtain $\Box\neg a \vee \Box b$ and $\Diamond\neg a \vee \Diamond b$ respectively. The translation into MEL lays bare the meaning of Kleene implication: $a \to_K b$ is "true" means that b is certain if a is possible.

Example 1. Consider the formula $\alpha = \neg(a \sqcap (\neg(b \sqcup \neg c)))$. Then, $\mathcal{T}(v(\alpha) = \mathbf{1}) = \mathcal{T}(v(a \sqcap (\neg(b \sqcup \neg c))) = \mathbf{0})$. So, we get $\mathcal{T}(v(a) = \mathbf{0}) \vee \mathcal{T}(v(\neg(b \sqcup \neg c)) = \mathbf{0}) = \Box a' \vee \mathcal{T}(v(b \sqcup \neg c) = \mathbf{1})$ and finally, $\Box a' \vee \mathcal{T}(v(b) = 1) \vee \mathcal{T}(v(\neg c) = \mathbf{1}) = \Box a' \vee \Box b \vee \Box c'$. Note that we could more simply put α in conjunctive normal form as $\neg a \vee b \vee \neg c$ then put $\Box$ in front of each literal.

A knowledge base B in Kleene logic is a conjunction of formulae supposed to have designated truth value $\mathbf{1}$. We can always transform this base in conjunctive normal form (CNF), that is, a conjunction of disjunction of literals (without simplifying terms of the form $a \sqcup \neg a$). Its translation into MEL consists of the same set of clauses, where we put the modality $\Box$ in front of each literal. It is easy to see that the translation of any propositional tautology (if we replace each literal l by $\Box l$ in its CNF) will no longer be a tautology in MEL.

Finally we see that the fragment of MEL that exactly captures the language of Kleene logic contains only the set (conjunctions) of disjunctions of the elementary formulae of the form $\Box a$ or $\Box a'$: $\mathcal{L}_{\Box}^{K} = \Box a | \Box a' | \phi \vee \psi | \phi \wedge \psi \subset \mathcal{L}_{\Box}^{\ell}$. We remark that the modal axioms of MEL cannot be expressed in this fragment.

Nevertheless, we can use MEL to reason in Kleene logic. We note that the modus ponens applies to literals (since from $\Box a$ and $\Box a' \vee \Box b$, we can derive $\Box b$ in MEL). The same counterpart of the resolution principle is also valid. It is like a propositional logic without tautologies but with such standard rules of inference. At the semantic level we can prove the following result.

Proposition 3. *Let α be a formula in Kleene logic. For each model v of α, the epistemic state E_v is a model (in the sense of MEL) of $\mathcal{T}(v(\alpha) = \mathbf{1})$. Conversely,* *for each model in the sense of MEL (epistemic state) E of* $\mathcal{T}(v(\alpha) = \mathbf{1})$ *the 3-valued interpretation* v_E *is a model of* α *in the sense that* $v_E(\alpha) = 1$.

We can easily verify the completeness of the Kleene fragment in MEL with respect to the models of the form E_v in the sense that $\mathcal{T}(B) \vdash \mathcal{T}(v(\alpha) = \mathbf{1})$ in MEL if and only if $\forall v, E_v \vDash \mathcal{T}(B)$ implies $E_v \vDash \mathcal{T}(v(\alpha) = \mathbf{1})$.

6 From Łukasiewicz Three-Valued Logic to MEL and Back

Łukasiewicz three-valued logic Ł3 is a language powerful enough to express all connectives laid bare in section 2. It has been axiomatized by M. Wajsberg [24], using a language based on $(\mathcal{V}, \rightarrow_L, \neg)$, the modus ponens rule and the following axioms:

(W1) $(\alpha \rightarrow_L \beta) \rightarrow_L ((\beta \rightarrow_L \gamma) \rightarrow_L (\alpha \rightarrow_L \gamma))$
(W2) $\alpha \rightarrow_L (\beta \rightarrow_L \alpha)$
(W3) $(\neg\beta \rightarrow_L \neg\alpha) \rightarrow_L (\alpha \rightarrow_L \beta))$
(W4) $(((\alpha \rightarrow_L \neg\alpha) \rightarrow_L \alpha) \rightarrow_L \alpha)$

The truth-table of the implication $\rightarrow_L$ is given by table 1 and the involutive negation of Kleene logic is recovered as $\neg a := a \rightarrow_L 0$. We can also define two

Table 1. Łukasiewicz implication, conjunction and disjunction truth tables

$\rightarrow_L$	0	$\frac{1}{2}$	1
0	1	1	1
$\frac{1}{2}$	$\frac{1}{2}$	1	1
1	0	$\frac{1}{2}$	1

$\odot$	0	$\frac{1}{2}$	1
0	0	0	0
$\frac{1}{2}$	0	0	$\frac{1}{2}$
1	0	$\frac{1}{2}$	1

$\oplus$	0	$\frac{1}{2}$	1
0	0	$\frac{1}{2}$	1
$\frac{1}{2}$	$\frac{1}{2}$	1	1
1	1	1	1

pairs of conjunction and disjunction connectives: $(\sqcap, \sqcup)$ and $(\odot, \oplus)$. The former pair is Kleene's, recovered as $a \sqcup b = (a \rightarrow_L b) \rightarrow_L b$ and $a \sqcap b = \neg(\neg a \sqcup \neg b)$. The other pair is $a \oplus b := \neg a \rightarrow_L b$ and $a \odot b := \neg(\neg a \oplus \neg b)$ explicitly described in Table 1. Łukasiewicz implication is translated into MEL as:

$$\begin{aligned}\mathcal{T}(v(\alpha \rightarrow_L \beta) = \mathbf{1}) = &[\mathcal{T}(v(\alpha) = \mathbf{1}) \rightarrow \mathcal{T}(v(\beta) = \mathbf{1})]\\ &\wedge [\mathcal{T}(v(\alpha) \geq \tfrac{1}{2}) \rightarrow \mathcal{T}(v(\beta) \geq \tfrac{1}{2})]\\ \mathcal{T}(v(\alpha \rightarrow_L \beta) \geq \tfrac{1}{2}) = &\mathcal{T}(v(\alpha) = \mathbf{1}) \rightarrow \mathcal{T}(v(\beta) \geq \tfrac{1}{2})\end{aligned}$$

The translation of $\mathcal{T}(v(\alpha \rightarrow_L \beta) = \mathbf{1})$ is the same for all the 3-valued residuated implications. In the case of atoms, we have $\mathcal{T}(v(a \rightarrow_L b) \geq \frac{1}{2}) = (\Box a)' \vee \Diamond b$ and $\mathcal{T}(v(a \rightarrow_L b) = \mathbf{1}) = ((\Box a)' \vee \Box b) \wedge ((\Diamond a)' \vee \Diamond b) = \Box a' \vee \Box b \vee ((\Box a)' \wedge \Diamond b))$. $\mathcal{T}(v(a \rightarrow_L b) = \mathbf{1})$ thus means : if a is certain then so is b and if a is possible then so is b. The translation of the connectives $\odot$ and $\oplus$ is:

$$\mathcal{T}(v(\alpha \oplus \beta) = \mathbf{1}) = \mathcal{T}(v(\alpha) = \mathbf{1}) \vee \mathcal{T}(v(\beta) = \mathbf{1}) \vee (\mathcal{T}(v(\alpha) \geq \tfrac{1}{2}) \wedge \mathcal{T}(v(\beta) \geq \tfrac{1}{2}))$$
$$\mathcal{T}(v(\alpha \oplus \beta) \geq \tfrac{1}{2}) = \mathcal{T}(v(\alpha) \geq \tfrac{1}{2}) \vee \mathcal{T}(v(\beta) \geq \tfrac{1}{2})$$

$\mathcal{T}(v(\alpha \odot \beta) = \mathbf{1}) = \mathcal{T}(v(\alpha) = \mathbf{1}) \wedge \mathcal{T}(v(\beta) = \mathbf{1})$
$\mathcal{T}(v(\alpha \odot \beta) \geq \frac{1}{2}) = [\mathcal{T}(v(\alpha) \geq \frac{1}{2}) \wedge \mathcal{T}(v(\beta) = \mathbf{1})] \vee [\mathcal{T}(v(\alpha) = \mathbf{1}) \wedge \mathcal{T}(v(\beta) \geq \frac{1}{2})]$

For the atoms, we see that $\mathcal{T}(v(a \oplus b) = \mathbf{1}) = \Box a \vee \Box b \vee (\Diamond a \wedge \Diamond b)$ and $\mathcal{T}(v(\alpha \odot \beta) = \mathbf{1}) = \Box a \wedge \Box b$. Note that while the truth of Kleene disjunction $a \sqcup b$ corresponds to the requirement that one of a and b be certain, $a \oplus b$ is closer to the usual meaning of the disjunction, whereby $\mathcal{T}(v(a \oplus b) = \mathbf{1})$ can be true with none of a or b being certain. Besides, asserting the truth of a conjunction in Ł$_3$ leads to the same translation for the two conjunctions. Note that in Ł$_3$ the top and bottom element are translated respectively into, $((\Box a)' \vee \Box a) \wedge ((\Diamond a)' \vee \Diamond a)$ and $\Box a \wedge (\Diamond a)'$ which are indeed tautologies and contradictions in MEL, respectively. More generally:

Proposition 4. *If α is an axiom in Ł$_3$, then $\mathcal{T}(v(\alpha) = \mathbf{1})$ is a tautology in MEL.*

The syntactic fragment of MEL capable of expressing Ł$_3$ is: $\Box a | \Box a' | \phi' | \phi \wedge \psi | \phi \vee \psi$, that is the MEL fragment $\mathcal{L}_{\Box}^{\ell}$ where modalities are just in front of literals. It is clear that $\mathcal{L}_{\Box}^{K}$ is a fragment of $\mathcal{L}_{\Box}^{\ell}$. From $\mathcal{L}_{\Box}^{\ell}$ to Ł$_3$, we can also prove:

Proposition 5. *For any formula in $\phi \in \mathcal{L}_{\Box}^{\ell}$, there exists a formula α in Ł$_3$ such that ϕ is logically equivalent to $\mathcal{T}(v(\alpha) = \mathbf{1})$ in MEL. In particular, if ϕ is a MEL axiom in $\mathcal{L}_{\Box}^{\ell}$, then the corresponding formula α is a tautology in Ł$_3$.*

Proof. Sketch. We just show that there exists a translation θ from $\mathcal{L}_{\Box}^{\ell}$ to Łukasiewicz logic, recursively defined as: $\theta(\Box a) = a$, $\theta(\Box a') = \neg a$, $\theta(\Diamond a') = a \rightarrow_L \neg a$, $\theta(\Diamond a) = \neg a \rightarrow_L a$, $\theta(\alpha \wedge \beta) = \theta(\alpha) \sqcap \theta(\beta)$, $\theta(\alpha \vee \beta) = \theta(\alpha) \sqcup \theta(\beta)$.

At the semantic level, Proposition 3 extends to Ł$_3$. Moreover, since the sublanguage $\mathcal{L}_{\Box}^{\ell}$ is exactly the Łukasiewicz fragment of MEL, we can apply Proposition 2 and obtain the completeness of this fragment of MEL with respect to the models of the form E_v in the sense that, given a knowledge base $B_L$in Ł$_3$ (a conjunction of Ł$_3$ formulas) $\mathcal{T}(B_L) \vdash \mathcal{T}(v(\alpha) = \mathbf{1})$ in MEL if and only if $\forall v, E_v \vDash \mathcal{T}(B_L)$ implies $E_v \vDash \mathcal{T}(v(\alpha) = \mathbf{1})$. Finally, we can prove that MEL restricted to $\mathcal{L}_{\Box}^{\ell}$ is the proper target language for reasoning in Ł$_3$, adopting an epistemic stance for truth-values. Indeed, from the above results we get the following.

Proposition 6. *Let α be a formula in Łukasiewicz logic Ł$_3$ and B_L a knowledge base in this logic. Then, $B_L \vdash \alpha$ in Ł$_3$ iff $\mathcal{T}(B_L) \vdash \mathcal{T}(v(\alpha) = \mathbf{1})$ in MEL .*

Proof. Sketch: both MEL and Łukasiewicz logic are sound and complete. So, it is enough to show that $B_L \vDash_{\text{Ł}} \alpha$ iff $\mathcal{T}(B_L) \vDash_{MEL} \mathcal{T}(v(\alpha) = \mathbf{1})$. One direction is the extension of Proposition 3 to the present case and the other follows by induction.

We note that all the results about Łukasiewicz logic also apply to the three-valued Nelson logic [23] $N_3 = (\mathcal{V}, \sqcap, \sqcup, \rightarrow_N, \neg, -)$ due to the equivalence of the two logics. Indeed, Nelson implication is defined by Łukasiewicz implication as $a \rightarrow_N b := a \rightarrow_L (a \rightarrow_L b)$ and Łukasiewicz implication can be defined as

$a \rightarrow_L b := (a \rightarrow_N b) \sqcap (\neg b \rightarrow_N \neg a)$. So, Nelson implication, once translated in the fragment $\mathcal{L}^{\ell}_{\Box}$ of MEL is defined (on the atoms) as $\mathcal{T}(v(a \rightarrow_N b) = \mathbf{1}) = (\Box a)' \vee \Box b$, which says that if α is certain then β is certain. This implication may look more natural in MEL than residuated ones or Kleene's.

7 Encoding Three-Valued Intuitionistic Logic into MEL

The three-valued Gödel logic [12] is based on the language built from the 4-tuple $(\mathcal{V}, \rightarrow_G, \sqcap, \sim)$, and the axioms are

(I1) $\alpha \rightarrow_G (\beta \rightarrow_G \alpha)$
(I2) $(\alpha \rightarrow_G (\beta \rightarrow_G \gamma)) \rightarrow_G ((\alpha \rightarrow_G \beta) \rightarrow_G (\alpha \rightarrow_G \gamma)$
(I3) $(\alpha \sqcap \beta) \rightarrow_G \alpha$
(I4) $(\alpha \sqcap \beta) \rightarrow_G \beta$
(I5) $\alpha \rightarrow_G (\beta \rightarrow_G (\alpha \sqcap \beta))$
(I6) $\alpha \rightarrow_G (\alpha \sqcup \beta)$
(I7) $\beta \rightarrow_G (\alpha \sqcup \beta)$
(I8) $(\alpha \rightarrow_G \beta) \rightarrow_G ((\gamma \rightarrow_G \beta) \rightarrow_G (\alpha \sqcup \gamma \rightarrow_G \beta))$
(I9) $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \sim\beta) \rightarrow (\sim\alpha))$
(I10) $\sim\alpha \rightarrow (\alpha \rightarrow \beta)$
(I11) $\alpha \sqcup (\sim \beta \sqcup (\alpha \rightarrow_G \beta))$

where $\rightarrow_G$ is the residuum of Kleene conjunction $\sqcap$, $\sim$ is the intuitionistic negation, and the Kleene disjunction $\sqcup$ is retrieved as $\alpha \sqcup \beta := [(\alpha \rightarrow_G \beta) \rightarrow_G \beta] \sqcap [(\beta \rightarrow_G \alpha) \rightarrow_G \alpha]$. The truth tables of the implication and negation are given by Table 2. Axiom (I10), due to Hosoi [14], ensures three-valuedness. The

Table 2. Truth table of Gödel implication and negation

$\rightarrow_G$	0	$\frac{1}{2}$	1
0	1	1	1
$\frac{1}{2}$	0	1	1
1	0	$\frac{1}{2}$	1

$\sim$	0
0	1
$\frac{1}{2}$	0
1	0

translation $\mathcal{T}(v(\sim \alpha) = \mathbf{1})$ in MEL of Gödel negation is the same as the translation of Kleene negation. The translation $\mathcal{T}(v(\alpha \rightarrow_G \beta) = \mathbf{1})$ is the same as for Łukasiewicz implication. However,

$$\mathcal{T}(v(\sim \alpha) = \mathbf{0}) = \mathcal{T}(v(\alpha) \geq \tfrac{1}{2})$$
$$\mathcal{T}(v(\alpha \rightarrow_G \beta) \geq \tfrac{1}{2}) = \mathcal{T}(v(\alpha) \geq \tfrac{1}{2}) \rightarrow \mathcal{T}(v(\beta) \geq \tfrac{1}{2})$$

In the case of atoms, $\mathcal{T}(v(a \rightarrow_G b) \geq \frac{1}{2}) = (\Diamond a)' \vee \Diamond b = \Box a' \vee \Diamond b$.

We note that the top element $\top = \alpha \rightarrow_G \alpha$ and the bottom element $\bot = \neg(\alpha \rightarrow_G \alpha)$ in Gödel logic translate into a tautology and to a contradiction in MEL. Their translation is the same as for Łukasiewicz logic Ł$_3$. More generally, we can justify the axioms of intuitionistic logic in MEL.

Proposition 7. *If α is an axiom of the three-valued Gödel logic, then $\mathcal{T}(v(\alpha) = \mathbf{1})$ is a tautology in MEL.*

Finding the syntactic fragment of MEL (or of KD) that exactly captures this three-valued logic is an open problem. It is contained in $\mathcal{L}_{\Box}^{\ell}$ and includes the formulas $\{\Box a, \Box a', \Diamond a, a \in \mathcal{V}\}$. However, Proposition 3 is still valid.

8 Conclusion

This work suggests that the multiplicity of three-valued logics is only apparent. If the third value means *unknown*, the elementary modal logic MEL, and more specifically its fragment where modalities appear only in front of literals, is a natural choice to encode all of these three-valued logics. In the framework of a given application, some connectives make sense, others do not and we can choose the proper fragment. The interest in our translation, which is both modular and faithful, is double:

1. Once translated into modal logic, the meaning of a formula becomes clear since its epistemic dimension is encoded in the syntax, even if in the worst case, the size of a translated formula may grow exponentially in the number of occurrences of the input variables.
2. We can better measure the expressive power of each three-valued system. In particular it shows that the truth-functionality of three-valued logic is paid by a severe restriction to expressing knowledge about literals only, and a very restrictive view of disjunction.

This work can be extended to more than 3 "epistemic" truth values. However, the target language is then a more expressive modal logic with more or less strong modalities, such as generalized possibilistic logic [9] (where the epistemic states are possibility distributions). For instance, the 5-valued so-called equilibrium logic [18] (which can encode "answer-set" programming) has been translated into generalized possibilistic logic with weak and strong necessity operators, the epistemic states being pairs of sets of nested models [8].

The idea of expressing a many-valued logic in a two-level Boolean language (one encapsulating the other) put here at work can be adapted to other interpretations of the third truth value (such as *contradictory, irrelevant*, etc.) by changing the target language. We can conjecture that only the case where this truth value has an ontic nature (that is *half-true*, admitting that truth is a matter of degree) enables to give a clear meaning to propositional languages using the syntax of Gödel, Łukasiewicz, etc. logics and to explain their violation of the Boolean axioms, as in the case of fuzzy logics.

References

1. Banerjee, M., Dubois, D.: A Simple Modal Logic for Reasoning about Revealed Beliefs. In: Sossai, C., Chemello, G. (eds.) ECSQARU 2009. LNCS (LNAI), vol. 5590, pp. 805–816. Springer, Heidelberg (2009)

2. Belnap, N.D.: A useful four-valued logic. In: Dunn, J.M., Epstein, G. (eds.) Modern Uses of Multiple-Valued Logic, pp. 8–37. D. Reidel (1977)
3. Bochvar, D.A.: On a three-valued logical calculus and its application to the analysis of the paradoxes of the classical extended functional calculus. History and Philosophy of Logic 2, 87–112 (1981)
4. Borowski, L. (ed.): Selected works of J. Łukasiewicz. North-Holland, Amsterdam (1970)
5. Ciucci, D., Dubois, D.: Relationships between Connectives in Three-Valued Logics. In: Greco, S., Bouchon-Meunier, B., Coletti, G., Fedrizzi, M., Matarazzo, B., Yager, R.R. (eds.) IPMU 2012, Part I. CCIS, vol. 297, pp. 633–642. Springer, Heidelberg (2012)
6. Dubois, D.: On ignorance and contradiction considered as truth-values. Logic Journal of the IGPL 16, 195–216 (2008)
7. Dubois, D., Prade, H.: Possibility theory, probability theory and multiple-valued logics: A clarification. Ann. Math. and AI 32, 35–66 (2001)
8. Dubois, D., Prade, H., Schockaert, S.: Stable models in generalized possibilistic logic. In: Proceedings KR 2012, Roma, pp. 519–529 (2012)
9. Dubois, D., Prade, H.: Generalized Possibilistic Logic. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 428–432. Springer, Heidelberg (2011)
10. Fox, J.: Motivation and demotivation of a four-valued logic. Notre Dame Journal of Formal Logic 31(1), 76–80 (1990)
11. Gaines, B.R.: Foundations of fuzzy reasoning. Int. J. of Man-Machine Studies 6, 623–668 (1976)
12. Gödel, K.: Zum intuitionistischen aussagenkalkül. Anzeiger Akademie der Wissenschaften Wien 69, 65–66 (1932)
13. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, Dordrecht (1998)
14. Hosoi, T.: The axiomatization of the intermediate propositional systems sn of gödel. J. Coll. Sci., Imp. Univ. Tokyo 13, 183–187 (1996)
15. Jaśkowski, S.: Propositional calculus for contradictory deductive systems. Studia Logica 24, 143–160 (1969)
16. Kleene, S.C.: Introduction to metamathematics. North–Holland Pub. Co., Amsterdam (1952)
17. Nelson, D.: Constructible falsity. J. of Symbolic Logic 14, 16–26 (1949)
18. Pearce, D.: Equilibrium logic. Annals of Mathematics and Artificial Intelligence 47, 3–41 (2006)
19. Sette, A.M.: On propositional calculus p_1. Math. Japon 16, 173–180 (1973)
20. Sobociński, B.: Axiomatization of a partial system of three-value calculus of propositions. J. of Computing Systems 1, 23–55 (1952)
21. Surma, S.: Logical Works. Polish Academy of Sciences, Wroclaw (1977)
22. Urquhart, A.: Many-valued logic. In: Gabbay, D.M., Guenthner, F. (eds.) Handbook of Philosophical Logic: vol. III, Alternatives to Classical Logic, Springer (1986)
23. Vakarelov, D.: Notes on n-lattices and constructive logic with strong negation. Studia Logica 36, 109–125 (1977)
24. Wajsberg, M.: Aksjomatyzacja trówartościowego rachunkuzdań (Axiomatization of the three-valued propositional calculus). Comptes Rendus des Séances de la Societé des Sciences et des Lettres de Varsovie 24, 126–148 (1931); English Translation in [21]

Exploiting Unfounded Sets for HEX-Program Evaluation[⋆]

Thomas Eiter, Michael Fink, Thomas Krennwallner,
Christoph Redl, and Peter Schüller

Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
{eiter,fink,tkren,redl,ps}@kr.tuwien.ac.at

Abstract. HEX programs extend logic programs with external computations through external atoms, whose answer sets are the minimal models of the Faber-Leone-Pfeifer-reduct. As already reasoning from Horn programs with nonmonotonic external atoms of polynomial complexity is on the second level of the polynomial hierarchy, answer set checking needs special attention; simply computing reducts and searching for smaller models does not scale well. We thus extend an approach based on unfounded sets to HEX and integrate it in a Conflict Driven Clause Learning framework for HEX program evaluation. It reduces the check to a search for unfounded sets, which is more efficiently implemented as a SAT problem. We give a basic encoding for HEX and show optimizations by additional clauses. Experiments show that the new approach significantly decreases runtime.

Keywords: Answer Set Programming, Nonmonotonic Reasoning, Unfounded Sets, FLP Semantics.

1 Introduction

Answer Set Programming (ASP) is a declarative programming approach which due to expressive and efficient systems like SMODELS, DLV and CLASP, has been gaining popularity for many applications [2]. Current trends in computing, such as context awareness or distributed systems, raised the need for access to external sources in a program, which, e.g., on the Web ranges from light-weight data access (e.g., XML, RDF, or data bases) to knowledge-intensive formalisms (e.g., description logics).

To cater for this need, HEX-programs [9] extend ASP with so-called external atoms, through which the user can couple any external data source with a logic program. Roughly, such atoms pass information from the program, given by predicate extensions, into an external source which returns output values of an (abstract) function that it computes. This convenient extension has been exploited for many different applications, including querying data and ontologies on the Web, multi-context reasoning,

⋆ This research has been supported by the Austrian Science Fund (FWF) project P20840, P20841, P24090, and by the Vienna Science and Technology Fund (WWTF) project ICT08-020.

or e-government, to mention a few (cf. [5]). It is highly expressive as recursive data exchange between the logic program and external sources is possible.

The semantics of HEX-programs is defined in terms of answer sets based on the FLP reduct [12]: an interpretation $\mathbf{A}$ is an answer set of a program Π, iff it is a $\subseteq$-minimal model of the FLP-reduct $f\Pi^{\mathbf{A}}$ of the program wrt. $\mathbf{A}$, which is the set of all rules whose body is satisfied by $\mathbf{A}$. This semantics is equivalent to the GL-reduct based semantics of ordinary logic programs [14], but has advantages for extensions with nonmonotonic aggregates [12] or the more general external atoms in HEX-programs.

Currently, a HEX-program Π is evaluated in two steps as follows. In step 1, external atoms are viewed as ordinary atoms (*replacement atoms*) and their truth values are guessed by added choice rules. The resulting ordinary ASP program $\hat{\Pi}$ is evaluated by an ordinary ASP solver and each answer set $\hat{\mathbf{A}}$ returned is checked against the external sources, i.e., the guess is verified. After that, the guess for the non-replacement atoms, called $\mathbf{A}$, is known to be a model of Π, but it is yet unknown whether $\mathbf{A}$ is also a subset-minimal model of the reduct $f\Pi^{\mathbf{A}}$. This has to be ensured in step 2 by an *FLP check*. A straightforward method, called *explicit FLP check*, is to compute $f\Pi^{\mathbf{A}}$ and to check whether it has some model smaller than $\mathbf{A}$. However, this approach is not efficient in practice, actually the explicit FLP check often dominates the overall runtime.

This calls for alternative methods to do the FLP check efficiently, which we address in this paper. For ordinary programs, unfounded sets proved to be a fruitful approach [16], which later had been extended to programs with aggregates [11]: an interpretation is an FLP-answer set of some program, if and only if it is unfounded-free, i.e., is disjoint from every unfounded set. Thus to decide whether a candidate is an answer set, one can simply search for unfounded sets, rather than to explicitly construct the reduct and enumerate its models in search for a smaller one.

Starting from this idea, we define unfounded sets for HEX-programs following [11] and explore their efficiency for FLP checking. Briefly, our main contributions are:

- We present an encoding of the unfounded set existence problem to a set of *nogoods*, i.e., constraints that have to be satisfied, and show that the solutions correspond 1-1 with the unfounded sets. They can then be computed using a SAT solver and a post-processing step which checks that the values of replacement atoms comply with the results of the external calls. Furthermore, we consider optimizations which hinge on dependency between external and ordinary atoms, determined in careful analysis. Benchmarks show that this strategy is already more efficient than the explicit FLP check.
- We consider how information gained in the FLP check can be used in generating candidate answer sets in step 1. Recently, adopting a Clause Driven Conflict Learning approach [3], this step has been enhanced by learning [6], in which nogoods describing the external source behavior are learned during the search (called *external behavior learning* or EBL), in order to guide it towards right guesses. We show how step 1 can learn additional nogoods from unfounded sets that avoid the reconstruction of the same or related unfounded sets, yielding further gain.

An experimental evaluation of the above techniques for advanced reasoning applications, including Multi-Context Systems [1,8], abstract argumentation [4] and UNSAT testing [11], shows that unfounded sets checking combined with learning methods from [6] improves HEX-program evaluation considerably. As unfounded-freeness may be ensured by syntactic criteria in relevant cases (which makes the FLP check obsolete), the new approach enables significant speedup and enlarges the scope of HEX applicability. Proofs of our results are given in an extended version [7].

2 Preliminaries

In this section, we start with some basic definitions, and then introduce HEX-programs.

In accordance with [13,6], a *(signed) literal* is a positive or a negative formula $\mathbf{T}a$ resp. $\mathbf{F}a$, where a is a ground atom of form $p(c_1,\dots,c_\ell)$, with predicate p and constants $c_1,\dots,c_\ell$, abbreviated $p(\mathbf{c})$. For a literal $\sigma = \mathbf{T}a$ or $\sigma = \mathbf{F}a$, let $\overline{\sigma}$ denote its opposite, i.e., $\overline{\mathbf{T}a} = \mathbf{F}a$ and $\overline{\mathbf{F}a} = \mathbf{T}a$. An *assignment* is a consistent set of literals $\mathbf{T}a$ or $\mathbf{F}a$, where $\mathbf{T}a$ expresses that $a \in \mathcal{A}$ and $\mathbf{F}a$ that $a \notin \mathcal{A}$. $\mathcal{A}$ is *complete*, also called an *interpretation*, if no assignment $\mathbf{A}' \supset \mathbf{A}$ exists. We denote by $\mathbf{A}^{\mathbf{T}} = \{a \mid \mathbf{T}a \in \mathbf{A}\}$ and $\mathbf{A}^{\mathbf{F}} = \{a \mid \mathbf{F}a \in \mathbf{A}\}$ the set of atoms that are true, resp. false in $\mathbf{A}$, and by $ext(q, \mathbf{A}) = \{\mathbf{c} \mid \mathbf{T}q(\mathbf{c}) \in \mathbf{A}\}$ the extension of a predicate q. Furthermore, $\mathbf{A}|_q$ is the set of all literals over atoms of form $q(\mathbf{c})$ in $\mathbf{A}$. For a list $\mathbf{q} = q_1,\dots,q_k$ of predicates we write $p \in \mathbf{q}$ iff $q_i = p$ for some $1 \le i \le n$, and let $\mathbf{A}|_{\mathbf{q}} = \bigcup_j \mathbf{A}|_{q_j}$.

A *nogood* is a set $\{L_1,\dots,L_n\}$ of literals $L_i, 1 \le i \le n$. An interpretation $\mathbf{A}$ is a *solution* to a nogood δ (resp. a set Δ of nogoods), iff $\delta \not\subseteq \mathbf{A}$ (resp. $\delta \not\subseteq \mathbf{A}$) for all $\delta \in \Delta$.

HEX-Program Syntax. As introduced in [9], HEX-programs are a generalization of (disjunctive) extended logic programs under the answer set semantics [14]; for details and background see [9]. HEX-programs extend ordinary ASP programs by *external atoms*, which enable a bidirectional interaction between a program and external sources of computation. External atoms have a list of input parameters (constants or predicate names) and a list of output parameters. Informally, to evaluate an external atom, the reasoner passes the constants and extensions of the predicates in the input tuple to the external source associated with the external atom. The external source computes output tuples which are with the output list. Formally, a *ground external atom* is of the form

$$\&g[\mathbf{p}](\mathbf{c}), \tag{1}$$

where $\mathbf{p} = p_1,\dots,p_k$ are constant input parameters (predicate names or object constants), and $\mathbf{c} = c_1,\dots,c_l$ are constant output terms.

Ground HEX-programs are then defined similar to ground ordinary ASP programs.

Definition 1 (Ground HEX-programs). *A ground HEX-program consists of rules*

$$a_1 \vee \dots \vee a_k \leftarrow b_1,\dots,b_m, \text{not } b_{m+1},\dots,\text{not } b_n\ , \tag{2}$$

where each a_i is an (ordinary) ground atom $p(c_1,\dots,c_\ell)$ with constants c_i, $1 \le i \le \ell$, each b_j is either an ordinary ground atom or a ground external atom, and $k + n > 0$.[1]

[1] For simplicity, we do not formally introduce strong negation but view, as customary, classical literals $\neg a$ as new atoms together with a nogood $\{\mathbf{T}a, \mathbf{T}\neg a\}$.

The *head* of a rule r is $H(r) = \{a_1, \dots, a_n\}$ and the *body* is $B(r) = \{b_1, \dots, b_m, \text{not } b_{m+1}, \dots, \text{not } b_n\}$. We call b or not b in a rule body a *default literal*; $B^+(r) = \{b_1, \dots, b_m\}$ is the *positive body*, $B^-(r) = \{b_{m+1}, \dots, b_n\}$ is the *negative body*. For a program Π, let $A(\Pi)$ be the set of all ordinary atoms occurring in Π. For a default literal b, let $\mathbf{t}b = \mathbf{T}a$ if $b = a$ for an atom a, and $\mathbf{t}b = \mathbf{F}a$ if $b = \text{not } a$. Conversely, $\mathbf{f}b = \mathbf{F}a$ if $b = a$ and $\mathbf{f}b = \mathbf{T}a$ if $b = \text{not } a$.

We also use non-ground programs. However, as suitable safety conditions allow for using a grounding procedure [10], we limit our investigation to ground programs.

HEX-Program Semantics and Evaluation. The semantics of a ground external atom $\&g[\mathbf{p}](\mathbf{c})$ wrt. an interpretation $\mathbf{A}$ is given by the value of a $1+k+l$-ary Boolean *oracle function*, denoted by $f_{\&g}$ following [9], that is defined for all possible values of $\mathbf{A}$, $\mathbf{p}$ and $\mathbf{c}$. Thus, $\&g[\mathbf{p}](\mathbf{c})$ is true relative to $\mathbf{A}$ if and only if it holds that $f_{\&g}(\mathbf{A}, \mathbf{p}, \mathbf{c}) = 1$. Satisfaction of ordinary rules and ASP programs [14] is then extended to HEX-rules and programs in the obvious way, and the notion of extension $ext(\cdot, \mathbf{A})$ for external predicates $\&g$ with input lists $\mathbf{p}$ is naturally defined by $ext(\&g[\mathbf{p}], \mathbf{A}) = \{\mathbf{c} \mid f_{\&g}(\mathbf{A}, \mathbf{p}, \mathbf{c}) = 1\}$.

An input predicate p of an external predicate with input list $\&g[\mathbf{p}]$ is *monotonic* (*antimonotonic*), iff $f_{\&g}(\mathbf{A}, \mathbf{p}, \mathbf{c}) = 1$ implies $f_{\&g}(\mathbf{A}', \mathbf{p}, \mathbf{c}) = 1$ ($f_{\&g}(\mathbf{A}, \mathbf{p}, \mathbf{c}) = 0$ implies $f_{\&g}(\mathbf{A}', \mathbf{p}, \mathbf{c}) = 0$) for all $\mathbf{A}'$ s.t. $ext(p, \mathbf{A}') \supseteq ext(p, \mathbf{A})$ and $ext(q, \mathbf{A}') = ext(q, \mathbf{A})$ for $q \in \mathbf{p}$ and $q \neq p$. The sublist of all monotonic resp. antimonotonic p is denoted by $m(\&g)$ resp. $a(\&g)$ and the sublist of neither monotonic nor antimonotonic (i.e., *nonmonotonic*) p by $n(\&g)$; we also write $\mathbf{p}_\tau$ for $\tau(\&g)$, $\tau \in \{m, a, n\}$.

Definition 2 (FLP-Reduct [12]). *For an interpretation* $\mathbf{A}$ *over a program* Π*, the* FLP-reduct *of* Π *wrt.* $\mathbf{A}$ *is the set* $f\Pi^{\mathbf{A}} = \{r \in \Pi \mid \mathbf{A} \models b, \textit{ for all } b \in B(r)\}$ *of all rules whose body is satisfied under* $\mathbf{A}$.

An assignment $\mathbf{A}_1$ is smaller or equal to another assignment $\mathbf{A}_2$ wrt. a program Π, denoted $\mathbf{A}_1 \leq_\Pi \mathbf{A}_2$ iff $\{\mathbf{T}a \in \mathbf{A}_1 \mid a \in A(\Pi)\} \subseteq \{\mathbf{T}a \in \mathbf{A}_2 \mid a \in A(\Pi)\}$.

Definition 3 (Answer Set). *An answer set of* Π *is a* $\leq_\Pi$*-minimal model* $\mathbf{A}$ *of* $f\Pi^{\mathbf{A}}$.

Since interpretations (thus answer sets) are complete assignments, slightly abusing notation, we uniquely identify them with the set of all positive literals they contain.

Example 1. Consider the program $\Pi = \{p \leftarrow \&id[p]()\}$, where $\&id[p]()$ is true iff p is true. Then Π has the answer set $\mathbf{A}_1 = \emptyset$; indeed it is a $\leq_\Pi$-minimal model of $f\Pi^{\mathbf{A}_1} = \emptyset$.

The answer sets of a HEX-program Π are determined by the DLVHEX solver using a transformation to ordinary ASP programs as follows. Each external atom $\&g[\mathbf{p}](\mathbf{c})$ in Π is replaced by an ordinary ground *external replacement atom* $e_{\&g[\mathbf{p}]}(\mathbf{c})$ and a rule $e_{\&g[\mathbf{p}]}(\mathbf{c}) \vee ne_{\&g[\mathbf{p}]}(\mathbf{c}) \leftarrow$ is added to the program. The answer sets of the resulting *guessing program* $\hat{\Pi}$ are determined by an ordinary ASP solver and projected to non-replacement atoms. However, the resulting assignments are not necessarily models of Π, as the value of $\&g[\mathbf{p}]$ under $f_{\&g}$ can be different from the one of $e_{\&g[\mathbf{p}]}(\mathbf{c})$. Each answer set of $\hat{\Pi}$ is thus merely a *candidate* which must be checked against the external sources. If no discrepancy is found, the model candidate is a *compatible set* of Π. More precisely,

Definition 4 (Compatible Set). *A* compatible set *of a program Π is an assignment $\hat{\mathbf{A}}$*

(i) which is an answer set [14] of the guessing program *$\hat{\Pi}$, and*

(ii) $f_{\&g}(\hat{\mathbf{A}}, \mathbf{p}, \mathbf{c}) = 1$ iff $\mathbf{T}e_{\&g[\mathbf{p}]}(\mathbf{c}) \in \hat{\mathbf{A}}$ for all external atoms $\&g[\mathbf{p}](\mathbf{c})$ in Π, i.e. the guessed values coincide with the actual output under the input from $\hat{\mathbf{A}}$.

The compatible sets of Π computed by DLVHEX include (modulo $A(\Pi)$) all (FLP) answer sets. For each answer set $\mathbf{A}$ there is a compatible set $\hat{\mathbf{A}}$ such that $\mathbf{A}$ is the restriction of $\hat{\mathbf{A}}$ to non-replacement atoms, but not vice versa. To filter out the compatible sets which are not answer sets, the current evaluation algorithm proceeds as follows. Each compatible set $\mathbf{A}$ is fed to the FLP check, which explicitly constructs $f\Pi^{\mathbf{A}}$. After that, all models of the reduct are enumerated and compared to $\mathbf{A}$. If there is a model which is strictly smaller than $\mathbf{A}$ wrt. Π, then $\mathbf{A}$ is rejected, otherwise $\mathbf{A}$ is an answer set.

Example 2 (cont'd). Reconsider the program $\Pi = \{\, p \leftarrow \&id[p]()\,\}$ from above. Then the corresponding guessing program is $\hat{\Pi} = \{p \leftarrow e_{\&id[p]}(); e_{\&id[p]} \vee ne_{\&id[p]} \leftarrow\}$ and has the answer sets $\mathbf{A}_1 = \emptyset$ and $\mathbf{A}_2 = \{\mathbf{T}p, \mathbf{T}e_{\&id[p]}\}$. While $\mathbf{A}_1$ is also a $\leq_\Pi$-minimal model of $f\Pi^{\mathbf{A}_1} = \emptyset$, A_2 is not a $\leq_\Pi$-minimal model of $f\Pi^{\mathbf{A}_2} = \Pi$.

3 Unfounded Set Detection

It appears that in most current application scenarios there is no smaller model of the reduct $f\Pi^{\mathbf{A}}$, i.e., most assignments $\mathbf{A}$ extracted from compatible sets $\hat{\mathbf{A}}$ pass the FLP check. Moreover, this check is computationally costly: all models of $f\Pi^{\mathbf{A}}$ must be enumerated, along with calls to the external sources to ensure compatibility. Even worse, as we need to search for a smaller *model* and not just for a smaller compatible set, $f\Pi^{\mathbf{A}}$ usually has even more models then the original program. More precisely, the explicit FLP check corresponds to the search for compatible sets of the following program:

$$f\hat{\Pi}^{\hat{\mathbf{A}}} \cup \{\leftarrow a \mid a \text{ is ordinary }, \mathbf{T}a \notin \hat{\mathbf{A}}\} \cup \{a \vee a' \leftarrow \mid \mathbf{T}a \in \hat{\mathbf{A}}\}$$
$$\cup \{\leftarrow \text{not } \mathit{smaller}\} \cup \{\mathit{smaller} \leftarrow \text{not } a \mid a \text{ is ordinary }, \mathbf{T}a \in \hat{\mathbf{A}}\}.$$

It consists of the reduct $f\hat{\Pi}^{\hat{\mathbf{A}}}$ and rules that restrict the search to proper subinterpretations of $\hat{\mathbf{A}}$, where *smaller* is a new atom. Moreover, as we actually need to search for models and not just compatible sets, rules of the form $a \vee a' \leftarrow$ (where a' is a new atom for each $\mathbf{T}a \in \hat{\mathbf{A}}$) make sure that atoms can be arbitrarily true without having a justifying rule in Π. Because of these guessing rules, the rules in the reduct $f\hat{\Pi}^{\hat{\mathbf{A}}}$—except for the guesses on replacement atoms—can be rewritten to constraints, which is more efficient. Our comparison in Section 5 uses this optimized version of the explicit check, but still demonstrates a significant performance gain by our novel approach.

In this section we present a novel FLP check algorithm based on unfounded sets (UFS). That is, instead of explicitly searching for smaller models of the reduct, we check if the candidate answer set is unfounded-free. For now, the unfounded set-based check is also realized as a post-check, i.e., it is carried out only after the interpretation has been completed. Nevertheless it performs much better than the explicit FLP check.

Investigating the effects of doing this check over partial interpretations and interleaving it with the main search for compatible sets is future work. We use unfounded sets for logic programs as introduced in [11] for programs with arbitrary aggregates.

Definition 5 (Unfounded Set). *Given a program Π and an assignment* $\mathbf{A}$*, let X be any set of ordinary ground atoms appearing in Π. Then, X is an* unfounded set for Π wrt. $\mathbf{A}$ *if, for each rule r having some atoms from X in the head, at least one of the following conditions holds, where $\mathbf{A} \dot\cup \neg.X = (\mathbf{A} \setminus \{\mathbf{T}a \mid a \in X\}) \cup \{\mathbf{F}a \mid a \in X\}$:*

(i) some literal of $B(r)$ is false wrt. $\mathbf{A}$*,*
(ii) some literal of $B(r)$ is false wrt. $\mathbf{A} \dot\cup \neg.X$*, or*
(iii) some atom of $H(r) \setminus X$ is true wrt. $\mathbf{A}$

Intuitively, an unfounded set is a set of atoms which only cyclically support each other. Answer sets can be characterized in terms of unfounded sets.

Definition 6 (Unfounded-free Interpretations). *An interpretation* $\mathbf{A}$ *of a program Π is* unfounded-free *iff* $\mathbf{A}^{\mathbf{T}} \cap X = \emptyset$*, for all unfounded sets X of Π wrt.* $\mathbf{A}$.

Theorem 1 (Characterization of Answer Sets). *A model* $\mathbf{A}$ *of a program Π is an answer set iff it is* unfounded-free.

Example 3. Consider the program Π and $\mathbf{A}_2$ from Example 2. Then $X = \{p\}$ is an unfounded set since X intersects with the head of $p \leftarrow \&id[p]()$ and $\mathbf{A} \dot\cup \neg.X \not\models \&id[p]()$. Therefore $\mathbf{A}_2$ is not unfounded-free and not an answer set.

3.1 Nogoods for Unfounded Set Search Encoding

We realize the search for unfounded sets using nogoods, i.e., for a given Π and an assignment $\mathbf{A}$ we construct a set of nogoods, such that solutions to this set correspond to (potential) unfounded sets. We then use a SAT solver to search for such unfounded sets.

Our encoding of unfounded set detection, which is related to [3] but respects external atoms, uses a set $\Gamma_\Pi^{\mathbf{A}} = N_\Pi^{\mathbf{A}} \cup O_\Pi^{\mathbf{A}}$, of nogoods where $N_\Pi^{\mathbf{A}}$ contains all *necessary* constraints and $O_\Pi^{\mathbf{A}}$ are *optional* optimization nogoods that prune irrelevant parts of the search space. The idea is that the set of ordinary atoms of a solution to $\Gamma_\Pi^{\mathbf{A}}$ represents a (potential) unfounded set U of Π wrt. $\mathbf{A}$, while the external replacement atoms encode the truth values of the corresponding external atoms under $\mathbf{A} \dot\cup \neg.U$.

Let $B_o^+(r)$ be the subset of $B^+(r)$ consisting of all *o*rdinary atoms, and $B_e(r)$ the subset of $B(r)$ consisting of all *e*xternal replacement atoms. Then, the nogood set $\Gamma_\Pi^{\mathbf{A}}$ is built over atoms $A(\Gamma_\Pi^{\mathbf{A}}) = A(\hat\Pi) \cup \{h_r, l_r \mid r \in \Pi\}$, where h_r, and l_r are additional atoms for every rule r in Π. The *mandatory part* $N_\Pi^{\mathbf{A}} = \{\{\mathbf{F}a \mid \mathbf{T}a \in \mathbf{A}\}\} \cup \left(\bigcup_{r \in \Pi} R_{r,\mathbf{A}}\right)$ consists of a nogood $\{\mathbf{F}a \mid \mathbf{T}a \in \mathbf{A}\}$, eliminating unfounded sets that do not intersect with true atoms in $\mathbf{A}$, as well as nogoods $R_{r,\mathbf{A}}$ for every $r \in \Pi$. The latter consist of a *head criterion* $H_{r,\mathbf{A}}$ and a *conditional part* $C_{r,\mathbf{A}}$ for each rule, defined by:

- $R_{r,\mathbf{A}} = H_{r,\mathbf{A}} \cup C_{r,\mathbf{A}}$, where
- $H_{r,\mathbf{A}} = \{\{\mathbf{T}h_r\} \cup \{\mathbf{F}h \mid h \in H(r)\}\} \cup \{\{\mathbf{F}h_r, \mathbf{T}h\} \mid h \in H(r)\}$
  encodes that h_r is true for a rule r iff some atom of $H(r)$ is in the unfounded set;

– $C_{r,\mathbf{A}} = \begin{cases} \{\{\mathbf{T}h_r\} \cup \\ \quad \{\mathbf{F}a \mid a \in B_o^+(r), \mathbf{A} \models a\} \cup \{\mathbf{t}a \mid a \in B_e(\hat{r})\} \cup \\ \quad \{\mathbf{T}h \mid h \in H(r), \mathbf{A} \models h\}\} & \textit{if } \mathbf{A} \models B(r), \\ \{\} & \textit{otherwise} \end{cases}$
encodes that condition (i), (ii) or (iii) of Definition 5 must hold if h_r is true.

More specifically, for an unfounded set U and a rule r with $H(r) \cap U \neq \emptyset$ (h_r is true) it must not happen that $\mathbf{A} \models B(r)$ (condition (i) fails), no $a \in B_o^+(r)$ with $\mathbf{A} \models a$ is in the unfounded set and all $a \in B_e(\hat{r})$ are true under $\mathbf{A} \dot{\cup} \neg.U$ (condition (ii) fails), and all $h \in H(r)$ with $\mathbf{A} \models h$ are in the unfounded set (condition (iii) fails).

Towards computing unfounded sets, observe that they can be extended to solutions to the set of nogoods $\Gamma_\Pi^\mathbf{A}$ over $A(\Gamma_\Pi^\mathbf{A})$. Conversely, the solutions to $\Gamma_\Pi^\mathbf{A}$ include specific extensions of all unfounded sets, characterized by *induced assignments*: That is, by assigning true to all atoms in U, to all h_r such that $H(r)$ intersects with U, and to all external replacement atoms $e_{\&g[\mathbf{p}]}(\mathbf{c})$ such that $\&g[\mathbf{p}](\mathbf{c})$ is true under $\mathbf{A} \dot{\cup} \neg.U$, and assigning false to all other atoms in $A(\Gamma_\Pi^\mathbf{A})$. More formally, we define:

Definition 7 (Induced Assignment of an Unfounded Set). *Let U be an unfounded set of a program Π wrt. assignment $\mathbf{A}$. The* assignment induced by U, *denoted $I(U, \Gamma_\Pi^\mathbf{A})$, is*

$$\begin{aligned} I(U, \Gamma_\Pi^\mathbf{A}) &= I'(U, \Gamma_\Pi^\mathbf{A}) \cup \{\mathbf{F}a \mid a \in A(\Gamma_\Pi^\mathbf{A}), \mathbf{T}a \notin I'(U, \Gamma_\Pi^\mathbf{A})\}, \textit{ where} \\ I'(U, \Gamma_\Pi^\mathbf{A}) &= \{\mathbf{T}a \mid a \in U\} \cup \{\mathbf{T}h_r \mid r \in \Pi, H(r) \cap U \neq \emptyset\} \cup \\ &\quad \{\mathbf{T}e_{\&g[\mathbf{p}]}(\mathbf{c}) \mid e_{\&g[\mathbf{p}]}(\mathbf{c}) \in A(\hat{\Pi}), \mathbf{A} \dot{\cup} \neg.U \models \&g[\mathbf{p}](\mathbf{c})\}. \end{aligned}$$

While concrete instances for $O_\Pi^\mathbf{A}$ are defined in Section 4, note that for the next result we simply require that the optimization part $O_\Pi^\mathbf{A}$ is conservative in the sense that for every unfounded set U of Π wrt. $\mathbf{A}$, it holds that $I(U, \Gamma_\Pi^\mathbf{A})$ is a solution to $O_\Pi^\mathbf{A}$ as well (which is shown for the different optimizations considered in the next section). Then, the solutions to $\Gamma_\Pi^\mathbf{A}$ include all assignments induced by unfounded sets of Π wrt. $\mathbf{A}$ (but not every solution corresponds to such an induced assignment, intuitively because it does not reflect the semantics of external sources).

Proposition 1. *Let U be an unfounded set of a program Π wrt. assignment $\mathbf{A}$ such that $\mathbf{A}^\mathbf{T} \cap U \neq \emptyset$. Then $I(U, \Gamma_\Pi^\mathbf{A})$ is a solution to $\Gamma_\Pi^\mathbf{A}$.*

Proof (Sketch). We make a proof by contraposition and show that if $I(U, \Gamma_\Pi^\mathbf{A})$ is not a solution to $\Gamma_\Pi^\mathbf{A}$, then U cannot be an unfounded set.

First observe that the nogoods in $H_{r,\mathbf{A}}$ demand $\mathbf{T}h_r$ to be true for a rule $r \in \Pi$ if and only if some head atom $h \in H(r)$ of this rule is in U. As the conditions in these nogoods are mutually exclusive and therefore consistent, and the truth value of h_r in $I(U, \Gamma_\Pi^\mathbf{A})$ is defined exactly to this criterion, $C_{r,\mathbf{A}}$ must be involved in a contradiction. Moreover, the nogood $\{\mathbf{F}a \mid \mathbf{T}a \in \mathbf{A}\} \in N^\mathbf{A}$ eliminates $I(U, \Gamma_\Pi^\mathbf{A})$ only if U does not intersect with the positive atoms in $\mathbf{A}$. This is no problem because we are only interested in such unfounded sets.

Therefore, if $I(U, \Gamma_\Pi^\mathbf{A})$ is not a solution to $\Gamma_\Pi^\mathbf{A}$, then for some rule $r \in \Pi$ the nogood in $C_{r,\mathbf{A}}$ must be violated. That is, we know the following: $\mathbf{T}h_r \in I(U, \Gamma_\Pi^\mathbf{A})$ (and therefore $H(r) \cap U \neq \emptyset$), $\mathbf{F}a \in I(U, \Gamma_\Pi^\mathbf{A})$ for all $a \in B_o^+(r)$, $\mathbf{t}a \in I(U, \Gamma_\Pi^\mathbf{A})$ for all

$a \in B_e(\hat{r})$, and $\mathbf{T}h \in I(U, \Gamma_\Pi^\mathbf{A})$ for all $h \in H(r)$ with $\mathbf{A} \models h$. Moreover, we have $C_{r,\mathbf{A}} \neq \emptyset$. We now show that this implies that none of the conditions of Definition 5 holds for r wrt. U and $\mathbf{A}$, which contradicts the assumption that U is an unfounded set.

Condition (i) does not hold for r because $\mathbf{A} \models B(r)$ (otherwise $C_{r,\mathbf{A}} = \emptyset$).

Condition (ii) does not hold for r. Suppose to the contrary that it holds. Then there must be some $b \in B(r)$ s.t. $\mathbf{A} \dot\cup \neg.U \not\models b$. Because $C_{r,\mathbf{A}} \neq \emptyset$, we know that $\mathbf{A} \models b$. We make a case distinction on the type of b:

– If b is a positive ordinary atom, then $\mathbf{F}b \in I(U, \Gamma_\Pi^\mathbf{A})$ and therefore $b \notin U$. Consequently $\mathbf{A} \dot\cup \neg.U \models b$. Contradiction.
– If b is a negative ordinary atom, then $\mathbf{A} \models b$ implies $\mathbf{A} \dot\cup \neg.U \models b$. Contradiction.
– If b is a positive or default-negated replacement atom of an external , then $\mathbf{t}b \in I(U, \Gamma_\Pi^\mathbf{A})$. But this implies, by definition of $I(U, \Gamma_\Pi^\mathbf{A})$, that $\mathbf{A} \dot\cup \neg.U \models b$. Contradiction.

Condition (iii) does not hold for r because $\mathbf{T}h \in I(U, \Gamma_\Pi^\mathbf{A})$ and thus, by definition of $I(U, \Gamma_\Pi^\mathbf{A})$, $h \in U$ for all $h \in H(r)$ with $\mathbf{A} \models h$. Thus $\mathbf{A} \not\models a$ for all $a \in H(r) \setminus U$. □

Corollary 1. *If $\Gamma_\Pi^\mathbf{A}$ has no solution, then $U \cap \mathbf{A}^\mathbf{T} = \emptyset$ for every unfounded set U of Π.*

The next property allows us to find the unfounded sets of Π wrt. $\mathbf{A}$ among all solutions to $\Gamma_\Pi^\mathbf{A}$ by using a postcheck on the external atoms.

Proposition 2. *Let S be a solution to $\Gamma_\Pi^\mathbf{A}$ such that*

(a) $\mathbf{T}e_{\&g[\mathbf{p}]}(\mathbf{c}) \in S$ *and* $\mathbf{A} \not\models \&g[\mathbf{p}](\mathbf{c})$ *implies* $\mathbf{A} \dot\cup \neg.U \models \&g[\mathbf{p}](\mathbf{c})$*; and*
(b) $\mathbf{F}e_{\&g[\mathbf{p}]}(\mathbf{c}) \in S$ *and* $\mathbf{A} \models \&g[\mathbf{p}](\mathbf{c})$ *implies* $\mathbf{A} \dot\cup \neg.U \not\models \&g[\mathbf{p}](\mathbf{c})$

where $U = \{a \mid a \in A(\Pi), \mathbf{T}a \in S\}$. Then U is an unfounded set of Π wrt. $\mathbf{A}$.

Informally, the proposition states that true non-replacement atoms in S which also appear in Π form an unfounded set, provided that truth of the external replacement atoms $e_{\&g[\mathbf{p}]}(\mathbf{c})$ in S coincides with the truth of the corresponding $\&g[\mathbf{p}](\mathbf{c})$ under $\mathbf{A} \dot\cup \neg.U$ (as in Definition 7). However, this check is just required if the truth value of $e_{\&g[\mathbf{p}]}(\mathbf{c})$ in S and of $\&g[\mathbf{p}](\mathbf{c})$ under $\mathbf{A}$ differ. This gives rise to an important optimization for the implementation: external atoms, whose (known) truth value of $\&g[\mathbf{p}](\mathbf{c})$ under $\mathbf{A}$ matches the truth value of $e_{\&g[\mathbf{p}]}(\mathbf{c})$ in S, do not need to be postchecked.

Proof (Sketch). Suppose U is not an unfounded set. Then there is a $r \in \Pi$ s.t. $H(r) \cap U \neq \emptyset$ and none of the conditions in Definition 5 is satisfied. We show now that S cannot be a solution to $\Gamma_\Pi^\mathbf{A}$.

Because condition (i) does not hold, there is a nogood of form

$$\{\{\mathbf{T}h_r\} \cup \{\mathbf{F}a \mid a \in B_o^+(r), \mathbf{A} \models a\} \cup \{\mathbf{t}a \mid a \in B_e(\hat{r})\} \cup \{\mathbf{T}h \mid h \in H(r), \mathbf{A} \models h\}\}$$

in $\Gamma_\Pi^\mathbf{A}$.

We now show that S contains all signed literals of this nogood, i.e., the nogood is violated under S.

Because of $H(r) \cap U \neq \emptyset$, $\mathbf{T}h_r \in S$ (otherwise a nogood in $H_r^\mathbf{A}$ is violated).

As U is not an unfounded set, condition (ii) in Definition 5 does not hold. Consider all $a \in B_o^+(r)$ s.t. $\mathbf{A} \models a$. Then $a \notin U$, otherwise $A \mathbin{\dot{\cup}} \neg.U \not\models a$ and we have a contradiction with the assumption that condition (ii) is unsatisfied. But then $\mathbf{F}a \in S$.

Now consider all $\&g[\mathbf{p}](\mathbf{c}) \in B_e(r)$. Then $\mathbf{A} \mathbin{\dot{\cup}} \neg.U \models \&g[\mathbf{p}](\mathbf{c})$ (as (ii) is violated). If $\mathbf{A} \not\models \&g[\mathbf{p}](\mathbf{c})$, then condition (i) would be satisfied, hence $\mathbf{A} \models \&g[\mathbf{p}](\mathbf{c})$. But then $\mathbf{T}e_{\&g[\mathbf{p}]}(\mathbf{c}) \in S$, otherwise $\mathbf{A} \mathbin{\dot{\cup}} \neg.U \not\models \&g[\mathbf{p}](\mathbf{c})$ by precondition (b) of this proposition. Next consider all $\mathrm{not}\, \&g[\mathbf{p}](\mathbf{c}) \in B_e(r)$. Then $\mathbf{A} \mathbin{\dot{\cup}} \neg.U \not\models \&g[\mathbf{p}](\mathbf{c})$ (as (ii) is violated). If $\mathbf{A} \models \&g[\mathbf{p}](\mathbf{c})$, then condition (i) would be satisfied, hence $\mathbf{A} \not\models \&g[\mathbf{p}](\mathbf{c})$. But then $\mathbf{F}e_{\&g[\mathbf{p}]}(\mathbf{c}) \in S$, otherwise $\mathbf{A} \mathbin{\dot{\cup}} \neg.U \models \&g[\mathbf{p}](\mathbf{c})$ by precondition (a) of this proposition. Therefore, we have $\mathbf{t}a \in S$ for all $a \in B_e(\hat{r})$.

Finally, because condition (iii) in Definition 5 does not hold, $h \in U$ and therefore also $\mathbf{T}h \in S$ for all $h \in H(r)$ with $\mathbf{A} \models a$.

This concludes the proof that S cannot be a solution to $\Gamma_{\Pi}^{\mathbf{A}}$ satisfying (a) and (b), if U is not an unfounded set. □

Example 4. Reconsider program $\Pi = \{r_1 : p \leftarrow \&id[p]()\}$ from Ex. 2 and the compatible set $\mathbf{A}_2 = \{\mathbf{T}p, \mathbf{T}e_{\&id[p]}\}$. The nogood set $N_{\Pi}^{\mathbf{A}_2} = \{\{\mathbf{T}h_{r_1}, \mathbf{F}p\}, \{\mathbf{F}h_{r_1}, \mathbf{T}p\}, \{\mathbf{T}h_{r_1}, \mathbf{T}e_{\&id[p]}(), \mathbf{T}p\}\}$ has solutions $S \supseteq \{\mathbf{T}h_{r_1}, \mathbf{T}p, \mathbf{F}e_{\&id[p]}()\}$, which correspond to the unfounded set $U = \{p\}$. Here, $\mathbf{F}e_{\&id[p]}()$ represents that $\mathbf{A}_2 \dot{\cup} \neg.U \not\models \&id[p]()$.

Note that due to the premises in Conditions (a) and (b) of Proposition 2, the postcheck is faster if $\mathbf{T}e_{\&g[\mathbf{p}]}(\mathbf{c}) \in S$ whenever $\mathbf{A} \models \&g[\mathbf{p}](\mathbf{c})$ holds for many external atoms in Π. This can be exploited during the construction of S as follows: If it is not absolutely necessary to set the truth value of $e_{\&g[\mathbf{p}]}(\mathbf{c})$ differently, then carry over the value from $\&g[\mathbf{p}](\mathbf{c})$ under $\mathbf{A}$. Specifically, this is successful if $e_{\&g[\mathbf{p}]}(\mathbf{c})$ does not occur in $\Gamma_{\Pi}^{\mathbf{A}}$.

4 Optimization and Learning

In this section we first discuss some refinements and optimizations of our encoding of nogoods for UFS search. In particular, we add additional nogoods which prune irrelevant parts of the search space. After that, we propose a strategy for learning nogoods from detected unfounded sets, avoiding that the same unfounded set is generated again later.

4.1 Optimization

The following optimizations turned out to be effective in improving UFS search.

Restricting the UFS Search to Atoms in the Compatible Set. First, not all atoms in a program are relevant for the unfounded set search: atoms that are false under $\mathbf{A}$ can be ignored. Formally one can show the following:

Proposition 3. *If U is an unfounded set of Π wrt. $\mathbf{A}$ and there is an $a \in U$ s.t. $\mathbf{A} \not\models a$, then $U \setminus \{a\}$ is an unfounded set of Π wrt. $\mathbf{A}$.*

Proof (Sketch). Let $r \in \Pi$ s.t. $H(r) \cap (U \setminus \{a\}) \neq \emptyset$. We have to show that one of the conditions of Definition 5 holds wrt. $\mathbf{A}$ and $U \setminus \{a\}$.

Because U is an unfounded set of Π wrt. $\mathbf{A}$ and $H(r) \cap (U \setminus \{a\}) \neq \emptyset$ implies $H(r) \cap U \neq \emptyset$, one of the conditions of Definition 5 holds wrt. $\mathbf{A}$ and U. If this is condition (i) or (iii), it also holds wrt. $U \setminus \{a\}$ because these condition depend only on r and $\mathbf{A}$. Also if condition (ii) holds, it also holds wrt. $U \setminus \{a\}$ because $\mathbf{A} \dot{\cup} \neg.U$ is equivalent to $\mathbf{A} \dot{\cup} \neg.(U \setminus \{a\})$ since $a \notin U$. □.

Avoiding Guesses of External Replacement Atoms. Second, in some situations the truth value of an external replacement atom b in a solution S to $\Gamma_{\Pi}^{\mathbf{A}}$ is void. That is, both $(S \setminus \{\mathbf{T}b, \mathbf{F}b\}) \cup \{\mathbf{T}b\}$ and $(S \setminus \{\mathbf{T}b, \mathbf{F}b\}) \cup \{\mathbf{F}b\}$ are solutions to $\Gamma_{\Pi}^{\mathbf{A}}$ (which represent the same unfounded set). Then we can set the truth value to an (arbitrary) fixed value instead of inspecting both alternatives. The following provides a sufficient criterion:

Proposition 4. *Let b be an external atom replacement, and let S be a solution to $\Gamma_{\Pi}^{\mathbf{A}}$. If for all rules $r \in \Pi$, such that $\mathbf{A} \models B(r)$ and where $b \in B^{+}(\hat{r})$ or $b \in B^{-}(\hat{r})$, either*

(a) for some $a \in B_{o}^{+}(r)$ such that $\mathbf{A} \models a$, it holds that $\mathbf{T}a \in S$; or
(b) for some $a \in H(r)$ such that $\mathbf{A} \models a$, it holds that $\mathbf{F}a \in S$;

then both $(S \setminus \{\mathbf{T}b, \mathbf{F}b\}) \cup \{\mathbf{T}b\}$ and $(S \setminus \{\mathbf{T}b, \mathbf{F}b\}) \cup \{\mathbf{F}b\}$ are solutions to $\Gamma_{\Pi}^{\mathbf{A}}$.

Proof (Sketch). Suppose that changing the truth value of b in S turns the solution to a counterexample of $\Gamma_{\Pi}^{\mathbf{A}}$. Then there must be a violated nogood $N \in \Gamma_{\Pi}^{\mathbf{A}}$ containing b, i.e., $\mathbf{T}b \in N$ or $\mathbf{F}b \in N$. But this nogood corresponds to a rule with $b \in B^{+}(\hat{r})$ or $b \in B^{-}(\hat{r})$ and $\mathbf{A} \models B(r)$, and it contains also the signed literals (1) $\mathbf{F}a$ for all $a \in B_{o}^{+}$ with $\mathbf{A} \models a$ and (2) $\mathbf{T}a$ for all $a \in H(r)$ with $\mathbf{A} \models a$.

By precondition of the proposition we have either (a) $\mathbf{T}a \in S$ for some $a \in B_{o}^{+}(r)$ with $\mathbf{A} \models a$, or (b) $\mathbf{F}a$ for some $a \in H(r)$ with $\mathbf{A} \models a$. But then the nogood cannot be violated, because (a) contradicts one of (1) and (b) contradicts one of (2). □

This property can be utilized by adding the following additional nogoods. Recall that $A(\Gamma_{\Pi}^{\mathbf{A}})$ contains atoms l_r for every $r \in \Pi$. They are intuitively used to encode for a solution S to $\Gamma_{\Pi}^{\mathbf{A}}$, whether the truth values of the external atom replacements in $B(r)$ are relevant, or whether they can be set arbitrarily for r. The following nogoods label relevant rules r, forcing l_r to be false iff one of the preconditions in Proposition 4 holds:

$$\begin{aligned} L_r^{\mathbf{A}} = & \{\{\mathbf{T}l_r, \mathbf{T}a\} \mid a \in B_o^{+}(r), \mathbf{A} \models a\} \cup \{\{\mathbf{T}l_r, \mathbf{F}a\} \mid a \in H(r), \mathbf{A} \models a\} \cup \\ & \{\{\mathbf{F}l_r\} \cup \{\mathbf{F}a \mid a \in B_o^{+}(r), \mathbf{A} \models a\} \cup \{\mathbf{T}a \mid a \in H(r), \mathbf{A} \models a\}\}. \end{aligned}$$

These constraints exclusively enforce $\mathbf{T}l_r$ or $\mathbf{F}l_r$. Hence, the truth value of l_r deterministically depends on the other atoms, i.e., the nogoods do not cause additional guessing.

By Prop. 4 we can set the truth value of an external replacement atom b arbitrarily, if l_r is false for all r such that $b \in B^{+}(\hat{r})$ or $b \in B^{-}(\hat{r})$. As mentioned after Prop. 2, it is advantageous to set the truth value of $e_{\&g[\mathbf{p}]}(\mathbf{c})$ to the one of $\&g[\mathbf{p}](\mathbf{c})$ under $\mathbf{A}$, because this can reduce the number of external atoms that must be checked. The following nogoods enforce a coherent interpretation of the external replacement atoms:

$$\begin{aligned} F_r^{\mathbf{A}} = & \{\{\mathbf{F}l_r \mid b \in B^{+}(\hat{r}) \cup B^{-}(\hat{r})\} \cup \{\mathbf{F}b\} \mid b \in B_e(\hat{r}), \mathbf{A} \models b\} \cup \\ & \{\{\mathbf{F}l_r \mid b \in B^{+}(\hat{r}) \cup B^{-}(\hat{r})\} \cup \{\mathbf{T}b\} \mid b \in B_e(\hat{r}), \mathbf{A} \not\models b\} \end{aligned}$$

In summary, our optimization part therefore is given by $O_{\Pi}^{\mathbf{A}} = \bigcup_{r \in \Pi} L_r^{\mathbf{A}} \cup F_r^{\mathbf{A}}$.

Example 5. Consider the program $\Pi = \{r_1 : p \leftarrow \&id[p]().,r_2 : q \leftarrow \&id[q]().\}$, and the compatible set $\mathbf{A} = \{\mathbf{T}p, \mathbf{F}q, \mathbf{T}e_{\&id[p]}(), \mathbf{F}e_{\&id[q]}()\}$. Then, $N_{\Pi}^{\mathbf{A}}$ has solutions $S_1 \supseteq \{\mathbf{T}h_{r_1}, \mathbf{T}p, \mathbf{F}e_{\&id[p]}(), \mathbf{F}h_{r_2}, \mathbf{F}q, \mathbf{F}e_{\&id[q]}()\}$ and $S_2 \supseteq \{\mathbf{T}h_{r_1}, \mathbf{T}p, \mathbf{F}e_{\&id[p]}(), \mathbf{F}h_{r_2}, \mathbf{F}q, \mathbf{T}e_{\&id[q]}()\}$ (which represent the same unfounded set $U = \{p\}$). Here, the optimization part for r_2, $L_{r_2}^{\mathbf{A}} \cup F_{r_2}^{\mathbf{A}} = \{\{\mathbf{T}l_{r_2}, \mathbf{F}q\}, \{\mathbf{F}l_{r_2}, \mathbf{T}q\}, \{\mathbf{F}l_{r_2}, \mathbf{T}e_{\&id[q]}()\}\}$, eliminates solutions S_2 for $\Gamma_{\Pi}^{\mathbf{A}}$. This is beneficial as for solutions S_1 the postcheck is easier ($e_{\&id[q]}()$ in S_1 and $\&id[q]()$ have the same truth value under $\mathbf{A}$).

Exchanging Nogoods between UFS and Main Search. The third optimization allows for the exchange of learned knowledge about external atoms between the UFS check and the main search for compatible sets. For this purpose, we first define nogoods which correctly describe the input-output relationship of external atoms.

Definition 8. *A nogood of the form* $N = \{\mathbf{T}t_1, \ldots, \mathbf{T}t_n, \mathbf{F}f_1, \ldots, \mathbf{F}f_m, \circ e_{\&g[\mathbf{p}]}(\mathbf{c})\}$, *where* $\circ$ *is* $\mathbf{T}$ *or* $\mathbf{F}$*, is a* valid input-output-relationship*, iff for all assignments* $\mathbf{A}$*,* $\mathbf{T}t_i \in \mathbf{A}$*, for* $1 \leq i \leq n$*, and* $\mathbf{F}f_i \in \mathbf{A}$*, for* $1 \leq i \leq m$*, implies* $\mathbf{A} \models \&g[\mathbf{p}](\mathbf{c})$ *if* $\circ = \mathbf{F}$*, and* $\mathbf{A} \not\models \&g[\mathbf{p}](\mathbf{c})$ *if* $\circ = \mathbf{T}$.

Let N be a nogood which is a valid input-output-relationship learned during the *main search*, i.e., for compatible sets of $\hat{\Pi}$, and let $\bar{\circ} = \mathbf{T}$ if $\circ = \mathbf{F}$, resp. $\bar{\circ} = \mathbf{F}$ if $\circ = \mathbf{T}$.

Definition 9 (Nogood Transformation $\mathcal{T}$). *For a valid input-output relationship* N *and an assignment* $\mathbf{A}$*, the nogood transformation* $\mathcal{T}$ *is defined as*

$$\mathcal{T}(N, \mathbf{A}) = \begin{cases} \emptyset & \textit{if } \mathbf{F}t_i \in \mathbf{A} \textit{ for some } 1 \leq i \leq n, \\ \{\{\mathbf{F}t_1, \ldots, \mathbf{F}t_n\} \cup \{\circ e_{\&g[\mathbf{p}]}(\mathbf{c})\}\} \cup & \\ \quad \{\mathbf{T}f_i \mid 1 \leq i \leq m, \mathbf{A} \models f_i\} & \textit{otherwise}. \end{cases}$$

The next result states that $\mathcal{T}(N, \mathbf{A})$ can be considered, for all valid input-output relationships N under all assignments $\mathbf{A}$, without losing unfounded sets.

Proposition 5. *Let* N *be a valid input-output relationship, and let* U *be an unfounded set wrt.* Π *and* $\mathbf{A}$*. Then* $I(U, \Gamma_{\Pi}^{\mathbf{A}})$ *is a solution to* $\mathcal{T}(N, \mathbf{A})$.

Proof (Sketch). If $\mathcal{T}(N, \mathbf{A}) = \emptyset$ then the proposition trivially holds. Otherwise $\mathcal{T}(N, \mathbf{A}) = \{C\}$ we know that $\mathbf{T}t_i \in \mathbf{A}$ for all $1 \leq i \leq n$. Suppose C is violated. Then $\mathbf{F}t_i \in I(U, \Gamma_{\Pi}^{\mathbf{A}})$ and therefore $t_i \notin U$ for all $1 \leq i \leq n$, and $\mathbf{T}f_i \in I(U, \Gamma_{\Pi}^{\mathbf{A}})$ for all $1 \leq i \leq m$ with $\mathbf{A} \models f_i$, and $\circ e_{\&g[\mathbf{p}]}(\mathbf{c}) \in I(U, \Gamma_{\Pi}^{\mathbf{A}})$.

But then $\mathbf{A} \dot{\cup} \neg.U \models t_i$ for all $1 \leq i \leq n$ and $\mathbf{A} \dot{\cup} \neg.U \not\models f_i$ for all $1 \leq i \leq m$. Because nogood N is a valid input-output-relationship, this implies $\bar{\circ}\&g[\mathbf{p}](\mathbf{c}) \in \mathbf{A} \dot{\cup} \neg.U$. Then by definition of $I(U, \Gamma_{\Pi}^{\mathbf{A}})$ we have $\bar{\circ}e_{\&g[\mathbf{p}]}(\mathbf{c}) \in I(U, \Gamma_{\Pi}^{\mathbf{A}})$, which contradicts the assumption that $\mathcal{T}(N, \mathbf{A})$ is violated. □

Hence, all valid input-output relationships for external atoms which are learned during the search for compatible sets, can be reused (applying the above transformation) for the unfounded set check. Moreover, during the evaluation of external atoms in the postcheck for candidate unfounded sets (solutions to $\Gamma_{\Pi}^{\mathbf{A}}$), further valid input-output relationships might be learned. These can in turn be used by further unfounded set checks.

Example 6 (Set Partitioning). Consider the program Π

$$\begin{aligned} sel(a) &\leftarrow domain(a), \&diff[domain, nsel](a) \\ nsel(a) &\leftarrow domain(a), \&diff[domain, sel](a) \\ domain(a) &\leftarrow \end{aligned}$$

where $\&diff[p, q](X)$ computes the set of all elements X which are in the extension of p but not in the extension of q. Informally, this program implements a choice from $sel(a)$ and $nsel(a)$. Consider the compatible set $\mathbf{A}^{\mathbf{T}} = \{domain(a), sel(a), e_{\&diff[nsel]}(a)\}$. Suppose the main search has learned the input-output relationship $N = \{\mathbf{T}domain(a),\ \mathbf{F}nsel(a), \mathbf{F}e_{\&diff[nsel]}(a)\}$. Then the transformed nogood is $\mathcal{T}(N,\mathbf{A})=\{\{\mathbf{F}domain(a),\ \mathbf{F}e_{\&diff[nsel]}(a)\}\}$, which intuitively encodes that, if $domain(a)$ is *not* in the unfounded set U, then $e_{\&diff[nsel]}(a)$ is true under $\mathbf{A} \dot{\cup} \neg.U$. This is clear because $e_{\&diff[nsel]}(a)$ is true under $\mathbf{A}$ and it can only change its truth value if $domain(a)$ becomes false.

However, it is important that this optimization *cannot* be simultaneously used with the previous one as this can result in contradictions due to (transformed) learned nogoods. Consequently, the previous optimization has been disabled in running our experiments.

4.2 Learning Nogoods from Unfounded Sets

Until now only detecting unfounded sets has been considered. A strategy to *learn* from detected unfounded sets for the main search for compatible sets is missing. Here we develop such a strategy and call it *unfounded set learning* (UFL).

Example 7. Consider the program $\Pi = \{\, p \leftarrow \&id[p]();\ x_1 \vee x_2 \vee \cdots \vee x_k \leftarrow \}$. As we know from Example 3, $\{p\}$ is a UFS wrt. $\mathbf{A} = \{\mathbf{T}p, \mathbf{T}e_{\&id}()\}$, regarding just the first rule. However, the same is true for any $\mathbf{A}' \supset \mathbf{A}$ regarding Π, i.e., p must never be true.

The program in Example 7 has many compatible sets, and half of them (all where p is true) will fail the UFS check for the same reason. We thus develop a strategy for generating additional nogoods to guide the further search for compatible sets in a way, such that the same unfounded sets are not reconsidered.

For an unfounded set U of Π wrt. $\mathbf{A}$ we define the following set of learned nogoods:

$$L(U, \Pi, \mathbf{A}) = \{\{\sigma_0, \sigma_1, \ldots, \sigma_j\} \mid \sigma_0 \in \{\mathbf{T}a \mid a \in U\}, \sigma_i \in H_i \textit{ for all } 1 \leq i \leq j)\},$$

where $H_i = \{\mathbf{T}h \in H(r_i) \mid h \notin U, \mathbf{A} \models h\} \cup \{\mathbf{F}b \in B_o^+(r_i) \mid \mathbf{A} \not\models b\}$ and $\{r_1, \ldots, r_j\} = \{r \in \Pi \mid H(r) \cap U \neq \emptyset, U \cap B_o^+(r) = \emptyset\}$ is the set of external rules of Π wrt. U, i.e., all rules which do not depend on U.

Formally we can show that adding this set of nogoods is correct:

Proposition 6. *If U is an unfounded set of Π wrt. $\mathbf{A}$, then every answer set of Π is a solution to the nogoods in $L(U, \Pi, \mathbf{A})$.*

Proof (Sketch). Suppose there is an answer set $\mathbf{A}'$ of Π which is not a solution to a nogood in $L_3(U, \Pi, \mathbf{A})$ We show that then U is an unfounded set of Π wrt. $\mathbf{A}'$ which intersects with $\mathbf{A}'$.

Let $\{\sigma_0, \sigma_1, \ldots, \sigma_n\}$ be a violated nogood. Let $r \in \Pi$ be a rule such that $H(r) \cap U \neq \emptyset$. We have to show that one of the conditions of Definition 5 holds.

If $B_o^+(r) \cap U \neq \emptyset$, then condition (ii) holds, therefore we can assume $B_o^+(r) \cap U = \emptyset$. Then r is an external rule of Π wrt. U. But then there is a σ_i with $1 \leq i \leq n$ such that either (1) $\sigma_i = \mathbf{T}h$ for some $h \in H(r)$ with $h \notin U$ and $\mathbf{A} \models h$, or (2) $\sigma_i = \mathbf{F}b$ for some $b \in B_o^+(r)$ with $\mathbf{A} \not\models b$. Because the nogood is violated by $\mathbf{A}'$ by assumption, we have $\sigma_i \in \mathbf{A}'$. If (1) then condition (iii) is satisfied, if (2) then condition (i) is satisfied.

Moreover, by definition of L_3 there is an $a \in U$ s.t. $\mathbf{T}a \in \mathbf{A}'$, i.e., $\mathbf{A}'$ intersects with U. □

Example 8. Consider the program Π from Example 7 and suppose we have found the unfounded set $U = \{p\}$ wrt. interpretation $\mathbf{A} = \{\mathbf{T}p, \mathbf{T}x_1\} \cup \{\mathbf{F}a_i \mid 1 < i \leq k\}$. Then the learned nogood $L_2(U, \mathbf{A}, \Pi) = \{\mathbf{T}p\}$ immediately guides the search to the part of the search tree where p is false, i.e., roughly half of the guesses are avoided.

We also considered a different learning strategy based on the models of $f\Pi^{\mathbf{A}}$ rather than the unfounded set U itself, hinging on the observation (cf. [12]) that for every unfounded set U, $\mathbf{A} \dot{\cup} \neg.U$ is a model of $f\Pi^{\mathbf{A}}$ (hence $U \neq \emptyset$ refutes $\mathbf{A}$ as a minimal model of $f\Pi^{\mathbf{A}}$). However, this strategy appeared to be inferior to the one above.

5 Implementation and Evaluation

For implementing our technique, we integrated CLASP into our prototype system DLVHEX; we use CLASP as an ASP solver for computing compatible sets and as a SAT solver for solving the nogood set of the UFS check. We evaluated the implementation on a Linux server with two 12-core AMD 6176 SE CPUs with 128GB RAM.

Table 1 summarizes our benchmark results (plain stands for disabling EBL and UFL). We can see a clear improvement both for synthetic and for application instances,[2] due to the UFS check and EBL. Moreover, a closer analysis shows that the UFS check in some cases not only decreases the runtime but also the numbers of enumerated candidates (UFS candidates resp. model candidates of the FLP reduct) and of external atom evaluations.

Set Partitioning. This benchmark extends the program from Ex. 6 by the additional constraint $\leftarrow sel(X), sel(Y), sel(Z), X \neq Y, X \neq Z, Y \neq Z$ and varies the size of $domain$. Here we see a big advantage of the UFS check over the explicit check, both for computing all answer sets and for finding the first one. A closer investigation shows that the improvement is mainly due to the optimizations described in Sec. 4 which make the UFS check investigate significantly fewer candidates than the explicit FLP check. Furthermore the UFS check requires fewer external computations.

Multi-Context Systems (MCSs). MCSs [1] are a formalism for interlinking knowledge based systems; in [8], *inconsistency explanations (IEs)* for an MCS were defined.

[2] Detailed instance information: `http://www.kr.tuwien.ac.at/staff/ps/unfoundedsets/`

This benchmark computes the IEs, which correspond 1-1 to answer sets of an encoding rich in cycles through external atoms (which evaluate local knowledge base semantics). We use random instances of different topologies created with an available benchmark generator.

For most instances, we observed that the number of candidates for smaller models of the FLP reduct equals the one of unfounded set candidates. This is intuitive as each unfounded set corresponds to a smaller model; the optimization techniques do not prune the search space in this case. However, as we stop the enumeration as soon as a smaller model resp. an unfounded set is found, depending on the specific program and solver heuristics, the explicit and the UFS check may consider different numbers of interpretations. This explains why the UFS check is sometimes slightly slower than the explicit check. However, it always has a smaller delay between different UFS candidates, which sometimes makes it faster even if it visits more candidates.

The effects of external behavior learning [6] and of unfounded set learning is clearly evident in the MCS benchmarks: the UFS check profits more from EBL than the explicit check, further adding to its advantage. By activating UFL (not possible in the explicit check) we gain another significant speedup.

Intuitively, consistent and inconsistent MCSs are dual, as for each candidate the explicit resp. UFS check fails, i.e., stops early, vs. for some (or many) candidates the check succeeds (stops late). However, the mixed results do not permit us to draw solid conclusions on the computational relationship of the evaluation methods.

Note that MCS topologies are bound to certain system sizes, and the difficulty of the instances varies among topologies; thus larger instances may have shorter runtimes.

Table 1. Benchmark Results (— indicates timeout (300s) of resp. instances)

(a) Inconsistent MCSs

#contexts	compute all answer sets					finding first answer set				
	explicit check		UFS check			explicit check		UFS check		
	plain	+EBL	plain	+EBL	+UFL	plain	+EBL	plain	+EBL	+UFL
3	9.08	6.11	6.29	2.77	0.85	4.01	2.53	3.41	1.31	0.57
4	89.71	36.28	80.81	12.63	5.27	53.59	16.99	49.56	6.09	1.07
5	270.10	234.98	268.90	174.23	18.87	208.62	93.29	224.01	32.85	3.90
6	236.02	203.13	235.55	179.24	65.49	201.84	200.06	201.24	166.04	28.34
7	276.94	241.27	267.82	231.08	208.47	241.09	78.72	240.72	66.56	16.41
8	286.61	153.41	282.96	116.89	69.69	201.10	108.29	210.61	103.11	30.98
9	—	208.92	—	191.46	175.26	240.75	112.08	229.14	76.56	44.73
10	—	—	—	289.87	289.95	—	125.18	—	75.24	27.05

(b) Consistent MCSs

#contexts	(no answer sets)				
	explicit check		UFS check		
	plain	+EBL	plain	+EBL	+UFL
3	8.61	4.68	7.31	2.44	0.50
4	86.55	48.53	80.31	25.98	1.89
5	188.05	142.61	188.10	94.45	4.62
6	209.34	155.81	207.14	152.32	14.39
7	263.98	227.99	264.00	218.94	49.42
8	293.64	209.41	286.38	189.86	124.23
9	—	281.98	—	260.01	190.56
10	—	274.76	—	247.67	219.83

(c) Argumentation (plain)

#args	all answer sets		first answer set	
	Explicit	UFS	Explicit	UFS
5	1.47	1.13	0.70	0.62
6	4.57	2.90	1.52	1.27
7	19.99	10.50	3.64	2.77
8	80.63	39.01	9.46	6.94
9	142.95	80.66	30.12	20.97
10	240.46	122.81	107.14	63.50

(d) Set Partitioning

	n	3	4	5	6	7	8	9	10	11	12	13	···	20
all answers	explicit	0.2	1.2	10.9	94.3	—	—	—	—	—	—	—	—	—
	+EBL	0.1	0.5	4.3	34.8	266.1	—	—	—	—	—	—	—	—
	UFS	0.1	0.1	0.2	0.3	0.8	1.8	4.5	11.9	32.4	92.1	273.9	—	—
	+EBL	0.1	0.1	0.1	0.1	0.2	0.2	0.3	0.4	0.6	0.8	1.2	···	11.1
first answer	explicit	0.1	0.2	0.7	4.3	26.1	163.1	—	—	—	—	—	—	—
	+EBL	0.1	0.2	0.8	4.9	31.1	192.0	—	—	—	—	—	—	—
	UFS	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	···	0.5
	+EBL	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	···	0.3

Abstract Argumentation. In this benchmark we compute ideal set extensions for randomized instances of abstract argumentation frameworks [4] of different sizes.

Table 1c shows average runtimes, each accumulated over 10 benchmark instances. In these instances, few unfounded sets exist, hence both the explicit and the UFS check often enumerate all candidates before they stop the search. As with MCS, the numbers of reduct model candidates and UFS candidates is in most cases equal, but the UFS check again enumerates its candidates faster; this explains the observed speedup.

Different from MCS, external learning prunes the search space only very little for these benchmarks, which can be explained by the structure of the encoding. Also UFL does not help much here, as few unfounded sets exist.

UNSAT. We also experimented with an encoding of the propositional UNSAT problem based on aggregates (not shown in figures), which was used in [11] to show the hardness result for the UFS decision problem. Here the optimizations discussed in Section 4 prune the search space (as in the set partitioning benchmark), which makes the UFS check enumerate fewer candidates, involving also fewer external atom calls.

6 Discussion and Conclusion

Related to our work is [15], which reduces stable model checking for disjunctive logic programs to unsatisfiability testing of CNFs, which like answer set checking from FLP-reducts is co-NP-complete [12]. The difference between ordinary disjunctive programs and FLP programs with external atoms is that co-NP-hardness holds already for Horn programs with nonmonotonic external atoms that are decidable in polynomial time. For computationally harder external atoms, the complexity increases relative to an oracle for the external function (see [12]). The approach of [15] is extended to conflict-driven learning and unfounded set checking in [3]. Here, two CLASP [13] instances generate and check answer set candidates. As model checking may become computationally harder in our setting, the results there do not carry over immediately.

We presented a new algorithm for deciding whether a model $\mathbf{A}$ of a HEX-program Π is a subset-minimal model of its FLP-reduct $f\Pi^{\mathbf{A}}$, adopting the notion of unfounded set in [11]. We realized unfounded set (UFS) checking by an encoding as a SAT instance, which produces candidate unfounded sets. Subsequently a (rather simple) postcheck decides whether there is indeed an unfounded set. Experiments have shown that this check is much more efficient than the explicit minimality check. We showed how to learn from identified unfounded sets, by deriving nogoods which guide future search in model generation and help avoiding to rediscover unfounded sets.

Our ongoing work includes interleaving UFS checks with the model generation process, i.e., on incomplete interpretations. If it is clear that a partial interpretation can never become an answer set, one can backtrack earlier; this may pay off for certain classes of instances. Furthermore, we investigate sufficient conditions to simplify the UFS check, aim at syntactic properties that are easy to check. Of particular interest are relevant program classes for which the UFS check can be skipped; this holds e.g. for programs without cyclic information flow through external atoms.

Another issue for future work is to study heuristics for guiding the search for an unfounded set. Currently, our implementation applies the same strategies as for the model generation task. Our experimental comparison with the explicit FLP check in terms of candidate sets considered, however, suggests that there might be room for improvement by employing specific choices. Developing appropriate such heuristics, and validating their effectiveness on candidate set enumeration remains to be explored.

References

1. Brewka, G., Eiter, T.: Equilibria in Heterogeneous Nonmonotonic Multi-Context Systems. In: AAAI 2007, pp. 385–390. AAAI Press (2007)
2. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Commun. ACM 54(12), 92–103 (2011)
3. Drescher, C., Gebser, M., Grote, T., Kaufmann, B., König, A., Ostrowski, M., Schaub, T.: Conflict-driven disjunctive answer set solving. In: KR 2008, pp. 422–432. AAAI Press (2008)
4. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. 77(2), 321–357 (1995)
5. Eiter, T., Fink, M., Ianni, G., Krennwallner, T., Schüller, P.: Pushing Efficient Evaluation of HEX Programs by Modular Decomposition. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 93–106. Springer, Heidelberg (2011)
6. Eiter, T., Fink, M., Krennwallner, T., Redl, C.: Conflict-driven ASP solving with external sources. Theor. Pract. Log. Prog. (to appear, 2012)
7. Eiter, T., Fink, M., Krennwallner, T., Redl, C., Schüller, P.: Improving HEX-Program Evaluation based on Unfounded Sets. Tech. Rep. INFSYS RR-1843-12-08. TU, Wein (2012)
8. Eiter, T., Fink, M., Schüller, P., Weinzierl, A.: Finding explanations of inconsistency in Multi-Context Systems. In: KR 2010, pp. 329–339. AAAI Press (2010)
9. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer-Set Programming. In: IJCAI 2005, pp. 90–96. Professional Book Center (2005)
10. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: Effective Integration of Declarative Rules with External Evaluations for Semantic-Web Reasoning. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 273–287. Springer, Heidelberg (2006)
11. Faber, W.: Unfounded Sets for Disjunctive Logic Programs with Arbitrary Aggregates. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 40–52. Springer, Heidelberg (2005)
12. Faber, W., Leone, N., Pfeifer, G.: Semantics and complexity of recursive aggregates in answer set programming. Artif. Intell. 175(1), 278–298 (2011)
13. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artif. Intell. 188, 52–89 (2012)
14. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generat. Comput. 9(3-4), 365–386 (1991)
15. Koch, C., Leone, N., Pfeifer, G.: Enhancing disjunctive logic programming systems by SAT checkers. Artif. Intell. 151(1-2), 177–212 (2003)
16. Leone, N., Rullo, P., Scarcello, F.: Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics and Computation. Inform. Comput. 135(2), 69–112 (1997)

Using Satisfiability for Non-optimal Temporal Planning

Masood Feyzbakhsh Rankooh, Ali Mahjoob, and Gholamreza Ghassem-Sani

Sharif University of Technology, Tehran, Iran
{masood.feyzbakhsh,alimahjoob}@gmail.com, sani@sharif.edu

Abstract. AI planning is one of the research fields that has benefited from employing satisfiability checking methods. These methods have been proved to be very effective in finding optimal plans for both classical and temporal planning. It is also known that by using planning-based heuristic information in solving SAT formulae, one can develop SAT-based planners that are competitive with state-of-the-art non-optimal planners in classical planning domains. However, using satisfiability for non-optimal temporal planning has not been investigated so far. The main difficulty in using satisfiability in temporal planning is the representation of time, which is a continuous concept. Previously introduced SAT-based temporal planners employed an explicit representation of time in the SAT formulation, which made the formulation too large for even very small problems. To overcome this problem, we introduce a novel method for converting temporal problems into a SAT encoding. We show how the size of the encoding can be reduced by abstracting out durations of planning actions. We also show that the new formulation is powerful enough to encode fully concurrent plans. We first use an off-the-shelf SAT solver to extract an abstract initial plan out of the new encoding. We then add the durations of actions to a relaxed version of the initial plan and verify the resulting temporally constrained plan by testing consistency of a certain related Simple Temporal Problem (STP). In the case of an inconsistency, a negative cycle within the corresponding Simple Temporal Network (STN) is detected and encoded into the SAT formulation to prevent the SAT solver from reproducing plans with similar cycles. This process is repeated until a valid temporal plan will be achieved. Our empirical results show that the new approach, while not using a planning-based heuristic function of any kind, is competitive with POPF, which is the state-of-the-art of expressively temporal heuristic planners.

Keywords: Satisfiability checking, AI planning, temporal planning, Simple Temporal Networks.

1 Introduction

Translating a given planning problem into a SAT formulation, solving the converted problem by using a SAT solver, and extracting the plan out of the SAT solution has proven to be an effective technique in classical planning. This approach was initially introduced in SATPLAN [1]. Since then, different techniques have been proposed to increase the speed of SAT-based planners. Some of the successful ideas include: extracting "mutex" relationships from a planning graph [2] and applying them to the

SAT encoding [3], operator splitting [4], and considering alternative semantics for parallel plans [5]. Most of the researches in employing satisfiability in AI planning have been concentrated on finding optimal or step-optimal [5] plans. However, it has been recently shown that by using planning-based heuristic information in solving SAT formulae, one can develop SAT-based planners that are competitive with the state-of-the-art in non-optimal planning [6].

Classical planning, however, has some major deficiencies when it comes to solving real-world problems. An important drawback of classical planning has its roots in the assumption of instantaneous actions and generating completely ordered plans. Since time is an important feature of our real life, one may reasonably expect real-world actions to have durations and real-world plans to be concurrent. The advent of temporal planning was a response to such expectations. Although the introduction of time and concurrency makes temporal planning to be essentially different from the classical planning, one can adopt well-developed classical planning methods to tackle temporal planning problems. In fact, many temporal planners have benefited from such approaches.

The so-called plan-space based planning methods have already been employed in temporal planning by many planners such as VHPOP [7] and CPT [8]. Some other planners have embedded the concept of time in the planning graphs and made it usable for temporal planning. Examples are: TGP [9] and TPSYS [10]. State-space based methods have also been utilized in temporal planning by several planners of which LPG [11], and TFD [12] are the most famous ones.

While satisfiability checking is a major approach in dealing with classical planning, it has been insufficiently exploited in the field of temporal planning. In fact, the only published SAT-based temporal planners are STEP [13] and T-SATPLAN[14], which are both optimal planners. Besides, none of well-developed techniques mentioned above for increasing the speed of SAT-based planners have been used in temporal planning.

Both STEP and T-SATPLAN use an explicit representation of time in their encodings. Generally speaking, in these SAT-based approaches, layer i is exactly one time unit ahead of layer $i+1$. If an action has a duration of d and its starting point is in layer i, then its ending point is restricted to be in layer $i+d$. Such a representation makes an enormous number of layers to be present in the SAT formulation while not being used in the final plan. This problem worsens as the duration ratio between the most durative action and the least one gets higher. Encoding more layers increases the size of the produced formula and may cause the planner to become highly memory sensitive. Even if the planner is not running out of memory, a linearly larger number of variables means an exponentially larger search space for finding a solution and as a result, a substantial decrease in the speed of the planner.

In this paper, we propose an alternative encoding approach in which instead of using an explicit representation of time, layers are used merely for determining the order of actions. In fact, in our representation, layer i is ahead of layer $i+1$, but the actual difference in their times is not fixed. Therefore, an action can have its start in layer i, and its end in layer $i+1$, regardless of its duration. In other words, the durations of actions are abstracted out at the time of producing the SAT encoding. We also show that by applying minor modifications to classical planning graphs, one can use the "mutex" information extracted from planning graphs to increase the planning speed of our new SAT representation of temporal planning problems.

Abstracting the durations of actions is not a new concept in the field of temporal planning. Some temporal planners have used such abstractions for guaranteeing the completeness in an important subset of temporal problems, called problems with required concurrency [15]. CRIKEY [16] was the first planner that separated the planning and scheduling processes by eliminating all durations in the planning phase. CRIKEY tests the validity of the resulting plan later in the scheduling phase, by solving a simple temporal problem. POPF [17], a descendant of CRIKEY, uses a linear programming method in its scheduling phase. Both POPF and CRIKEY take benefit from an enforced hill-climbing search method [18] in addition to standard best-first search mechanism. POPF is regarded as the state-of-the-art planner for solving the temporal problems with required concurrency.

The block diagram of our approach is depicted in figure 1. Similar to CRIKEY and POPF, our planner, named ITSAT (Implicit Time SAT planner), will need a scheduling step, when an abstract plan is produced by the SAT solver. The durations should be given back to the actions and an exact time value should be assigned to each of them by solving a simple temporal problem. However, such a consistent temporal assignment may not exist. As we show later in section 4, the reason of the inconsistency can be easily identified as negative cycles in the graph representation of the corresponding STP [19]. We then, add certain variables and clauses to the SAT formula in order to prevent the SAT solver from reproducing such cycles.

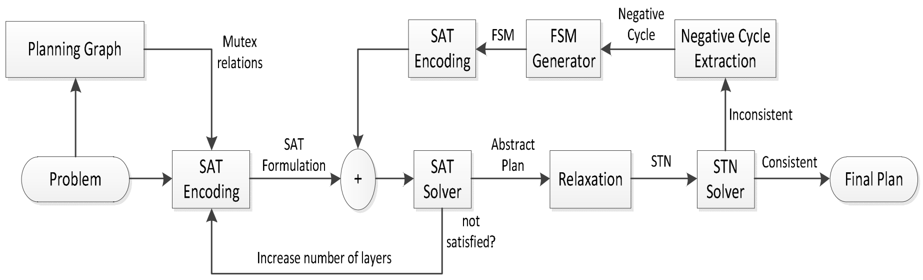


Fig. 1. The block diagram of ITSAT

2 Temporal Planning

We now give a formal description of temporal planning. Before that, we need a formal definition of a classical action. The definitions presented here are compatible with PDDL2.1 [20] where temporal actions can have separate preconditions and effects upon starting or ending.

Definition 1. A classical action o is a triple ($pre(o)$, $add(o)$, $del(o)$). All $pre(o)$, $add(o)$, and $del(o)$ are sets of facts and each fact is an atomic proposition. The set $pre(o)$ includes all the preconditions of o. The sets $add(o)$ and $del(o)$ contain the add and delete effects of o, respectively.

Definition 2. A temporal action a is defined by a positive rational duration $d(a)$, a starting event a^s, an ending event a^e, and an over-all condition $over(a)$. Each event is a classical action and the over-all condition is defined as a classical precondition. We also have: $action(a^s)=action(a^e)=a$. Action a is applicable in time t, if $pre(a^s)$ and $pre(a^e)$ are respectively held in time t and $t+d(a)$; and furthermore, $over(a)$ is held in open interval $(t,t+d(a))$. The application of a will cause the effects of a^s and a^e to take place in time t and $t+d(a)$, respectively.

Definition 3. A temporal planning problem is a quadruple $P=(A,F,I,G)$, where A is a set of temporal actions, F is the set of all the given facts, I is a classical initial state, and G is a set of classical goal conditions. The set of all events of P is defined by $E= \cup a \in A \{a^s,a^e\}$.

Definition 4. Suppose that $P=(A,F,I,G)$ is a temporal planning problem. A plan is represented by $\pi=\{(a_1,t_1),\ldots,(a_n,t_n)\}$, which means that action a_i is performed in time t_i. The makespan of π is calculated by relation (1).

$$Makespan(\pi)=max(t_i+d(a_i)) \qquad (1)$$

We say π is a valid plan for P if for each i, action a_i is applicable in time t_i, given that all the propositions in I are held in time 0. Furthermore, all the propositions in G must hold in time $Makespan(\pi)$. Plan π is concurrent if for some j and k we have: $t_k \leq t_j \leq t_k+d(a_k)$. A planning problem with only concurrent valid plans is called a problem with required concurrency.

3 SAT Encoding

We now explain how a temporal planning problem $P=(A,F,I,G)$ is encoded into its corresponding SAT formula. Suppose that E is the set of all starting and ending events of P, and we are constructing a formula with n conceptual layers. Variables and clauses that are essential for the soundness and completeness of the encoding are defined as follows.

3.1 Necessary Variables and Clauses

We define some variables to represent all the facts of P.

- For every fact $f \in F$, and every $1 \leq i \leq n$, a variable $X_{f,i}$ is defined. Assigning 1 to $X_{f,i}$ means that the fact f is true in the conceptual layer i.

Although previous SAT-based temporal planners, STEP and T-SATPLAN, encode each temporal action with a single variable, our encoding has two variables for the starting and ending events of each action, and another variable for representing the situations when the action is still running i. e., it is open.

- For every event $e \in E$, and every $1 \leq i < n$, a variable $Y_{e,i}$ is defined. Assigning 1 to $Y_{e,i}$ means that the event e is performed in the conceptual layer i.
- For every action $a \in A$, and every $1 \leq i < n$, a variable $W_{a,i}$ is defined. Assigning 1 to $W_{a,i}$ means that the action a is open (started but has not yet finished) in the conceptual layer i .

We also need some clauses for constraining the values of the variables of the first and the last layers in an appropriate way.

- For every fact $f \in I$, a clause $X_{f,1}$ is asserted to ensure the presence of initial facts in layer 1.
- For every fact $f \in F\text{-}I$, a clause $\neg X_{f,1}$ is added to ensure that only the facts of initial state can be present at layer 1.
- For every fact $g \in G$, a clause $X_{g,n}$ is asserted to ensure the presence of all the goal conditions in the last layer.
- For every action $a \in A$, a clause $\neg W_{a,1}$ is asserted to ensure that no action is open in layer 1.
- For every action $a \in A$, a clause $W_{a,n} \rightarrow Y_{a^e,n}$ is added to ensure that no action is remained unfinished in layer n.

We need certain clauses to handle the necessary conditions of performing an event in a layer and propagating the effects of that event to the next layer. The following clauses guarantee that if an event e is being performed in layer i, then its preconditions are available in layer i, and its effects are held in layer i+1.

- For every event $e \in E$, every $f \in pre(e)$, and every $1 \leq i < n$, a clause $Y_{e,i} \rightarrow X_{f,i}$ is added.
- For every event $e \in E$, every $f \in add(e)$, and every $1 \leq i < n$, a clause $Y_{e,i} \rightarrow X_{f,i+1}$ is asserted.
- For every event $e \in E$, every $f \in del(e)$, and every $1 \leq i < n$, a clause $Y_{e,i} \rightarrow \neg X_{f,i+1}$ is asserted.

Note that these constraints are not sufficient to make a valid plan. Many planning representations, including PDDL2.1 [20], forbid a proposition to be deleted by an event if it is needed by at least one other event at the same time. If the effects of each event take place only in the next layer, detecting such conflicts will be impossible. To overcome this flaw, quite similar to previous classical and temporal SAT-based planners, we use clauses that explicitly encode the mutual exclusion between conflicting events. As an example, if event a deletes a precondition of event b, then the clause $Y_{a,i} \rightarrow \neg Y_{b,i}$ is added to the formula for every i to ensure a and b will never be performed at the same layer.

Beside preconditions of an event, some other conditions must hold to enable that event to occur in a certain layer. We assume that two copies of the same ground action can never be concurrent. Even though PDDL2.1 allows this kind of concurrency, none of the domains we observed during our empirical analysis had such requirements.

- For every $1 \leq i < n$ and every event $e \in E$, if e is the starting event of action a, a clause $Y_{e,i} \rightarrow \neg W_{a,i} \wedge W_{a,i+1}$ is added to ensure that two copies of action a will not be concurrent. This clause also assures us that if a is started in layer i, it will be considered as an open action in layer $i+1$.
- For every $1 \leq i < n$ and every event $e \in E$, if e is the ending event of action a, a clause $Y_{e,i} \rightarrow W_{a,i} \wedge \neg W_{a,i+1}$ is added to ensure that only open actions can be ended; moreover, if an action is ended in layer i, it will no longer be considered as an open action in layer $i+1$.
- For every $1 \leq i < n$, every action $a \in A$, and every fact $f \in over(a)$, we add a clause $W_{a,i} \rightarrow X_{f,i}$. This clause assures us that when an action is still open, its over-all conditions are maintained.

Finally, several kinds explanatory frame axioms are needed for preventing the values of the variables from changing without any reason.

- For every $1 \le i < n$ and every $f \in F$, we add a clause $\neg X_{f,i} \wedge X_{f,i+1} \rightarrow (\bigvee_{f \in add(e)} Y_{e,i})$ to prevent any fact from being arbitrarily asserted in layer i.
- For every $1 \le i < n$ and every $f \in F$, we add a clause $X_{f,i} \wedge \neg X_{f,i+1} \rightarrow (\bigvee_{f \in del(e)} Y_{e,i})$ to prevent any fact from being arbitrarily deleted from layer i.
- For every $1 \le i < n$ and every $a \in A$, we add a clause $\neg W_{a,i} \wedge W_{a,i+1} \rightarrow Y_{e,i}$, where e is the starting event of action a. This clause ensures that no action can become open in layer $i+1$, without having its starting event in layer i.
- For every $1 \le i < n$ and every $a \in A$, we add a clause $W_{a,i} \wedge \neg W_{a,i+1} \rightarrow \mathrm{Y}_{\mathrm{e,i}}$, where e is the ending event of action a. This clause ensures that if an action is open in layer i but not open in layer $i+1$, its ending event must be present in layer i.

It should be obvious from our SAT encoding that durations of actions play no role in the encoding. However, every other constraint of temporal planning is encoded into the SAT formulation. In other words, if we solve the SAT formula with a SAT solver, the resulting plan will be a valid temporal plan, unless there is an inconsistency among the durations of actions. We will discuss this issue in more details in section 4.

3.2 Mutex Relations as Auxiliary Clauses

Omitting any of the clauses stated in section 3.1 causes planner to lose either its soundness or its completeness. In this section, we present other kinds of clauses that can increase the speed of finding a valid plan but are not obligatory for preserving soundness or completeness of the planner. For this purpose, by using a planning graph [2], we automatically find pairs of propositions that cannot be true in a certain layer of our SAT encoding. A thorough description of the classical planning graph can be found in [2].

Due to the similarity between the structures of planning graph and SAT encoding of planning problems, one can add the SAT encoding of mutex relations to the corresponding layer of SAT formula and thereby prevent unnecessary effort in searching for a valid plan. In other words, if p and q are mutex in layer i, the clause $X_{p,i} \rightarrow \sim X_{q,i}$ can be added to the formula.

Planning graphs have been effectively used in SAT-based classical planners before [3,21]. However, they have not been employed in SAT-based temporal planning. Note that since we abstract out the duration of actions, the structure of our encodings becomes very similar to a classical planning graph. In ITSAT, we convert a temporal problem to a classical one and achieve mutex relations by constructing a slightly modified classical planning graph.

We split any temporal action a into two classical actions a^s and a^e, which are respectively starting and ending events of a. In addition to their normal effects and preconditions, a^s adds an exclusive proposition named p_a, which is required and deleted by a^e. These new propositions are not propagated by the classical no-op actions. Instead, for each action a, p_a is propagated by a no-op action that requires the over-all conditions of a in addition to requiring and adding p_a. The new kind of no-op actions is used to cover the overall conditions of actions, while reasoning about mutex relations.

4 The Scheduling Phase

Assume that we have solved a SAT formula produced by the encoding phase described is section 3 and obtained a solution for it. Let $e_1,\ldots,e_m$ be the events whose corresponding SAT variables have a value of 1 in the solution. Suppose that we have given different names to different occurrences of the same action in the solution, so all the events $e_1,\ldots,e_m$ are unique. Let $layer(e_i)$ be the number of layer in which e_i has occurred.

Events $e_1,\ldots,e_m$ must then be scheduled in order to produce a valid temporal plan. In other words, exact time values should be assigned to the events. However, these time values cannot be assigned arbitrarily. In fact, there exist certain constraints between these values.

4.1 Temporal Constraints

Let the time value assigned to event e_i be $T(e_i)$. Since it is assumed in the generation of a SAT formula, that layer i is ahead of layer $i+1$, a straightforward way to maintain the validity of the final plan is to schedule each event of layer i, before all events of layer $i+1$. Now, suppose that $layer(e_k)=i$, and $layer(e_j)=i+1$ and furthermore, e_j and e_k have no causal relationship with each other. Now, if the order between e_j and e_k is eliminated, not only will the validity of the plan remain unchanged, but also it will be more likely to find a consistent assignment of time values for the events.

In order to pursue the above idea, we define new ordering constrains between different pairs of events. For satisfying all these constraints, we solve an instance of a Simple Temporal Problem (STP) [19]. Each STP is associated with a weighted graph named a Simple Temporal Network (STN). We construct an STN in which each node corresponds to an event. For each i, let x_i be the node that is corresponding to event e_i in the STN. Let ϵ be an arbitrary small rational number and e_j and e_k be different events such that $layer(e_j)\geq layer(e_k)$. The constraint $T(e_j) \geq T(e_k)+\epsilon$ must hold if one of following condition is satisfied:

1. e_k adds a precondition of e_j .
2. e_j deletes a precondition of e_k.
3. either e_j or e_k delete any effect of the other.
4. e_j is a starting event and e_k deletes an all-over condition of $action(e_j)$

Consequently, an edge with the weight $-\epsilon$ will be present in the STN from the node x_k to the node x_j . Moreover, if one of the following conditions holds, the constraint $T(e_j)\geq T(e_k)$ will be necessary, and an edge with weight 0 will be present in the STN from the node x_k to the node x_j.

5. e_k is an ending event and e_j deletes an all-over condition of $action(e_k)$.
6. e_j is a starting event and e_k adds an all-over condition of $action(e_j)$.

As before, in addition to the constraints stated above, the following constraint must also hold if e_k and e_j are respectively the starting and ending effect of certain action a to ensure that the durations of a in the final plan is set correctly:

7. $T(e_j)\geq T(e_k)+d(a)$
8. $T(e_k)+d(a)\geq T(e_j)$

These constraints will be inserted into the STN by adding an edge with the weight –*d(a)* from node x_k to node x_j, and another edge with weight *d(a)* from node x_j to node x_k. We also add a reference node x_0 to the constructed STN which has an edge with weight 0 to every other node. A solution of the STP can be found by computing the length of the shortest path form x_0 to all other nodes. Suppose that such shortest paths exist and the length of the shortest from x_0 to x_i is shown by *distance*(x_0, x_i). For each event e_i, we assign -*distance*(x_0, x_i) to $T(x_i)$. Constraints 1 to 8 guarantee that the resulting plan has all the specifications of a valid temporal plan. However, there are situations in which such shortest paths do not exist. It happens when the STN has a negative cycle. In these situations the STP is inconsistent and consequently, all temporal constraints can not be satisfied at the same time. An example of such cases is depicted in figure 2.

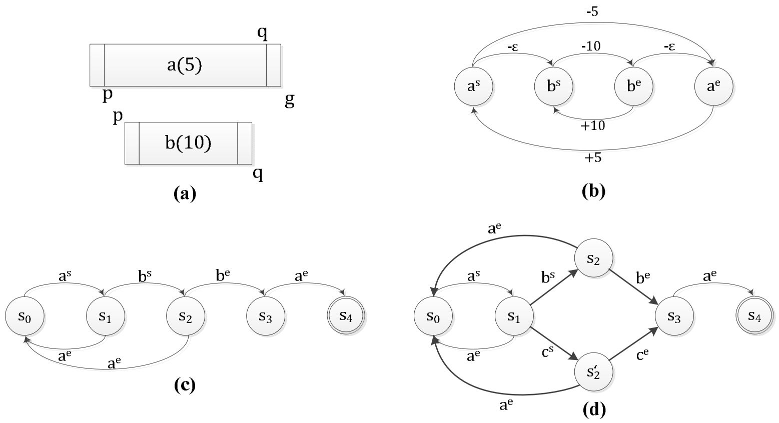


Fig. 2. Negative cycle in an STN, and the detecting FSM

Suppose that 1) action *a* adds proposition *p* and *g* respectively by its starting and ending event. 2) *a* needs proposition *q* as a precondition for its ending event. 3) action *b* that requires *p* upon beginning and adds *q* upon ending. 4) durations of actions *a*, and *b*, are 5 and 10, respectively. 5) action *c* is identical to *b* except for its duration which is 15. 6) The goal of planning is reaching fact *g*. If a SAT formula with four layers is produced for this problem, then a SAT solver can satisfy the formula by putting the starting event of *a* in layer 1, the starting event of *b* in layer 2, the ending event of *b* in layer 3, and the ending event of *a* in layer 4. This means action *b* must be entirely performed inside of action *a*. However, this situation is impossible considering the fact that duration of *a* is less than that of *b*. The abstract plan produced by SAT solver is depicted in figure 2(a). Preconditions and effects of each action are respectively written above and below it.

The invalidity of this plan is caused by the fact that all the durations are abstracted out when the formula is being produced. In fact, the SAT solver assumes that the execution of ending event of *a* can be postponed, as long as is needed. The STN constructed for the plan of figure 2(a) is depicted in figure 2(b). $a^s b^s b^e a^e a^s$ is a negative cycle with total weight $-5-2\epsilon$. Note that if action *b* had been replaced by action *c* in fig 1(a), we would have had another negative cycle with weight $-10-2\epsilon$.

4.2 Negative Cycle Prevention

As it was stated before, if the STN of an abstract plan has a negative cycle, it cannot be transformed to a valid temporal plan. In such cases, we run the SAT solver again to find a different solution. This can be done by adding an extra blocking clause to the SAT formula so that at least one of the events of invalid plan cannot occur in its current layer. Nevertheless, after adding such a blocking clause, the SAT solver can produce new plans that are not essentially different from previous one. For instance, consider the example given in figure 2(a). Suppose a^s,b^s,b^e, and a^e are true in layers 1 to 4, respectively. Assume that the SAT formula have 5 layers. If we forbid the exact occurrence of this plan, a new temporally invalid abstract plan can still be produced by shifting a^e to layer 5 and maintaining other events in their current layers. The new solution will have the same negative cycle. In fact, the main cause of the invalidity of the plan has remained unchanged. On the other hand, by replacing action *b* by action *c*, another temporally invalid abstract plan can be produced. This time, the cycle $a^sb^sb^ea^ea^s$ has been replaced by cycle $a^sc^sc^ea^ea^s$. However, since *b* and *c* have identical preconditions and effects, and *c* has a longer duration, we could have inferred from structure of $a^sb^sb^ea^ea^s$ that $a^sc^sc^ea^ea^s$ is also a negative cycle. We now show that negative cycles of particular structure can be prevented more effectively. We also explain how other similar negative cycles can prevented.

The negative cycles in which we are interested are quite similar to the one depicted in figure 2(b). Such cycles happen when several actions of a temporal plan are to be executed consecutively within the execution of another action. These cycles can be regarded as sequences of events. For instance, in the previous example, each time that the a^s, a^e, b^s, and b^e occur with the order $a^sb^sb^ea^e$, we can be sure that the corresponding STN has a negative cycle. Fortunately, such sequences can be detected by using a simple Finite State Machine (FSM). For example, the FSM depicted in figure 2(c) can detect the negative cycle of figure 2(b).

On the other hand, in the STN depicted in figure 2(b) there exists an edge from a^s to b^s only because condition number 1 (stated in section 4.1) is held between the two events. The same condition causes an edge from b^e to a^e. It should be clear that if we replace action *b* by any other action with the same conditions but with a longer duration, the result will be again a negative cycle. Here, action *c* fulfills the stated requirements. We call *c* an alternative for *b*. In fact, ITSAT automatically detects all the alternatives for each action in a given negative cycle. The FSM that detects both $a^sb^sb^ea^e$ and $a^sc^sc^ea^e$ is depicted in figure 2(d). Note that in general, more than one action, each of which having several alternatives, can occur consecutively inside a covering action.

To prevent a cycle, one can start from the first layer and follow the transitions that occur in the FSM. A certain transition occurs in layer *i*, only if the value of the variable corresponding to the label of that transition is equal to one.

If the FSM is constructed, its corresponding negative cycles can be prevented by making the SAT solver simulate the function of the FSM and banning it from being in the terminal state. In order to do so, we must add extra variables and clauses to the SAT formula.

Let $s_0,\ldots,s_m$ be the states of an FSM that detects a certain negative cycle. Suppose that s_0 and s_m are the initial state and the terminal state of the FSM, respectively. Let *T* be the set of all transitions of the FSM, and $label(s_i,s_j)$ denote the event that causes the transition from s_i to s_j. Furthermore, assume that $exit(s_i)$ denotes all the transitions through which the FSM goes from s_i to another state. The following variables and clauses are sufficient for preventing the SAT solver from reproducing negative cycles that are associated with the FSM.

- For every $1 \leq i \leq n$ and every $0 \leq j \leq m$, we define a variable $S_{j,i}$. Assigning one to $S_{j,i}$ means that the FSM can be in state j after observing all the events that are present in the layers prior to i.
- For every $1 \leq i \leq n$, every $0 \leq j \leq m$, and every $(s_j,s_k) \in exit(s_j)$, we add a clause $S_{j,i} \wedge e \rightarrow \neg S_{j,i+1} \wedge S_{k,i+1}$, where $e=lable(s_j,s_k)$, to ensure that in each layer, FSM performs only correct transitions.
- For every $1 \leq i \leq n$ and every $0 \leq j \leq m$, we add a clause $S_{j,i} \wedge (\neg \bigvee_{e \in E} e) \rightarrow S_{j,i+1}$, where $E=\{label(s_j,s_k) \mid (s_j,s_k) \in exit(s_j)\}$, to ensure that FSM stays in state s_j when it is necessary.
- For every $1 \leq i \leq n$ and every $0 \leq j \leq m$, we assert a clause $\neg S_{j,i} \wedge S_{j,i+1} \rightarrow \bigvee_{(k,e) \in N} (S_k \wedge e)$ where $N=\{(k,e) \mid e=lable(s_k,s_j)\}$. This clause prevents the FSM from arbitrarily moving to some state.
- We assert a clause $S_{0,1}$ to ensure that the FSM will start from its initial state in layer one. We also add $\neg S_{j,1}$ for every $1 \leq j \leq m$, to prevent FSM from being in any other state in layer one.
- For every $1 \leq i \leq n$, we assert a clause $\neg S_{m,i}$ to ensure that the FSM will never go to its terminal state, detecting a negative cycle.

After constructing the SAT formulation of the FSM, we add it to our previous encoding of the problem. Then, SAT solver is called again to satisfy the new formula. The process of solving the formula and adding negative cycle detectors (i. e., appropriate FSMs) is repeated until a valid plan will be achieved.

5 Empirical Results and Discussion

ITSAT has been implemented using C++ programming language. We have used an open source SAT solver, MINISAT2 [22] for solving SAT formulas. To evaluate ITSAT, it has been compared with POPF [17] on the temporal problem sets of recent international planning competitions plus two more domains (i.e., driverlogshift and matchlift) defined by the Strathclyde planning group [23]. POPF is currently one of the most efficient heuristic temporal planners. The experiments were conducted on a 3.1GHz corei5 CPU with 4GB main memory and 30 minutes time-limit for solving each problem. All the results are presented in Table 1.

ITSAT and POPF are compared based on the number of problems they can solve in each domain and also by the total score given to each planner using the scoring strategy of recent IPCs: if a planner cannot solve a problem, it will get score 0 for it; Otherwise, the score of planner is equal to the makespan of the best plan found for the problem divided by the makespan of the plan found by the planner. The last columns of Table 1, shows the makespan ratio, averaged only on problems that are solved by both planners. Ratios less than one indicate better average quality of the solutions produced by ITSAT in comparison with that of POPF.

Table 1. Comparing ITSAT to POPF

Domain	IPC	# of Problems	# of Problems Solved		Total Score		makespan ratio
			ITSAT	POPF2	ITSAT	POPF2	
zenotravel	2004	20	13	13	**12.22**	11.52	**0.96**
driverlog		20	15	15	**13.96**	11.63	**0.76**
rovers		20	16	**19**	**15.97**	14.04	**0.87**
depots		22	**12**	7	**11.77**	5.7	**0.85**
airport	2006	50	**22**	15	**22**	14.44	**0.95**
satellite		36	13	**14**	**12.63**	8.96	**0.72**
pegsol	2011	20	19	19	**19**	18.62	**0.98**
crewplanning		20	15	**20**	14.31	**20**	1.05
openstacks		20	0	**20**	0	**20**	---
parking		20	0	**20**	0	**20**	---
elevators		20	0	**2**	0	**2**	---
floortile		20	**20**	0	**20**	0	1.33
storage		20	**5**	0	**5**	0	---
matchcellar		20	4	**20**	4	**20**	1
sokoban		20	1	**3**	1	**2.87**	**0.87**
parcprinter		20	**15**	0	**15**	0	---
turnandopen		20	3	**9**	3	**8.72**	**0.9**
tms		20	**20**	4	**20**	4	1
driverlogshift	Strathclyde	10	10	10	**9.83**	9.03	**0.95**
matchlift		14	**14**	13	**13.96**	12.08	**0.98**
total		482	217	**223**	**213.65**	203.61	---

As it is shown in table (1), the total score of ITSAT is greater than that of POPF. In fact, ITSAT has outperformed POPF in 13 out of 20 domains. Furthermore, considering the official results of the most recent planning competition (IPC 2011) [24], none of participating planner has solved as many problems as ITSAT has solved in tms, parcprinter, and floortile domains. These achievements are very promising because, unlike other efficient non-optimal classical or temporal planners such as POPF, LPG, TFD, FF [18], etc., in current version of ITSAT no planning based heuristic function has been used.

On the other hand, since the encoding used in ITSAT can also represent fully concurrent plans, ITSAT is in fact a temporally expressive planner, i.e., it is able to solve problems with required concurrency. This can be verified by the results of ITSAT in driverlogshift, matchlift, matchcellar, tms, and turnandopen domains, which all have required concurrency. Note that each planner has outperformed the other in two of these five domains. Thus, in temporally expressive domains, ITSAT is comparable with POPF, which is currently the state-of-the-art planner in such domains.

It should be clear that the size of our encoding and the amount of the memory needed for saving the encoding increase as the number of ground actions is increased. That's why ITSAT performs rather inefficiently in domains where the number of ground actions is relatively high. Such domains include: elevators, storage, openstacks, parking, and turnandopen. ITSAT runs out of memory in some of these domains. In addition to the memory consumption problem, we observed that the speed of the SAT solver declines drastically as the number of variables increases. To tackle this problem, we suggest using a lifted form of actions, instead of fully ground actions. The idea is very similar to the so-called operator splitting technique [4], which has been used in classical planning. Operator splitting has been shown to be quite effective in reducing the number of variables. We consider using lifted actions to be a promising extension of the current version of ITSAT.

6 Conclusion

In this paper, we introduced a new method for producing SAT encoding of a given temporal planning problem. In the new encoding, durations of all actions are initially abstracted out in order to construct a compact SAT formula. The produced SAT formula is then solved by an off-the-shelf SAT solver. The resulting abstract plan is then relaxed by deleting unnecessary ordering constraints. In order to assign the exact time values to the events of the relaxed plan, based on their order and action durations, a Simple Temporal Network is constructed and resolved. We showed that possible negative cycles of such STNs can be detected by using certain Finite State Machines. We also introduce an automatic method for encoding such FSMs. We showed how adding FSMs encodings to the SAT formula could prevent the SAT solver from reproducing solutions with similar negative cycles. Our empirical results showed that our new planner ITSAT, while not using any planning based heuristic functions, is comparable with POPF, which is currently the state-of-the-art in non-optimal temporal planning.

References

1. Kautz, H., Selman, B.: Planning as Satisfiability. In: Proceedings of 10th European Conference on Artificial Intelligence, pp. 359–363. IOS Press (1992)
2. Blum, A., Furst, M.: Fast Planning Through Planning Graph Analysis. Artificial Intelligence 90(1-2), 281–300 (1997)
3. Kautz, H., Selman, B.: Unifying SAT-based and Graph-based Planning. In: Proceedings of 16th International Joint Conference on Artificial Intelligence, pp. 318–325. AAAI Press (1999)
4. Robinson, N., Gretton, C., Pham, D.N., Sattar, A.: SAT-Based Parallel Planning Using a Split Representation of Actions. In: Proceedings of 19th International Conference on Automated Planning and Scheduling. AAAI Press (2009)
5. Rintanen, J., Heljanko, K., Niemelä, I.: Parallel Encodings of Classical Planning as Satisfiability. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 307–319. Springer, Heidelberg (2004)
6. Rintanen, J.: Planning with Specialized SAT Solvers. In: Proceedings of 25th AAAI Conference on Artificial Intelligence. AAAI Press (2011)
7. Younes, H.L.S., Simmons, R.G.: VHPOP: Versatile Heuristic Partial Order Planner. Journal of Artificial Intelligence Research 20, 405–430 (2003)
8. Vidal, V., Geffner, H.: Branching and pruning: An optimal temporal POCL planner based on constraint programming. Artificial Intelligence 170(3), 298–335 (2006)
9. Smith, D.E., Weld, D.S.: Temporal planning with mutual exclusion reasoning. In: Proceedings of 16th International Joint Conference on Artificial Intelligence, pp. 326–337. AAAI Press (1999)
10. Garrido, A., Fox, M., Long, D.: A temporal planning system for durative actions of PDDL2.1. In: Proceedings of 15th European Conference on Artificial Intelligence, pp. 586–590. IOS Press (2002)

11. Gerevini, A., Serina, I.: LPG: a Planner based on Local Search for Planning Graphs. In: Proceedings of 6th International Conference on Artificial Intelligence Planning Systems, pp. 13–22. AAAI Press (2002)
12. Eyerich, P., Mattmuller, R., Roger, G.: Unifying Context-Enhanced Additive Heuristic for Temporal and Numeric Planning. In: Proceedings of 19th International Conference on Automated Planning and Scheduling. AAAI Press (2009)
13. Huang, R., Chen, Y., Zhang, W.: An optimal temporally expressive planner: Initial results and application to P2P network optimization. In: Proceedings of 19th International Conference on Automated Planning and Scheduling. AAAI Press (2009)
14. Mali, A.D., Liu, Y.: T-SATPLAN: A SAT-based Temporal Planner. International Journal of Artificial Intelligence Tools 15(5), 779–802 (2006)
15. Cushing, W., Kambhampati, S., Weld, D.S.: When is temporal planning really temporal? In: Proceedings of 20th International Joint Conference on Artificial Intelligence, pp. 1852–1859. AAAI Press (2007)
16. Halsey, K., Long, D., Fox, M.: Managing Concurrency in Planning Using Planner-Scheduler interaction. Artificial Intelligence 173(1), 1–44 (2009)
17. Coles, A.J., Coles, A., Fox, M., Long, D.: Forward-Chaining Partial-Order Planning. In: Proceedings of 20th International Conference on Automated Planning and Scheduling, pp. 42–49. AAAI Press (2010)
18. Hoffmann, J., Nebel, B.: The FF Planning System: Fast Plan Generation Through Heuristic Search. Journal of Artificial Intelligence Research 14, 253–302 (2001)
19. Dechter, R., Meiri, I., Pearl, J.: Temporal Constraint Networks. Artificial Intelligence 49(1-3), 61–95 (1991)
20. Fox, M., Long, D.: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. Journal of Artificial Intelligence Research 20, 61–124 (2003)
21. Kautz, H., Selman, B., Hoffmann, J.: SatPlan: Planning as Satisfiability. IPC (2006)
22. Eén, N., Sörensson, N.: An Extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004)
23. Strathclyde Planning Group, `http://planning.cis.strath.ac.uk`
24. Coles, A., Coles, A., Olaya, A.G., Jiménez, S., López, C.L., Sanner, S., Yoon, S.: A Survey of the Seventh International Planning Competition. AI Magazine 33(1) (2012)

How to Exploit Parametric Uniformity for Maximum Entropy Reasoning in a Relational Probabilistic Logic*

Marc Finthammer and Christoph Beierle

Dept. of Computer Science, FernUniversität in Hagen, Germany

Abstract. The relational probabilistic conditional logic FO-PCL employs the principle of maximum entropy (ME). We show that parametric uniformity of an FO-PCL knowledge base $\mathcal{R}$ can be exploited for solving the optimization problem required for ME reasoning more efficiently. The original ME optimization problem containing a large number of linear constraints, one for each ground instance of a conditional, can be replaced by an optimization problem containing just one linear constraint for each conditional. We show that both optimization problems have the same ME distribution as solution. An implementation employing Generalized Iterative Scaling illustrates the benefits of our approach.

1 Introduction

Various extensions of propositional probabilistic logic to the relational case have been proposed. Besides well-known approaches like Markov Logic Networks or Bayesian Logic Programs, the *principle of maximum entropy (ME)* [19] has also been employed, e.g. [15,16]. Here, we use the probabilistic relational conditional logic FO-PCL [8,9].

Example 1 (Elephant keeper). The elephant keeper example, adapted from [4] and also [8], models the relationships among elephants in a zoo and their keepers. Elephants usually like their keepers, except for keeper Fred. But elephant Clyde gets along with everyone, and therefore he also likes Fred. This is expressed by $\mathcal{R}_{EK}$ containing:

$EK_1 : \langle (\mathit{likes}(E, K))[0.9], K \neq \mathit{fred} \rangle$
$EK_2 : \langle (\mathit{likes}(E, \mathit{fred}))[0.05], E \neq \mathit{clyde} \rangle$
$EK_3 : \langle (\mathit{likes}(\mathit{clyde}, \mathit{fred}))[1.0], \top \rangle$

EK_1 and EK_2 represent the statements that elephants normally like their keeper, except for Fred. EK_3 represents the exceptional fact that elephant Clyde likes Fred. The constraint formulas $K \neq \mathit{fred}$ and $E \neq \mathit{clyde}$ of EK_1 and EK_2 make sure that there will not be an inconsistency when substituting the variables with constants. If the constraint formula of EK_2 was $\top$ instead (denoting a tautology), and if E was substituted by clyde, the result would be the ground conditional $\langle (\mathit{likes}(\mathit{clyde}, \mathit{fred}))[0.05], \top \rangle$, which would be inconsistent with EK_3 since no probability distribution can satisfy $\mathit{likes}(\mathit{clyde}, \mathit{fred})$ both with probability 1.0 and with probability 0.05.

The ME semantics of FO-PCL is defined by referring to all groundings of a conditional in a knowledge base $\mathcal{R}$. Thus, for ME reasoning, the resulting optimization problem

* The research reported here was partially supported by the DFG (grant BE 1700/7-2).

involves a potentially huge number of linear constraints, one for *each ground instance* of a conditional. In this paper, we show that parametric uniformity of $\mathcal{R}$ [8,17] can be exploited to significantly simplify the ME model computation by having to consider only *one* linear constraint for each conditional, opening the road for using techniques of lifted inference [20,3] and tailored optimization algorithms like Generalized Iterative Scaling [2].

After briefly sketching the basics of FO-PCL as far as needed here (Sec. 2), we use feature functions to develop a series of equation systems expressing the FO-PCL satisfaction relation and exploiting parametric uniformity (Sec. 3 and 4). In Sec. 5, we use an implementation of the maximum entropy computation based on the final optimized equation system in order to illustrate the benefits of our approach, and in Sec. 6, we conclude and point out further work.

2 Background: FO-PCL

FO-PCL Syntax. FO-PCL uses function-free signatures of the form $\Sigma = (S, D, \mathit{Pred})$ where S is a set of sorts, $D = \bigcup_{s \in S} D^{(s)}$ is a finite set of (disjoint) sets of sorted constant symbols, and Pred is a set of predicate symbols, each coming with an arity of the form $s_1 \times \ldots \times s_n \in S^n$ indicating the required sorts for the arguments. Variables $\mathcal{V}$ also have a unique sort, and all formulas and variable substitutions must obey the obvious sort restrictions. In the following, we will adopt the unique names assumption, i. e. different constants denote different elements.

An FO-PCL *conditional* $R = \langle(\phi_R|\psi_R)[\xi_R], C_R\rangle$ is composed of a *premise* ψ_R and a *conclusion* ϕ_R, which are quantifier and function free first-order formulas (over Σ and $\mathcal{V}$) without equality, a probability value $\xi_R \in [0, 1]$, and a *constraint formula* C_R which is a quantifier-free first-order formula using only the equality predicate. For $\neg(V = X)$ we also write $(V \neq X)$, and $\top$ resp. $\bot$ denote a tautology resp. a contradiction. An *FO-PCL knowledge base* is a pair $(\Sigma, \mathcal{R})$ where $\mathcal{R}$ is a set of conditionals over $\Sigma, \mathcal{V}$. In the following, we will call just $\mathcal{R}$ a knowledge base and Σ will be given by the context. We will use the notation $C_R \models V \neq c$ to express that under the unique names assumption, the constraint formula C_R of a conditional R entails the constraint $V \neq c$, and $C_R \not\models V \neq c$ means that C_R does not entail the constraint. Let $\mathit{const}(\mathcal{R})$ denote the set of all constants appearing in the FO-PCL conditionals in $\mathcal{R}$. The constants in $\mathit{const}(\mathcal{R})$ are called *specific constants*, whereas the constants in $D \setminus \mathit{const}(\mathcal{R})$ are *generic constants*.

The constraint formula makes it possible to explicitly express that a generic conditional is not applicable with respect to a particular individual. Without constraint formulas, having a generic conditional and a corresponding conditional for a specific individual, that specific conditional might formally contradict the general one when considering all instances (cf. Example 1). When the constraint formula of a ground instance of R evaluates to true, that instance is called *admissible*, and $\mathit{gnd}(R)$ denotes the set of all admissible instances of R (over Σ), in the following also just called instances.

FO-PCL Models. The Herbrand base $\mathcal{H}(\mathcal{R})$ is the set of all atoms in all $\mathit{gnd}(R_k)$ with $R_k \in \mathcal{R}$, and every subset $x \subseteq \mathcal{H}(\mathcal{R})$ is a Herbrand interpretation, defining a logical semantics for $\mathcal{R}$. The set $X(\mathcal{R}) = \{x \mid x \subseteq \mathcal{H}(\mathcal{R})\}$ denotes the set of all

Herbrand interpretations. The probabilistic semantics of $\mathcal{R}$ is a possible world semantics [11] where the ground atoms in $\mathcal{H}(\mathcal{R})$ are binary random variables. An FO-PCL *interpretation* $p_{X(\mathcal{R})}$ of $\mathcal{R}$ is thus a probability distribution over $X(\mathcal{R})$.

For $R_k \in \mathcal{R}$ and every $g_{R_k} \in gnd(R_k)$, let θ_{R_k} be an admissible ground substitution for the variables in R_k so that $g_{R_k} = \langle(\theta_{g_{R_k}}(\phi_{R_k}) \mid \theta_{g_{R_k}}(\psi_{R_k}))[\xi_{R_k}], \top\rangle$. Then $p_{X(\mathcal{R})}$ satisfies R_k iff for every instance $g_{R_k} \in gnd(R_k)$ it holds:

$$p_{X(\mathcal{R})}(\theta_{R_k}(\phi_{R_k}) \wedge \theta_{R_k}(\psi_{R_k})) = \xi_{R_k} \cdot p_{X(\mathcal{R})}(\theta_{R_k}(\psi_{R_k}))$$

Note that for the case of $p_{X(\mathcal{R})}(\theta_{R_k}(\psi_{R_k})) > 0$, this equation is equivalent to $\frac{p_{X(\mathcal{R})}(\theta_{R_k}(\phi_{R_k}) \wedge \theta_{R_k}(\psi_{R_k}))}{p_{X(\mathcal{R})}(\theta_{R_k}(\psi_{R_k}))} = \xi_{R_k}$ and thus to $p_{X(\mathcal{R})}((\theta_{g_{R_k}}(\phi_{R_k}) \mid \theta_{g_{R_k}}(\psi_{R_k}))) = \xi_{R_k}$, expressing conditional probability. $p_{X(\mathcal{R})}$ is a *model* of $\mathcal{R}$ if it satisfies every $R_k \in \mathcal{R}$.

Maximum Entropy Model. A knowledge base $\mathcal{R} = \{R_1, \ldots, R_m\}$ may have many different models, and the principle of maximum entropy [19,15,13,14] provides a method to select a model that is optimal in the sense that it is the most unbiased one. The entropy of a probability distribution $p_{X(\mathcal{R})}$ is defined as

$$H(p_{X(\mathcal{R})}) = - \sum_{x \in X(\mathcal{R})} p_{X(\mathcal{R})}(x) \log p_{X(\mathcal{R})}(x)$$

The computation of the uniquely determined maximum entropy model

$$p^*_{X(\mathcal{R})} = \arg \max_{p_{X(\mathcal{R})} \models \mathcal{R}} H(p_{X(\mathcal{R})}) \tag{1}$$

is an optimization problem whose solution $p^*_{X(\mathcal{R})}$ can be represented by a Gibbs distribution [10]:

$$p^*_{X(\mathcal{R})}(x) = \frac{1}{Z} \exp\left(\sum_{k=1}^{m} \sum_{g_{R_k} \in gnd(R_k)} \lambda_{g_{R_k}} f_{g_{R_k}}(x) \right) \tag{2}$$

where $f_{g_{R_k}}$ is the feature function determined by g_{R_k}, $\lambda_{g_{R_k}}$ is a Lagrange multiplier [1] and Z is a normalization constant. We will not elaborate on the details of Equation (2) as they are not important for the rest of this work (see [8] for a detailed explanation). What is important to note is that according to (2), one optimization parameter $\lambda_{g_{R_k}}$ has to be determined for *each single ground instance* g_{R_k} of each conditional R_k. This readily yields a computationally infeasible optimization problem for larger knowledge bases because there might be just too many ground instances.

However, there are FO-PCL knowledge bases for which the ground instances of a conditional share the same entropy-optimal parameter. Parametric uniformity [8] means that for each conditional all its ground instances share the same entropy-optimal parameter value. The advantage of parametric uniformity is that just *one* optimization parameter λ_{R_k} per conditional R_k has to be computed instead of one parameter per ground instance :

$$p^*_{X(\mathcal{R})}(x) = \frac{1}{Z} \exp\left(\sum_{k=1}^{m} \lambda_{R_k} \sum_{g_{R_k} \in gnd(R_k)} f_{g_{R_k}}(x) \right) \tag{3}$$

Whereas parametric uniformity is a semantic notion, in [8] a syntactic criterion using so-called *probabilistic constraint involutions* is presented which is sufficient to ensure parametric uniformity. This syntactic criterion is based on the observation that parametric uniformity indicates identical knowledge about all ground instances of the same conditional for an FO-PCL knowledge base $\mathcal{R}$. Due to this, one should be able to transpose two ground instances g_{R_k}, g'_{R_k} of a conditional in $\mathcal{R}$ without changing the joint probability function with maximum entropy. In this case the transposed ground instances must possess the same entropy optimal parameter, as the Gibbs distribution in (2) is determined by a unique set of Lagrange multipliers.

Example 2. We continue Example 1 where likes takes one argument of sort $\mathit{Elephant}$ and one of sort Keeper. If D contains the constants $\{\mathit{clyde}, \mathit{dumbo}, \mathit{nirvan}\}$ of sort $\mathit{Elephant}$ and the constants $\{\mathit{fred}, \mathit{paul}, \mathit{simon}\}$ of sort Keeper, the set of ground instances of the conditionals is:

$gr_{EK_{1\text{–}1}}: \langle(\mathit{likes}(\mathit{clyde}, \mathit{paul}))[0.9], \top\rangle$ $\quad gr_{EK_{1\text{–}6}}: \langle(\mathit{likes}(\mathit{nirvan}, \mathit{simon}))[0.9], \top\rangle$
$gr_{EK_{1\text{–}2}}: \langle(\mathit{likes}(\mathit{dumbo}, \mathit{paul}))[0.9], \top\rangle$ $\quad gr_{EK_{2\text{–}1}}: \langle(\mathit{likes}(\mathit{dumbo}, \mathit{fred}))[0.05], \top\rangle$
$gr_{EK_{1\text{–}3}}: \langle(\mathit{likes}(\mathit{nirvan}, \mathit{paul}))[0.9], \top\rangle$ $\quad gr_{EK_{2\text{–}2}}: \langle(\mathit{likes}(\mathit{nirvan}, \mathit{fred}))[0.05], \top\rangle$
$gr_{EK_{1\text{–}4}}: \langle(\mathit{likes}(\mathit{clyde}, \mathit{simon}))[0.9], \top\rangle$ $\quad gr_{EK_{3\text{–}1}}: \langle(\mathit{likes}(\mathit{clyde}, \mathit{fred}))[1.0], \top\rangle$
$gr_{EK_{1\text{–}5}}: \langle(\mathit{likes}(\mathit{dumbo}, \mathit{simon}))[0.9], \top\rangle$

A possible probabilistic constraint involution transposes $gr_{EK_{1-1}}$ and $gr_{EK_{1-2}}$, and another one transposes $gr_{EK_{1-1}}$ and $gr_{EK_{1-4}}$.

An *involution covering* for $\mathcal{R}$ is a set Π of probabilistic constraint involutions such that for any two instances $g_{R_k}, g'_{R_k} \in gnd(R_k)$ with with $R_k \in \mathcal{R}$, there exists a sequence of involutions from Π transforming g_{R_k} into g'_{R_k}. The importance of involution coverings is given by Corollary 7.4.4 in [8], stating:

Proposition 1 (Involution covering implies parametric uniformity). *If there is an involution covering for $\mathcal{R}$, then $\mathcal{R}$ is parametrically uniform.*

For Example 2, it is particularly easy to find an involution covering as each ground instance of a conditional uses only one ground atom, and each ground atom appears in only one ground instance. In the next example, we present a simple knowledge base that is not parametrically uniform.

Example 3 (Misanthrope). The knowledge base $\mathcal{R}_{MI}$, adapted from [8], models friendship relations within a group of people, with one exceptional member, a misanthrope. In general, if a person V likes another person U, then it is very likely that U likes V, too. But there is one person, the misanthrope, who generally does not like other people:
$MI_1 : \langle(\mathit{likes}(U, V)|\mathit{likes}(V, U))[0.9], U \neq V\rangle$
$MI_2 : \langle(\mathit{likes}(a, V))[0.05], V \neq a\rangle$
The ground instances for the set of constants $D = \{a, b, c\}$ are:

$gr_{MI_{1\text{–}1}}: \langle(\mathit{likes}(a, b)|\mathit{likes}(b, a))[0.9], \top\rangle$ $\quad gr_{MI_{1\text{–}5}}: \langle(\mathit{likes}(c, a)|\mathit{likes}(a, c))[0.9], \top\rangle$
$gr_{MI_{1\text{–}2}}: \langle(\mathit{likes}(a, c)|\mathit{likes}(c, a))[0.9], \top\rangle$ $\quad gr_{MI_{1\text{–}6}}: \langle(\mathit{likes}(c, b)|\mathit{likes}(b, c))[0.9], \top\rangle$
$gr_{MI_{1\text{–}3}}: \langle(\mathit{likes}(b, a)|\mathit{likes}(a, b))[0.9], \top\rangle$ $\quad gr_{MI_{2\text{–}1}}: \langle(\mathit{likes}(a, b))[0.05], \top\rangle$
$gr_{MI_{1\text{–}4}}: \langle(\mathit{likes}(b, c)|\mathit{likes}(c, b))[0.9], \top\rangle$ $\quad gr_{MI_{2\text{–}2}}: \langle(\mathit{likes}(a, c))[0.05], \top\rangle$

There is no probabilistic constraint involution that transposes $gr_{MI_{1\text{-}1}}$ with $gr_{MI_{1\text{-}3}}$, the problem being that ground instances of *likes* are shared by MI_1 and MI_2 in an imbalanced way.

Luckily, each knowledge base $\mathcal{R}$ that is not parametrically uniform can be transformed into an equivalent $\mathcal{R}$' that is parametrically uniform. This is achieved by the set of transformations rules $\mathcal{PU}$ developed in [17].

Proposition 2 ([17]). *Exhaustively applying $\mathcal{PU}$ to a knowledge base $\mathcal{R}$ yields a knowledge base $\mathcal{PU}(\mathcal{R})$ such that $\mathcal{R}$ and $\mathcal{PU}(\mathcal{R})$ have the same maximum-entropy model and $\mathcal{PU}(\mathcal{R})$ is parametrically uniform.*

Example 4. By applying $\mathcal{PU}$ to $\mathcal{R}_{MI}$ from Ex. 3, the conditional MI_1 is replaced by

$$\begin{aligned}
MI_{1'} &: \langle(\mathit{likes}(U,V)|\mathit{likes}(V,U))[0.9], U \neq V \wedge U \neq a \wedge V \neq a\rangle \\
MI_{1''} &: \langle(\mathit{likes}(a,V)|\mathit{likes}(V,a))[0.9], V \neq a\rangle \\
MI_{1'''} &: \langle(\mathit{likes}(U,a)|\mathit{likes}(a,U))[0.9], U \neq a\rangle
\end{aligned}$$

and the resulting set $\mathcal{R}_{MIpu} := \mathcal{PU}(\mathcal{R}_{MI}) = \{MI_{1'}, MI_{1''}, MI_{1'''}, MI_2\}$ is parametrically uniform and has the same maximum entropy model as $\mathcal{R}_{MI}$.

3 Feature Functions and Parametric Uniformity

The argumentation used in Sec. 2 when comparing (2) and (3) implies that computing the maximum-entropy model for $\mathcal{R}$ via $\mathcal{PU}(\mathcal{R})$ reduces the number of optimization parameters that have to be determined. For the rest of this paper, $\mathcal{R} = \{R_1, \ldots, R_m\}$ will always denote an FO-PCL knowledge base with m conditionals, and in this section, we will show how parametric uniformity of $\mathcal{R}$ can be exploited when computing the maximum entropy model $p^*_{X(\mathcal{R})}$ from (1).

3.1 Defining Satisfaction via Feature Functions

Let G_{R_k} be the number of admissible ground instances of a conditional $R_k \in \mathcal{R}$, so we can enumerate the ground instances in $gnd(R_k)$ by defining $gnd(R_k) =: \{g^{(1)}_{R_k}, \ldots, g^{(G_{R_k})}_{R_k}\}$. For $R_k \in \mathcal{R}$ and every $g^{(i)}_{R_k} = \langle(\theta_{g^{(i)}_{R_k}}(\phi_{R_k}) \mid \theta_{g^{(i)}_{R_k}}(\psi_{R_k}))[\xi_{R_k}], \top\rangle \in gnd(R_k)$ define a *feature function* $f_{g^{(i)}_{R_k}}$ with

$$f_{g^{(i)}_{R_k}}(x) := \begin{cases} 1 & \mathit{iff}\ x \models \left(\theta_{g^{(i)}_{R_k}}(\phi_{R_k}) \wedge \theta_{g^{(i)}_{R_k}}(\psi_{R_k})\right), \\ 0 & \mathit{iff}\ x \models \left(\neg\theta_{g^{(i)}_{R_k}}(\phi_{R_k}) \wedge \theta_{g^{(i)}_{R_k}}(\psi_{R_k})\right), \\ \xi_{R_k} & \mathit{iff}\ x \models \left(\neg\theta_{g^{(i)}_{R_k}}(\psi_{R_k})\right) \end{cases} \tag{4}$$

The sum $\sum_{x \in X(\mathcal{R})} f_{g^{(i)}_{R_k}}(x) \cdot p_{X(\mathcal{R})}(x)$ is called the *expected value* of the feature function $f_{g^{(i)}_{R_k}}$ with respect to the distribution $p_{X(\mathcal{R})}$. In [8] it is shown that the satisfaction relation $p_{X(\mathcal{R})} \models \mathcal{R}$ can be expressed by using the notion of feature functions:

Proposition 3. *A probability distribution $p_{X(\mathcal{R})}$ is a model of $\mathcal{R}$ iff for all $R_k \in \mathcal{R}$:*

$$\begin{aligned} \sum_{x \in X(\mathcal{R})} f_{g^{(1)}_{R_k}}(x) \cdot p_{X(\mathcal{R})}(x) &= \xi_{R_k} \\ &\vdots \\ \sum_{x \in X(\mathcal{R})} f_{g^{(G_{R_k})}_{R_k}}(x) \cdot p_{X(\mathcal{R})}(x) &= \xi_{R_k} \end{aligned} \qquad (EQ^{orig}_{R_k})$$

For each conditional $R_k \in \mathcal{R}$, there are G_{R_k} feature functions $f_{g^{(1)}_{R_k}}(x), \ldots, f_{g^{(G_{R_k})}_{R_k}}(x)$, leading to G_{R_k} linear constraints which have to be satisfied by a distribution $p_{X(\mathcal{R})}$. Each such linear constraint enforces the expected value of the respective feature function. Note that all feature functions emerging from the ground instances of a conditional R_k must have the same expected value ξ_{R_k}. The set of solutions to this equation system is denoted by

$$Mod^{orig}_{R_k} := \{p_{X(\mathcal{R})} \mid p_{X(\mathcal{R})} \models EQ^{orig}_{R_k}\}$$

Let

$$\bigcup_{R_k \in \mathcal{R}} EQ^{orig}_{R_k} \qquad (EQ^{orig}_{\mathcal{R}})$$

denote the equation system consisting of all the linear constraints emerging from all admissible ground instances of $\mathcal{R}$. Thus

$$Mod^{orig}_{\mathcal{R}} := \{p_{X(\mathcal{R})} \mid p_{X(\mathcal{R})} \models EQ^{orig}_{\mathcal{R}}\} = \bigcap_{R_k \in \mathcal{R}} Mod^{orig}_{R_k}$$

denotes the set of probability distributions which are models of $\mathcal{R}$, i. e. which are solutions to the equation system $EQ^{orig}_{\mathcal{R}}$. Let

$$p^*_{orig} := \arg \max_{p_{X(\mathcal{R})} \in Mod^{orig}_{\mathcal{R}}} H(p_{X(\mathcal{R})})$$

denote the maximum entropy distribution regarding $Mod^{orig}_{\mathcal{R}}$. From Proposition 3 it follows directly that $p^*_{X(\mathcal{R})} = p^*_{orig}$ (cf. (1)).

3.2 Relaxing the Expected Value of Feature Functions

The following equation system is a relaxation of $EQ^{orig}_{R_k}$ since it introduces additional variables $c^{(i)}_{R_k} \in \mathbb{R}$, $1 \leq i \leq G_{R_k}$, which allow each expected value of $f^{(i)}_{g_{R_k}}$ to differ from ξ_{R_k} by some amount $c^{(i)}_{R_k}$. An additional linear constraint assures that the sum of all these differences remains 0:

$$\begin{aligned} \sum_{x \in X(\mathcal{R})} f_{g^{(1)}_{R_k}}(x) \cdot p_{X(\mathcal{R})}(x) &= \xi_{R_k} + c^{(1)}_{R_k} \\ &\vdots \\ \sum_{x \in X(\mathcal{R})} f_{g^{(G_{R_k})}_{R_k}}(x) \cdot p_{X(\mathcal{R})}(x) &= \xi_{R_k} + c^{(G_{R_k})}_{R_k} \\ \textstyle\sum_{i=1}^{G_{R_k}} c^{(i)}_{R_k} &= 0 \end{aligned} \qquad (EQ^{relax}_{R_k})$$

Using these equations, we define the two equation systems

$$\bigcup_{R_k \in \mathcal{R}} EQ_{R_k}^{relax} \qquad (EQ_{\mathcal{R}}^{relax})$$

$$(EQ_{\mathcal{R}}^{orig} \setminus EQ_{R_k}^{orig}) \cup EQ_{R_k}^{relax} \qquad (EQ_{\mathcal{R}}^{relax(R_k)})$$

where the last equation system originates from $EQ_{\mathcal{R}}^{orig}$ by just relaxing the equations regarding R_k. Let $Mod_{\mathcal{R}}^{relax}$ and $Mod_{\mathcal{R}}^{relax(R_k)}$, respectively, denote their set of solutions. Since $EQ_{\mathcal{R}}^{relax}$ is a relaxation of $EQ_{\mathcal{R}}^{orig}$ and $EQ_{\mathcal{R}}^{relax(R_k)}$ is a relaxation of $EQ_{\mathcal{R}}^{orig}$ for every $R_k \in \mathcal{R}$, the following proposition holds:

Proposition 4. $\quad Mod_{\mathcal{R}}^{orig} \subseteq Mod_{\mathcal{R}}^{relax(R_k)} \subseteq Mod_{\mathcal{R}}^{relax}$

The set $Mod_{\mathcal{R}}^{diff(R_k)} := Mod_{\mathcal{R}}^{relax(R_k)} \setminus Mod_{\mathcal{R}}^{orig}$ denotes the set of all those solutions of $EQ_{\mathcal{R}}^{relax(R_k)}$ for which it holds that the expected value of at least one feature function of R_k does not match ξ_{R_k}. In the following, we will show that the maximum entropy distribution regarding $Mod_{\mathcal{R}}^{relax(R_k)}$, namely

$$p^*_{relax(R_k)} := \arg \max_{p_{X(\mathcal{R})} \in Mod_{\mathcal{R}}^{relax(R_k)}} H(p_{X(\mathcal{R})}), \tag{5}$$

does not belong to $Mod_{\mathcal{R}}^{diff(R_k)}$. We start with the observation that if $\mathcal{R}$ is parametrically uniform, the ground instances of a conditional differ only in generic constants:

Proposition 5. *Let i be an argument position in a literal P in a conditional $R \in \mathcal{R}$, and let c_1, c_2 be the constants in that position i in P in two different ground instances of R. If $\mathcal{R}$ is parametrically uniform, then $c_1 = c_2$, or $c_1, c_2 \in D \setminus const(\mathcal{R})$.*

Note that in Ex. 3, the first argument in the head literal $likes(a, b)$ of $MI_{1\text{–}2}$ is the specific constant a, while the corresponding argument b in $MI_{1\text{–}3}$ is a generic constant. Thus, according to Prop. 5, this is only possible since $\mathcal{R}_{MI}$ is not parametrically uniform.

Example 5. Consider the conditional $MI_{1''}$ from the parametrically unifrom $\mathcal{R}_{MIpu}$ in Ex. 4. Using the notation from above, let $EQ_{\mathcal{R}_{MIpu}}^{relax(MI_{1''})}$ be the equation system which contains relaxed constraints for $MI_{1''}$. Let $p^*_{relax(MI_{1''})}$ be the ME-distribution regarding $EQ_{\mathcal{R}_{MIpu}}^{relax(MI_{1''})}$. For the set $D = \{a, b, c\}$ of constants, $MI_{1''}$ has the two ground conditionals $g_{MI^1_{1''}} = (likes(a, b)|likes(b, a))$ and $g_{MI^2_{1''}} = (likes(a, c)|likes(c, a))$. Since $g_{MI^1_{1''}}$ and $g_{MI^2_{1''}}$ differ only in the generic constants b and c for which $EQ_{\mathcal{R}_{MIpu}}^{relax(MI_{1''})}$ contains equivalent constraints, the maximum entropy principle ensures that these ground conditionals have the same probability under $p^*_{relax(MI_{1''})}$.

In general, assuming $p^*_{relax(R_k)} \neq p^*_{orig}$ implies $p^*_{relax(R_k)} \in Mod_{\mathcal{R}}^{diff(R_k)}$ and that there are two ground instances $g_{R_k}^{(i)}, g_{R_k}^{(j)} \in gnd(R_k)$ for which $c^{(i)} \neq c^{(j)}$ and therefore

$$p^*_{relax(R_k)}(g_{R_k}^{(i)}) = \xi_{R_k} + c_{R_k}^{(i)} \quad \neq \quad \xi_{R_k} + c_{R_k}^{(j)} = p^*_{relax(R_k)}(g_{R_k}^{(j)})$$

holds. Thus, the two ground instances $g^{(i)}_{R_k}$ and $g^{(j)}_{R_k}$ would have different probabilities under the maximum entropy distribution $p^*_{relax(R_k)}$ which is not possible if $\mathcal{R}$ is parametrically uniform. Thus, $p^*_{relax(R_k)} \notin Mod^{diff(R_k)}_{\mathcal{R}}$ must hold, yielding:

Proposition 6. *If $\mathcal{R}$ is parametrically uniform, then for $R_k \in \mathcal{R}$ it holds:*

$$p^*_{relax(R_k)} = p^*_{orig} \tag{6}$$

Two equation systems EQ_1 and EQ_2 are called *ME-equivalent* iff

$$\arg \max_{p_{X(\mathcal{R})} \in Mod_1} H(p_{X(\mathcal{R})}) = \arg \max_{p_{X(\mathcal{R})} \in Mod_2} H(p_{X(\mathcal{R})})$$

holds, with Mod_1, Mod_2 being the set of solutions to the respective equation system. From Proposition 6 we directly get:

Proposition 7. *If $\mathcal{R}$ is parametrically uniform, then for $R_k \in \mathcal{R}$, the equation systems $EQ^{orig}_{\mathcal{R}}$ and $EQ^{relax(R_k)}_{\mathcal{R}}$ are ME-equivalent.*

Therefore, $EQ^{orig}_{\mathcal{R}}$ can ME-equivalently be replaced by $EQ^{relax(R_k)}_{\mathcal{R}}$ since it yields the same maximum-entropy distribution. By applying such replacements successively for all conditionals $R_1, \ldots, R_m$, the original equation system $EQ^{orig}_{\mathcal{R}}$ can finally be replaced by $EQ^{relax}_{\mathcal{R}}$.

Proposition 8. *If $\mathcal{R}$ is parametrically uniform, then the equation systems $EQ^{orig}_{\mathcal{R}}$ and $EQ^{relax}_{\mathcal{R}}$ are ME-equivalent and therefore $p^*_{orig} = p^*_{relax}$ holds, where*

$$p^*_{relax} := \arg \max_{p_{X(\mathcal{R})} \in Mod^{relax}_{\mathcal{R}}} H(p_{X(\mathcal{R})})$$

3.3 Reducing the Number of Linear Constraints

For this subsection, we generally assume that $\mathcal{R}$ is parametrically uniform. The next proposition allows us, roughly speaking, to sum up the equations from $EQ^{relax}_{R_k}$ to just one equation, yielding for $R_k \in \mathcal{R}$:

$$\sum_{x \in X(\mathcal{R})} p_{X(\mathcal{R})}(x) \sum_{i=1}^{G_{R_k}} f_{g^{(i)}_{R_k}}(x) = G_{R_k} \cdot \xi_{R_k} \tag{$EQ^{sum}_{R_k}$}$$

Let $Mod^{sum}_{R_k}$ denote the solutions of $EQ^{sum}_{R_k}$. Then we have:

Proposition 9. $\quad Mod^{relax}_{R_k} = Mod^{sum}_{R_k}$

Thus, the equations systems $EQ^{relax}_{R_k}$ and $EQ^{sum}_{R_k}$ are equivalent, since they have the same set of solutions. Analogously to before, we define the equation system

$$\bigcup_{R_k \in \mathcal{R}} EQ^{sum}_{R_k} \tag{$EQ^{sum}_{\mathcal{R}}$}$$

and we let $Mod^{sum}_{\mathcal{R}}$ denote its set of solutions, yielding:

Proposition 10. $Mod_{\mathcal{R}}^{relax} = Mod_{\mathcal{R}}^{sum}$

Therefore, using Propositions 8 and 10 we get:

Proposition 11. *The equation systems* $EQ_{\mathcal{R}}^{orig}$ *and* $EQ_{\mathcal{R}}^{sum}$ *are ME-equivalent and therefore* $p^*_{orig} = p^*_{sum}$ *holds, where*

$$p^*_{sum} := \arg \max_{p_{X(\mathcal{R})} \in Mod_{\mathcal{R}}^{sum}} H(p_{X(\mathcal{R})})$$

The following proposition puts together the results obtained so far:

Proposition 12.

$$\begin{array}{ccccc} Mod_{\mathcal{R}}^{orig} & \subseteq & Mod_{\mathcal{R}}^{relax} & = & Mod_{\mathcal{R}}^{sum} \\ p^*_{orig} & = & p^*_{relax} & = & p^*_{sum} \end{array}$$

4 Simplification of Linear Constraints in ME-Computation

The equation system $EQ_{\mathcal{R}}^{orig}$ consists of $G_{\mathcal{R}} := \sum_{k=1}^{m} G_{R_k}$ linear constraints, i. e. one linear constraint for each ground instance of each conditional. Since the number of ground instances grows very fast in the number of constants in D, $EQ_{\mathcal{R}}^{orig}$ consists of a large number of linear constraints which all have to be respected when computing the ME-distribution p^*_{orig}. Proposition 12 allows us to replace the original equation system $EQ_{\mathcal{R}}^{orig}$ with the much smaller equation system $EQ_{\mathcal{R}}^{sum}$ containing just m linear constraints, i. e. exactly one linear constraint for each conditional. Therefore, computing the ME-distribution with respect to the solutions of $EQ_{\mathcal{R}}^{sum}$ is much easier, since only m linear constraints have to be respected.

Example 6. Consider the parametrically uniform knowledge base $\mathcal{R}_{pq}$ with

$$\begin{array}{l} R_1 : \langle (P(U)|Q(Z,Z))[\xi], U \neq Z \rangle \\ R_2 : \langle (P(U)|Q(V,Z))[\xi], U \neq V \wedge U \neq Z \wedge V \neq Z \rangle \end{array}$$

together with a set of 10 constants. Then there exist $G_{R_1} = 90$ admissible ground instances of R_1 and $G_{R_2} = 720$ instances of R_2. Therefore $G_{\mathcal{R}} = G_{R_1} + G_{R_1} = 810$ linear constraints have to be respected when computing the ME-distribution p^*_{orig} regarding the equation system $EQ_{\mathcal{R}}^{orig}$. However, by alternatively computing the ME-distribution p^*_{sum} (which is identical to p^*_{orig}) of the equation system $EQ_{\mathcal{R}}^{sum}$, merely 2 linear constraints have to be respected.

A closer look at an equation $EQ_{R_k}^{sum}$ from $EQ_{\mathcal{R}}^{sum}$ reveals that the distinct values of the individual feature functions $f_{g_{R_k}^{(1)}}(x), \ldots, f_{g_{R_k}^{(G_{R_k})}}(x)$ are not relevant insofar as only their sum $\sum_{i=1}^{G_{R_k}} f_{g_{R_k}^{(i)}}(x)$ is used as a factor in the equation. Therefore, we can define a new feature function F_{R_k} for each $R_k \in \mathcal{R}$ which encapsulates $f_{g_{R_k}^{(1)}}, \ldots, f_{g_{R_k}^{(G_{R_k})}}$ by

$$F_{R_k}(x) := \sum_{i=1}^{G_{R_k}} f_{g_{R_k}^{(i)}}(x) \tag{7}$$

According to the definition of $f_{g_{R_k}^{(i)}}(x)$ in (4), it holds $f_{g_{R_k}^{(i)}}(x) = 1$ if the ground conditional $g_{R_k}^{(i)}$ is verified by x, or $f_{g_{R_k}^{(i)}}(x) = 0$ if $g_{R_k}^{(i)}$ is falsified by x; otherwise $f_{g_{R_k}^{(i)}}(x) = \xi_{R_k}$ holds, i. e. $g_{R_k}^{(i)}$ is not applicable to x. The *counting functions* given by

$$
\begin{aligned}
\mathit{ver}_{R_k}(x) &:= \left|\left\{ g_{R_k}^{(i)} \in \mathit{gnd}(R_k) \mid x \models (\theta_{g_{R_k}^{(i)}}(\phi_{R_k}) \wedge \theta_{g_{R_k}^{(i)}}(\psi_{R_k})) \right\}\right| \\
\mathit{napp}_{R_k}(x) &:= \left|\left\{ g_{R_k}^{(i)} \in \mathit{gnd}(R_k) \mid x \models (\neg\theta_{g_{R_k}^{(i)}}(\psi_{R_k})) \right\}\right|
\end{aligned}
$$

represent the number of ground instances of R_k which are verified by a worlds x or which are not applicable to x, respectively. Using these counting functions, we can now state F_{R_k} without the notion of feature functions $f_{g_{R_k}^{(i)}}(x)$ as:

$$F_{R_k}(x) = \mathit{ver}_{R_k}(x) + \mathit{napp}_{R_k}(x) \cdot \xi_{R_k} \tag{8}$$

Using (8), we define the following equation system consisting of m linear constraints:

$$\sum_{x \in X(\mathcal{R})} p_{X(\mathcal{R})}(x) F_{R_k}(x) = G_{R_k} \cdot \xi_{R_k}, \; 1 \le k \le m \qquad (EQ_{\mathcal{R}}^{sumF})$$

Proposition 13. *If $\mathcal{R}$ is a parametrically uniform, then the maximum entropy model $p^*_{X(\mathcal{R})}$ of $\mathcal{R}$ under FO-PCL semantics is the solution of the optimization problem*

$$p^*_{X(\mathcal{R})} = \arg \max_{p_{X(\mathcal{R})} \in Mod_{\mathcal{R}}^{sumF}} H(p_{X(\mathcal{R})})$$

with $Mod_{\mathcal{R}}^{sumF}$ being the solutions of $EQ_{\mathcal{R}}^{sumF}$.

Using the method of Lagrange multipliers [1], it can be shown that there exist m unique Lagrange parameters $\lambda_1, \ldots, \lambda_m$ so that $p^*_{X(\mathcal{R})}$ can be represented by the Gibbs distribution

$$p^*_{X(\mathcal{R})}(x) = \frac{1}{Z} \exp\left(\sum_{k=1}^{m} \lambda_k F_{R_k}(x) \right) \tag{9}$$

with Z being a normalization constant. By defining $\alpha_k := \exp(\lambda_k)$, the distribution can also be expressed as the product

$$p^*_{X(\mathcal{R})}(x) = \frac{1}{Z} \prod_{k=1}^{m} \alpha_k^{F_{R_k}(x)} \tag{10}$$

The representation in (9) matches the result from [8], stating that for an parametrically uniform set $\mathcal{R}$ a representation as Gibbs distribution with just one Lagrange parameter λ_k per conditional $R_k \in \mathcal{R}$ must exist and that therefore just m optimization parameters have to be determined. However, the determination of these m parameters would still require to respect $\sum_{k=1}^{m} G_{R_k}$ linear constraints. Our results developed above exploit the parametric uniformity of $\mathcal{R}$ by showing that it is sufficient to solve an alternative optimization problem with just m linear constraints to determine $p^*_{X(\mathcal{R})}$.

Table 1. Results for computing the ME-distribution with a GIS algorithm approach

Knowledge	Size of		Linear Constraints in		Iteration	Computation Time	
Base	D	Ω	$EQ_{\mathcal{R}}^{orig}$	$EQ_{\mathcal{R}}^{sumF}$	Steps	$EQ_{\mathcal{R}}^{orig}$	$EQ_{\mathcal{R}}^{sumF}$
$\mathcal{R}_{CC}$	3	2^{15}	12	3	37,356	26 sec	2 sec
$\mathcal{R}_{EK}$	7	2^{12}	12	3	1,028	4 sec	<1 sec
$\mathcal{R}_{EK}$	8	2^{16}	16	3	1,275	147 sec	2 sec
$\mathcal{R}_{MIpu}$	3	2^{9}	8	4	14,343	2 sec	<1 sec
$\mathcal{R}_{MIpu}$	4	2^{16}	15	4	25,670	159 sec	6 sec

5 Implementation and First Evaluation Results

Within the framework of the KREATOR system [7], we implemented the computation of the maximum entropy distribution for FO-PCL knowledge bases using the *Generalized Iterative Scaling* (GIS) approach from [2]; the concrete algorithm is a variant of the algorithm described in [5]. Since the optimization problem to be solved is convex, general algorithm techniques for this class of problems could be applied [1], but the GIS algorithm technique computes the ME-distribution under linear constraints, making this algorithm tailored for our problem at hand. GIS iteratively computes a sequence of distributions which converges to the ME-distribution. In every iteration step, each linear constraint has to be considered, therefore the number of linear constraints has a significant impact on the computation speed of the algorithm.

For illustrating the practical computational benefits of solving the ME-optimization problem with respect to the equation system $EQ_{\mathcal{R}}^{sumF}$ compared to the equation system $EQ_{\mathcal{R}}^{orig}$, we consider $\mathcal{R}_{EK}$ and $\mathcal{R}_{MIpu}$ from Examples 1 and 4, respectively, as well as the set $\mathcal{R}_{CC}$ from the following example.

Example 7 (Common Cold). The common cold example from [8] models the probability of catching a common cold, depending on a persons's general susceptibility and his contacts within a group of people, with the set $\mathcal{R}_{CC}$ containing the conditionals:

$$CC_1 : \langle (commoncold(U))[0.01], \top \rangle$$
$$CC_2 : \langle (commoncold(U) | susceptible(U))[0.1], \top \rangle$$
$$CC_3 : \langle (commoncold(U) | contact(U, V) \wedge commoncold(V))[0.6], U \neq V \rangle$$

Since $\mathcal{R}_{CC}$, $\mathcal{R}_{EK}$, and $\mathcal{R}_{MIpu}$ are all parametrically uniform, according to Proposition 13 it is sufficient to solve the corresponding optimization problem with respect to the equation system $EQ_{\mathcal{R}}^{sumF}$, i. e., an equation system containing only one linear constraint for each conditional in the knowledge base.

The results in Table 1 show that the computation time with respect to $EQ_{\mathcal{R}}^{orig}$ is several times higher compared to $EQ_{\mathcal{R}}^{sumF}$. The number of linear constraints in $EQ_{\mathcal{R}}^{orig}$ corresponds to the number of groundings of all conditionals and thereby depends on the number of constants in D, whereas the number of linear constraints in $EQ_{\mathcal{R}}^{sumF}$ is given by the number of conditionals in $\mathcal{R}$, i. e. it is independent of the size of D. A comparison of the computation times makes clear that an increased number of linear constraints causes a severe increase in computation time; e. g. considering the set $\mathcal{R}_{EK}$ together with 8 constants, the 16 linear constraints of $EQ_{\mathcal{R}_{EK}}^{orig}$ cause a computation of 147 seconds, whereas considering only the 3 linear constraints of $EQ_{\mathcal{R}_{EK}}^{sumF}$ allows to compute the solution within 2 seconds. Since the number of linear constraints in $EQ^{sumF}_{\mathcal{R}}$ is fixed, the positive effects compared to $EQ^{orig}_{\mathcal{R}}$ become even more evident when the number of constants is increased.

As an illustration of the benefits of transforming an FO-PCL knowledge base $\mathcal{R}$ into a parametrically uniform one as given by $\mathcal{PU}(\mathcal{R})$, reconsider $\mathcal{R}_{MI}$ (Ex. 3). $\mathcal{R}_{MI}$ has the same maximum entropy distribution as $\mathcal{R}_{MIpu}$, but unlike $\mathcal{R}_{MIpu}$, it is not parametrically uniform. Thus, when computing the ME distribution on the basis of $\mathcal{R}_{MI}$, we can not use the optimized equation system $EQ^{sumF}_{\mathcal{R}_{MI}}$, but we must use the original $EQ^{orig}_{\mathcal{R}_{MI}}$ (which is identical to $EQ^{orig}_{\mathcal{R}_{MIpu}}$), requiring 159 seconds in the presence of 4 constants (cf. the last row in Table 1), as opposed to just 6 seconds when using $\mathcal{R}_{MIpu}$ and exploiting the parametric uniformity according to $EQ^{sumF}_{\mathcal{R}_{MIpu}}$.

6 Conclusions and Further Work

After briefly summarizing the basics of FO-PCL and its maximum entropy model, we showed that the determination of the ME model for a set of FO-PCL conditionals $\mathcal{R}$ requires to solve an optimization problem under, in general, a large number of linear constraints even if $\mathcal{R}$ is parametrically uniform. We developed an approach how the property of parametric uniformity can be exploited to reduce the number of linear constraints of the ME optimization problem significantly: Just one linear constraint per conditional has to be considered, independent of the number of available constants, whereas the number of linear constraints in the original equation system grows rapidly in the number of constants of the knowledge base. Our first example applications show promising results, but further experimental and theoretical investigations are needed.

The focus of our paper was on the reduction of the number of linear constraints in the FO-PCL maximum entropy optimization problem, which has not been dealt with previously. Another source of complexity stems from the fact that the ME model of a knowledge base $\mathcal{R}$ with constant set D is a complete joint probability distribution over all ground atoms in $\mathcal{H}(\mathcal{R})$, i. e. the size of the distribution is exponential in the number of constants in D. Hence, techniques of lifted inference [20,3,12,18] should be applied to FO-PCL to reduce the exponential size of models. Towards this end, for *aggregating semantics* [16], we have already developed an approach exploiting the equivalences of worlds induced by generic constants [6], and we are currently transferring these results to FO-PCL. Aggregating and *averaging* semantics [16] are two alternative semantics for relational probabilistic conditionals using a syntax similar to the FO-PCL syntax, but without constraint formulas. In future work, we will also address the question how our results obtained in this paper can be used for ME inferencing for these alternative semantics.

References

1. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York (2004)
2. Darroch, J.N., Ratcliff, D.: Generalized iterative scaling for log-linear models. Annals of Mathematical Statistics 43(5), 1470–1480 (1972)

3. de Salvo Braz, R., Amir, E., Roth, D.: Lifted first-order probabilistic inference. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI 2005, pp. 1319–1325. Professional Book Center (2005)
4. Delgrande, J.: On first-order conditional logics. Artificial Intelligence 105, 105–137 (1998)
5. Finthammer, M.: An iterative scaling algorithm for maximum entropy reasoning in relational probabilistic conditional logic. In: Hüllermeier, E. (ed.) SUM 2012. LNCS, vol. 7520, pp. 351–364. Springer, Heidelberg (2012)
6. Finthammer, M., Beierle, C.: Using equivalences of worlds for aggregation semantics of relational conditionals. In: KI 2012: Proceedings of 35th Annual German Conference on Advances in Artificial Intelligence AI, Saarbrücken, Germany, September 24-27. LNCS (LNAI), Springer (to appear, 2012)
7. Finthammer, M., Thimm, M.: An integrated development environment for probabilistic relational reasoning. Logic Journal of the IGPL (to appear, 2012)
8. Fisseler, F.: Learning and Modeling with Probabilistic Conditional Logic. Dissertations in Artificial Intelligence, vol. 328. IOS Press (2010)
9. Fisseler, J.: First-order probabilistic conditional logic and maximum entropy. Logic Journal of the IGPL (to appear, 2012)
10. Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence 6, 721–741 (1984)
11. Halpern, J.: Reasoning About Uncertainty. MIT Press (2005)
12. Jaimovich, A., Meshi, O., Friedman, N.: Template based inference in symmetric relational markov random fields. In: Proc. of the 23rd Conference on Uncertainty in Artificial Intelligence. AUAI Press (2007)
13. Kern-Isberner, G.: Characterizing the principle of minimum cross-entropy within a conditional-logical framework. Artificial Intelligence 98, 169–208 (1998)
14. Kern-Isberner, G.: Conditionals in Nonmonotonic Reasoning and Belief Revision. LNCS (LNAI), vol. 2087. Springer, Heidelberg (2001)
15. Kern-Isberner, G., Lukasiewicz, T.: Combining probabilistic logic programming with the power of maximum entropy. Artificial Intelligence, Special Issue on Nonmonotonic Reasoning 157(1-2), 139–202 (2004)
16. Kern-Isberner, G., Thimm, M.: Novel semantical approaches to relational probabilistic conditionals. In: Lin, F., Sattler, U., Truszczyński, M. (eds.) Proc. of the 12th Int'l. Conference on the Principles of Knowledge Representation and Reasoning (KR 2010), May 2010, pp. 382–392. AAAI Press (May 2010)
17. Krämer, A., Beierle, C.: On Lifted Inference for a Relational Probabilistic Conditional Logic with Maximum Entropy Semantics. In: Lukasiewicz, T., Sali, A. (eds.) FoIKS 2012. LNCS, vol. 7153, pp. 224–243. Springer, Heidelberg (2012)
18. Milch, B., Zettlemoyer, L., Kersting, K., Haimes, M., Kaelbling, L.P.: Lifted probabilistic inference with counting formulas. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17. AAAI Press (2008)
19. Paris, J.: The uncertain reasoner's companion – A mathematical perspective. Cambridge University Press (1994)
20. Poole, D.: First-order probabilistic inference. In: Gottlob, G., Walsh, T. (eds.) Proc. IJCAI 2003, pp. 985–991. Morgan Kaufmann (2003)

Exact Query Reformulation with First-Order Ontologies and Databases

Enrico Franconi, Volha Kerhet, and Nhung Ngo

Free University of Bozen-Bolzano, Italy
lastname@inf.unibz.it

Abstract. We study a general framework for query rewriting in the presence of an arbitrary first-order logic ontology over a database signature. The framework supports deciding the existence of a safe-range first-order equivalent reformulation of a query in terms of the database signature, and if so, it provides an effective approach to construct the reformulation based on interpolation using standard theorem proving techniques (e.g., tableau). Since the reformulation is a safe-range formula, it is effectively executable as an SQL query. At the end, we present an application of the framework with $\mathcal{SHOQ}$ ontologies.

1 Introduction

We address the problem of query reformulation with expressive ontologies over databases. An ontology provides a conceptual view of the database and it is composed by constraints on a vocabulary extending the basic vocabulary of the data. Querying a database using the terms in such a richer ontology allows for more flexibility than using only the basic vocabulary of the relational database directly.

In this paper we study and develop a query rewriting framework applicable to knowledge representation systems where data is stored in a classical finite relational database, in a way that in the literature has been called *locally-closed world* assumption [1], *exact views* [2,3,4], or *DBox* [5,6]. A DBox is a set of ground atoms which semantically behaves like a database, i.e., the interpretation of the database predicates in the DBox is exactly equal to the database relations. The DBox predicates are *closed*, i.e., their extensions are the same in every interpretation, whereas the other predicates in the knowledge base are *open*, i.e., their extensions may vary among different interpretations. We do not consider here the *open* interpretation for the database predicates (also called *ABox* or *sound views*). In an ABox, the interpretation of database predicates contains the database relations and possibly more. This notion is less faithful in the representation of a database semantics since it would allow for spurious interpretations of database predicates with additional unwanted tuples not present in the original database.

In our general framework an ontology is a set of first-order formulas, and queries are (possibly open) first-order formulas. Within this setting, the framework provides support to decide the existence of a safe-range first-order equivalent reformulation of a query in terms of the database signature. It also provides an effective approach to construct the reformulation. We are interested in safe-range reformulations of queries because their range-restricted syntax is needed to reduce the original query answering problem to a relational algebra evaluation (e.g., via SQL) over the original database [7]. Our framework points out several conditions on the ontologies and the queries to guarantee the existence of a safe-range reformulation. We show that these conditions are not infeasible in practice and we also provide an efficient method to ensure their validation. Standard theorem proving techniques can be used to compute the reformulation.

In order to be complete, our framework is applicable to ontologies and queries expressed in any fragment of first-order logic enjoying finitely controllable determinacy [3], a stronger property than the finite model property of the logic. If the employed logic does not enjoy finitely controllable determinacy our approach would become sound but incomplete, by still effectively implementable using standard theorem proving techniques. We have explored non-trivial applications where the framework is complete; in this paper, the application with $\mathcal{SHOQ}$ ontology and concept queries is discussed. We show how (i) to check whether the answers to a given query with an ontology are *solely* determined by the extension of the DBox predicates and, if so, (ii) to find an equivalent rewriting of the query in terms of the DBox predicates to allow the use of standard database technology for answering the query. This means we benefit from the low computational complexity in the size of the data for answering queries on relational databases. In addition, it is possible to reuse standard techniques of description logics reasoning to find rewritings, such as in [5].

The query reformulation problem has received strong interest in classical relational database research as well as modern knowledge representation studies. Differently from the mainstream research on query reformulation [8], which is mostly based on perfect or maximally contained rewritings with sound views (see, e.g., the DL-Lite approach [9]), we focus here on exact rewritings with exact views, since it characterises more precisely the query answering problem with ontologies and databases, and it allows for very expressive ontology languages.

This work extends the works on exact rewritings with exact views [2,5,3] by focussing on *safe-range reformulations* and on the conditions ensuring their existence, and by considering general first-order ontologies extending the database signature, rather than just *local as view* constraints over the database predicates [8]. This paper extends to full FOL the results limited to description logics published in the paper [10].

The paper is organised as follows: Section 2 provides the necessary formal background and definitions; Section 3 introduces the notion of a query determined by a database; Section 4 introduces a characterisation of the query reformulation problem; in Section 5 the conditions allowing for an effective reformulation are analysed. At the end, we present an application with $\mathcal{SHOQ}$ ontologies.

2 Preliminaries

Let $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ be a classical function-free first-order language with equality over a signature $\Sigma = (\mathbb{C}, \mathbb{P})$, where $\mathbb{C}$ is a (possibly infinite) set of *constants* and $\mathbb{P}$ is a set of *predicates* with associated arities. We call $\mathcal{L}$ a fragment of $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$.

We denote as $\sigma(\varphi_1, \ldots, \varphi_n)$ the signature of the formulas $\{\varphi_1, \ldots, \varphi_n\}$, i.e., the union of all predicates and constants occurring in each formula φ_i $(1 \leq i \leq n)$. Given a formula φ, we denote the set of all variables appearing in φ as $\textsc{var}(\varphi)$, and the set of the free variables appearing in φ as $\text{FREE}(\varphi)$; we may use for φ the notation $\varphi_{[\mathbb{X}]}$ – where $\mathbb{X} = \text{FREE}(\varphi)$ is the (possibly empty) set of free variables of the formula.

A *database (instance)* $\mathcal{DB}$ is a *finite* set of ground atoms of the form $P(c_1, \ldots, c_n)$, where $P \in \mathbb{P}$, n-ary predicate, and $c_i \in \mathbb{C}$ $(1 \leq i \leq n)$. The set of all predicates appearing in a database $\mathcal{DB}$ is denoted as $\mathbb{P}_{\mathcal{DB}}$, and the set of all constants appearing in $\mathcal{DB}$ is called the *active domain of* $\mathcal{DB}$, and is denoted as $\mathbb{C}_{\mathcal{DB}}$. A (possibly empty) finite set $\mathcal{KB}$ of closed formulas will be called an *ontology*.

As usual, an *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ includes a non-empty set – the domain $\Delta^{\mathcal{I}}$ – and an interpretation function $\cdot^{\mathcal{I}}$ defined over constants and predicates of the signature. An interpretation $\mathcal{I}$ *embeds a database* $\mathcal{DB}$, written $\mathcal{I}_{(\mathcal{DB})}$, if it holds that $a^{\mathcal{I}} = a$ for every database constant $a \in \mathbb{C}_{\mathcal{DB}}$ (the *standard name assumption (SNA)*, customary in databases, see [7]) and that $(c_1, \ldots, c_n) \in P^{\mathcal{I}}$ if and only if $P(c_1, \ldots, c_n) \in \mathcal{DB}$. In other words, in every interpretation embedding a $\mathcal{DB}$ the interpretation of any database predicate is always the same and it is given exactly by its content in the database; this is, in general, not the case for the interpretation of the non-database predicates. We say that all the database predicates are *closed*, while all the other predicates are *open* and may be interpreted differently in different interpretations. We do not consider here the *open world* assumption (the *ABox*) for embedding a database in an interpretation. In an open world, an interpretation $\mathcal{I}$ soundly embeds a database if it holds that $(c_1, \ldots, c_n) \in P^{\mathcal{I}}$ if (but *not* only if) $P(c_1, \ldots, c_n) \in \mathcal{DB}$.

In order to allow for an arbitrary database to be embedded, we generalise the standard name assumption to all the constants in $\mathbb{C}$; this implies that the domain of any interpretation necessarily includes the set of all the constants $\mathbb{C}$.

We denote an interpretation $\mathcal{I}$ with a specific domain Δ as $\mathcal{I}^{(\Delta)}$. Given an interpretation $\mathcal{I}$, we denote as $\mathcal{I}|_{\mathbb{S}}$ the interpretation restricted to the smaller signature $\mathbb{S} \subseteq \mathbb{P} \cup \mathbb{C}$, i.e., the interpretation with the same domain $\Delta^{\mathcal{I}}$ and the same interpretation function $\cdot^{\mathcal{I}}$ defined only for the constants and predicates from the set $\mathbb{S}$. The *semantic active domain* of a signature $\sigma' \subseteq \mathbb{P} \cup \mathbb{C}$ in an interpretation $\mathcal{I}$, denoted $adom(\sigma', \mathcal{I})$, is the set of all elements of the domain $\Delta^{\mathcal{I}}$ occurring in interpretations of predicates and constants from σ' in $\mathcal{I}$:

$$adom(\sigma', \mathcal{I}) := \bigcup_{P \in \sigma'} \bigcup_{(a_1, \ldots, a_n) \in P^{\mathcal{I}}} \{a_1, \ldots, a_n\} \cup \bigcup_{c \in \sigma'} c^{\mathcal{I}}.$$

If $\sigma' \subseteq \mathbb{P}_{\mathcal{DB}} \cup \mathbb{C}$, then for any $\mathcal{I}_{(\mathcal{DB})}$ and $\mathcal{J}_{(\mathcal{DB})}$, $adom(\sigma', \mathcal{I}_{(\mathcal{DB})}) = adom(\sigma', \mathcal{J}_{(\mathcal{DB})})$; so, for such case we introduce the notation $adom(\sigma', \mathcal{DB}) := adom(\sigma', \mathcal{I}_{(\mathcal{DB})})$, where $\mathcal{I}_{(\mathcal{DB})}$ is any interpretation embedding the database $\mathcal{DB}$. Intuitively $adom(\sigma', \mathcal{DB})$ includes the constants from σ' and from DB appearing in the relations corresponding to the predicates from σ'.

Let $\mathbb{X}$ be a set of variable symbols and $\mathbb{S}$ a set; we define a *substitution* $\Theta_{\mathbb{X}}^{\mathbb{S}}$ to be a total function $\mathbb{X} \mapsto \mathbb{S}$ assigning an element in $\mathbb{S}$ to each variable in $\mathbb{X}$, including the empty substitution ϵ when $\mathbb{X} = \emptyset$. The image of a substitution $\Theta_{\mathbb{X}}^{\mathbb{S}}$ is written as *act-range*$(\Theta_{\mathbb{X}}^{\mathbb{S}})$. Given a subset of the set of constants $\mathbb{C}' \subseteq \mathbb{C}$, we write that a formula $\varphi_{[\mathbb{X}]}$ is true in an interpretation $\mathcal{I}$ with its free variables substituted according to a substitution $\Theta_{\mathbb{X}}^{\mathbb{C}'}$ as $(\mathcal{I} \models \varphi_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}'}]})$. Given an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, a subset of its domain $\Delta \subseteq \Delta^{\mathcal{I}}$, we write that a formula $\varphi_{[\mathbb{X}]}$ is true in $\mathcal{I}$ with its free variables interpreted according to a substitution $\Theta_{\mathbb{X}}^{\Delta}$ as $(\mathcal{I}, \Theta_{\mathbb{X}}^{\Delta} \models \varphi)$. The *extension domain* of a formula $\varphi_{[\mathbb{X}]}$ with respect to the interpretation $\mathcal{I}$ is defined as the set of domain elements $\bigcup\{\textit{act-range}(\Theta^{\Delta}_{\mathbb{X}}) \mid \mathcal{I}, \Theta^{\Delta}_{\mathbb{X}} \models \varphi_{[\mathbb{X}]}\}$.

As usual, an interpretation in which a closed formula is true is called a *model* for the formula; the set of all models of a formula φ (resp. $\mathcal{KB}$) is denoted as $M(\varphi)$ (resp. $M(\mathcal{KB})$). A *database* $\mathcal{DB}$ *is legal for an ontology* $\mathcal{KB}$ if there exists a model of $\mathcal{KB}$ embedding $\mathcal{DB}$. In the following we will consider only consistent non-tautological ontologies and legal databases.

2.1 Queries

A *query* is a (possibly closed) formula. Given a query $\mathcal{Q}_{[\mathbb{X}]}$, we define its *certain answer* over $\mathcal{KB}$ and $\mathcal{DB}$ as follows:

Definition 1 (Certain Answer). *The* (certain) answer of a query $\mathcal{Q}_{[\mathbb{X}]}$ to a database $\mathcal{DB}$ under the ontology $\mathcal{KB}$ *is the set of substitutions with constants:*

$$\{\Theta^{\mathbb{C}}_{\mathbb{X}} \mid \forall \mathcal{I}_{(\mathcal{DB})} \in M(\mathcal{KB}) \ : \ \mathcal{I}_{(\mathcal{DB})} \models \mathcal{Q}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}\}.$$

Query answering is defined as an *entailment* problem, and as such it is going to have the same (high) complexity as entailment.

Note, that if a query $\mathcal{Q}$ is closed (i.e., a Boolean query), then the certain answer is $\{\epsilon\}$ if $\mathcal{Q}$ is true in all the models of the ontology embedding the database, and $\emptyset$ otherwise. In the following, we assume that – given a substitution $\Theta^{\mathbb{C}}_{\mathbb{X}}$ assigning to variables *distinct* constants not appearing in $\mathcal{Q}$, nor in $\mathcal{KB}$, nor in $\mathbb{C}_{\mathcal{DB}}$ – the closed formula $\mathcal{Q}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}$ is neither valid nor inconsistent under the ontology $\mathcal{KB}$: this would lead to trivial reformulations.

We now show that we can weaken the standard name assumption for the constants by just assuming *unique names*, without changing the certain answers. As we said before, an interpretation $\mathcal{I}$ embedding a database $\mathcal{DB}$ satisfies the standard name assumption – written $\mathcal{I}_{(\mathcal{DB})^{\textit{SNA}}}$ – if $c^{\mathcal{I}} = c$ for any $c \in \mathbb{C}$. Alternatively, an interpretation $\mathcal{I}$ embedding a database $\mathcal{DB}$ satisfies the unique name assumption – written $\mathcal{I}_{(\mathcal{DB})^{\textit{UNA}}}$ – if $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for any different $a, b \in \mathbb{C}$. The following proposition allows us to freely interchange the standard name and the unique name assumptions with interpretations embedding databases. This is of practical advantage, since we can encode the unique name assumption in classical first-order logic reasoners, and most description logics reasoners do have a native unique name assumption.

Proposition 1 (SNA vs UNA). *For any query* $\mathcal{Q}_{[\mathbb{X}]}$*, ontology* $\mathcal{KB}$ *and database* $\mathcal{DB}$*,*

$$\{\Theta^{\mathbb{C}}_{\mathbb{X}} \mid \forall \mathcal{I}_{(\mathcal{DB})^{\textit{SNA}}} \in M(\mathcal{KB}) \ : \ \mathcal{I}_{(\mathcal{DB})^{\textit{SNA}}} \models \mathcal{Q}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}\} =$$
$$\{\Theta^{\mathbb{C}}_{\mathbb{X}} \mid \forall \mathcal{I}_{(\mathcal{DB})^{\textit{UNA}}} \in M(\mathcal{KB}) \ : \ \mathcal{I}_{(\mathcal{DB})^{\textit{UNA}}} \models \mathcal{Q}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}\}.$$

Since a query can be an arbitrary first-order formula, its answer can be infinite (since the domain is not restricted to be finite) or it may depend on the domain. For example, the query $Q(x) = \neg Student(x)$ over the database $Student(A), Student(B)$, with domain $\{A, B, C\}$ has the answer $\{x = C\}$, with domain $\{A, B, C, D\}$ has the answer $\{x = C, x = D\}$, and if we change the domain to an infinite one, the answer will be infinite even in presence of such a finite database. Therefore, the notion of *domain independent* queries has been introduced in relational databases. Here we adapt the classical definitions [11,7] to our framework:

Definition 2 (Domain Independence). *A formula $\mathcal{Q}_{[\mathbb{X}]}$ is domain independent iff for every two interpretations $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and $\mathcal{J} = \langle \Delta^{\mathcal{J}}, \cdot^{\mathcal{J}} \rangle$ which agree on the interpretation of the predicates and constants (i.e. $\cdot^{\mathcal{I}} = \cdot^{\mathcal{J}}$), and for every substitution $\Theta_{\mathbb{X}}^{\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}}$:*

$$\begin{aligned} &\textit{act-range}(\Theta_{\mathbb{X}}^{\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}}) \subseteq \Delta^{\mathcal{I}} \ \textit{and}\ \mathcal{I}, \Theta_{\mathbb{X}}^{\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}} \models \mathcal{Q}_{[\mathbb{X}]} \ \textit{iff} \\ &\textit{act-range}(\Theta_{\mathbb{X}}^{\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}}) \subseteq \Delta^{\mathcal{J}} \ \textit{and}\ \mathcal{J}, \Theta_{\mathbb{X}}^{\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}} \models \mathcal{Q}_{[\mathbb{X}]}. \end{aligned}$$

A weaker version of domain independence is the following.

Definition 3 (Ground Domain Independence). *A formula $\mathcal{Q}_{[\mathbb{X}]}$ is ground domain independent iff $\mathcal{Q}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}}]}$ is domain independent for every substitution $\Theta_{\mathbb{X}}^{\mathbb{C}}$.*

The problem of checking whether a FOL formula is domain independent is undecidable [7]. The well known *safe-range* syntactic fragment of FOL introduced by Codd is an *equally expressive* language; indeed any safe-range formula is domain independent, and any domain independent formula can be easily transformed into a logically equivalent safe-range formula. Intuitively, a formula is safe-range if and only if its variables are bounded by positive predicates or equalities – for the exact syntactical definition see, e.g., [7]. For example, the formula $\neg A(x) \wedge B(x)$ is safe-range, while queries $\neg A(x)$ and $\forall x.\, A(x)$ are not. To check whether a formula is safe-range, the formula is transformed into a logically equivalent *safe-range normal form* and its *range restriction* is computed according to a set of syntax based rules; the range restriction of a formula is a subset of its free variables, and if it coincides with the free variables then the formula is said to be safe-range [7]. Similar to domain independence, a formula is *ground safe-range* if any grounding of this formula is safe-range. An ontology $\mathcal{KB}$ is safe-range (domain independent), if every formula in $\mathcal{KB}$ is safe-range (domain independent).

The safe-range fragment of first-order logic with the standard name assumption is equally expressive to the relational algebra, which is the core of SQL [7].

3 Determinacy

The certain answer to a query includes all the substitutions which make the query true in *all* the models of the ontology embedding the database: so, if a substitution would make the query true only in some model, then it would be discarded from the certain answer. In other words, it may be the case that the answer to the query is not necessarily the same among all the models of the knowledge base embedding the database. In this case, the query is not fully determined by the given source data; indeed, there is some answer which is possible, but not certain. Due to the indeterminacy of the query with respect to the data, the complexity to compute the certain answer in general increases up to the complexity of entailment in the logic. In this paper we focus on the case when a query has the same answer over all the models of the ontology embedding the database, namely, when the information requested by the query is fully available from the source data without ambiguity. In this way, the indeterminacy disappears, and the complexity of the process may decrease (see Section 4). The *determinacy* of a query w.r.t. a source database [3,2,4] has been called *implicit definability* of a formula (the query) from a set of predicates (the database predicates) by [12].

Definition 4 (Finite Determinacy or Implicit Definability). *Let $\mathcal{I}$ and $\mathcal{J}$ be any two models of the ontology $\mathcal{KB}$, both with a* finite *interpretation to the database predicates $\mathbb{P}_{\mathcal{DB}}$. A query $\mathcal{Q}_{[\mathbb{X}]}$ is (finitely) determined by (or implicitly definable from) the database predicates $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$ iff $\mathcal{I}|_{\mathbb{P}_{\mathcal{DB}} \cup \mathbb{C}} = \mathcal{J}|_{\mathbb{P}_{\mathcal{DB}} \cup \mathbb{C}}$ implies that for every substitution $\Theta_{\mathbb{X}}^{\Delta^{\mathcal{I}}} : \mathcal{I}, \Theta_{\mathbb{X}}^{\Delta^{\mathcal{I}}} \models \mathcal{Q}_{[\mathbb{X}]}$ iff $\mathcal{J}, \Theta_{\mathbb{X}}^{\Delta^{\mathcal{I}}} \models \mathcal{Q}_{[\mathbb{X}]}$.*

Intuitively, the answer of an implicitly definable query does not depend on the interpretation of non-database predicates. Once the database and a domain are fixed, it is never the case that a substitution would make the query true in some model of the knowledge base and false in others, since the truth value of an implicitly defined query depends only on the interpretation of the database predicates and constants and on the domain (which are fixed).

In the following we focus on those fragments of $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ for which determinacy under models with a finite interpretation of database predicates (finite determinacy) and determinacy under models with an unrestricted interpretation of database predicates (unrestricted determinacy) coincide. We say that these fragments have *finitely controllable determinacy*: we require that whenever a query is finitely determined then it is also determined in unrestricted models (the reverse is trivially true). Indeed, the results in this paper would fail if finite determinacy and unrestricted determinacy do not coincide: it can be shown (from [13]) that Theorem 1 below fails if we consider only models with a finite interpretation of database predicates.

Example 1 (Example from database theory). *Let $\mathbb{P} = \{P, R, Q\}$, $\mathbb{P}_{\mathcal{DB}} = \{P, R\}$,*

$$\mathcal{KB} = \{\forall x, y, z.\, R(x, y) \wedge R(x, z) \rightarrow y = z, \quad \forall x, y.\, R(x, y) \rightarrow \exists z.\, R(z, x),$$
$$(\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(y, z)) \rightarrow (\forall x.\, Q(x) \leftrightarrow P(x))\}.$$

The formula $\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(y, z)$ is entailed from the first two formulas only *over finite interpretations of R. The query $Q(x)$ is finitely determined by P (it is equivalent to it under the models with a finite interpretation of R), but it is not determined by any database predicate under models with an unrestricted interpretation of R. This fragment of $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ does not enjoy finitely controllable determinacy.*

The *exact reformulation* of a query [3] (also called *explicit definition* [12]) is a formula logically equivalent to the query which makes use *only* of database predicates and constants.

Definition 5 (Exact Reformulation or Explicit Definability). *A query $\mathcal{Q}_{[\mathbb{X}]}$ is explicitly definable from the database predicates $\mathbb{P}_{\mathcal{DB}}$ under the ontology $\mathcal{KB}$ iff there is some formula $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$, such that $\mathcal{KB} \models \forall \mathbb{X}. \mathcal{Q}_{[\mathbb{X}]} \leftrightarrow \widehat{\mathcal{Q}}_{[\mathbb{X}]}$ and $\sigma(\widehat{\mathcal{Q}}) \subseteq \mathbb{P}_{\mathcal{DB}}$. We call this formula $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ an exact reformulation of $\mathcal{Q}_{[\mathbb{X}]}$ under $\mathcal{KB}$ over $\mathbb{P}_{\mathcal{DB}}$.*

Determinacy of a query is completely characterised by the existence of an exact reformulation of the query: it is well known that a first-order query is determined by database predicates *if and only if* there exists a first-order exact reformulation.

Theorem 1 (Projective Beth definability [12]). *A query $\mathcal{Q}$ is implicitly definable from the database predicates $\mathbb{P}_{\mathcal{DB}}$ under an ontology $\mathcal{KB}$, iff it is explicitly definable as a formula $\widehat{\mathcal{Q}}$ in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ over $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$.*

Let $\mathcal{Q}$ be any formula in $\mathcal{L}$ and $\widetilde{\mathcal{Q}}$ the formula obtained from it by uniformly replacing every occurrence of each non-database predicate P with a new predicate $\widetilde{P}$. We extend this renaming operator $\widetilde{\cdot}$ to any set of formulas in a natural way. One can check whether a query is implicitly definable by using the following theorem.

Theorem 2 (Testing Determinacy [12]). *A query $\mathcal{Q}_{[\mathbb{X}]}$ is implicitly definable from the database predicates $\mathbb{P}_{\mathcal{DB}}$ under the constraints $\mathcal{KB}$ iff $\mathcal{KB} \cup \widetilde{\mathcal{KB}} \models \forall X.\mathcal{Q}_{[\mathbb{X}]} \leftrightarrow \widetilde{\mathcal{Q}}_{[\mathbb{X}]}$.*

4 Exact Safe-Range Query Reformulation

In this section we analyse the conditions under which the original query answering problem – corresponding to an entailment problem – can be reduced systematically to a model checking problem of a safe-range formula over the database (e.g., using a database system with SQL). Given a database signature $\mathbb{P}_{\mathcal{DB}}$, an ontology $\mathcal{KB}$, and a query $\mathcal{Q}_{[\mathbb{X}]}$ expressed in $\mathcal{L}$ and determined by the database predicates, our goal is to find a safe-range reformulation $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ of $\mathcal{Q}_{[\mathbb{X}]}$ in $\mathcal{FOL}(\mathbb{C},\mathbb{P})$, that when evaluated as a relational algebra expression over a legal database instance gives the same answer as the certain answer of $\mathcal{Q}_{[\mathbb{X}]}$ to the database under $\mathcal{KB}$. This can be reformulated as the following problem:

*Problem 1 (**Exact Safe-Range Query Reformulation**).* Find an exact reformulation $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ of $\mathcal{Q}_{[\mathbb{X}]}$ under $\mathcal{KB}$ as a safe-range query in $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ over $\mathbb{P}_{\mathcal{DB}}$.

Since an exact reformulation is equivalent under the ontology to the original query, the certain answer of the original query and of the reformulated query are identical. More precisely, the following proposition holds.

Proposition 2. *Let $\mathcal{Q}_{[\mathbb{X}]}$ be implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$ and $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ be an exact reformulation of $\mathcal{Q}_{[\mathbb{X}]}$ under $\mathcal{KB}$ over $\mathbb{P}_{\mathcal{DB}}$, then:*

$$\{\Theta_{\mathbb{X}}^{\mathbb{C}} \mid \forall \mathcal{I}_{(\mathcal{DB})} \in M(\mathcal{KB}) : \mathcal{I}_{(\mathcal{DB})} \models \mathcal{Q}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}}]}\} = \{\Theta_{\mathbb{X}}^{\mathbb{C}} \mid \forall \mathcal{I}_{(\mathcal{DB})} \in M(\mathcal{KB}) : \mathcal{I}_{(\mathcal{DB})} \models \widehat{\mathcal{Q}}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}}]}\}.$$

From the above equation it is clear that in order to answer an exactly reformulated query, one still may need to consider all the models $\mathcal{I}_{(\mathcal{DB})}$ of the ontology embedding the database – i.e., we still have an entailment problem to solve. The following theorem states the condition to reduce the original query answering problem – based on entailment – to the problem of checking the validity of the exact reformulation over a *single* model: the condition is that the reformulation should be domain independent.

Theorem 3 (Adequacy of Exact safe-range Query Reformulation). *Let $\mathcal{DB}$ be a database which is legal for $\mathcal{KB}$, and $\mathcal{Q}_{[\mathbb{X}]}$ be a query. If $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ is an exact domain independent (or safe-range) reformulation of $\mathcal{Q}_{[\mathbb{X}]}$ under $\mathcal{KB}$ over $\mathbb{P}_{\mathcal{DB}}$, then:*

$$\{\Theta_{\mathbb{X}}^{\mathbb{C}} \mid \forall \mathcal{I}_{(\mathcal{DB})} \in M(\mathcal{KB}) : \mathcal{I}_{(\mathcal{DB})} \models \mathcal{Q}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}}]}\} =$$
$$\{\Theta_{\mathbb{X}}^{adom(\sigma(\widehat{\mathcal{Q}}),\mathcal{DB})} \mid \mathcal{I}_{(\mathcal{DB})}^{(\mathbb{C})}|_{\mathbb{P}_{\mathcal{DB}} \cup \mathbb{C}} \models \widehat{\mathcal{Q}}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{adom(\sigma(\widehat{\mathcal{Q}}),\mathcal{DB})}]}\}.$$

A safe-range reformulation is *necessary* to transform a first-order query to a relational algebra query which can then be evaluated by using SQL techniques. The theorem above shows in addition that being safe-range is also a *sufficient* property for an exact reformulation to be correctly evaluated as an SQL query. Let us now see an example in which we can not reduce the problem of answering an exact reformulation to model checking over a database, if the exact reformulation is not safe-range.

Example 2. *Let* $\mathbb{P} = \{P, A\}$, $\mathbb{P}_{\mathcal{DB}} = \{P\}$, $\mathbb{C} = \{a\}$,
$\mathcal{DB} = \{P(a, a)\}$, $\mathcal{KB} = \{\forall y.\, P(a, y) \lor A(y)\}$,
$\mathcal{Q} = \widehat{\mathcal{Q}} = \forall y.\, P(x, y)$.

- *$\mathcal{DB}$ is legal for $\mathcal{KB}$ because there is $\mathcal{I}_{(\mathcal{DB})} = \langle\{a\}, \cdot^{\mathcal{I}(\mathcal{DB})}\rangle$ such that $P^{\mathcal{I}(\mathcal{DB})} = \{(a, a)\}$, $A^{\mathcal{I}(\mathcal{DB})} = \emptyset$ and obviously, $\mathcal{I}_{(\mathcal{DB})} \in M(\mathcal{KB})$.*
- *$\{\Theta^{\mathbb{C}}_{\mathbb{X}} \mid \forall \mathcal{I}_{(\mathcal{DB})} \in M(\mathcal{KB}) : \mathcal{I}_{(\mathcal{DB})} \models \mathcal{Q}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}\} = \emptyset$ because one can take $\mathcal{I}_{(\mathcal{DB})} = \langle\{a, b\}, \cdot^{\mathcal{I}(\mathcal{DB})}\rangle$ such that $P^{\mathcal{I}(\mathcal{DB})} = \{(a, a)\}$, $A^{\mathcal{I}(\mathcal{DB})} = \{b\}$; then $\mathcal{I}_{(\mathcal{DB})} \in M(\mathcal{KB})$, but for the only possible substitution $\{x \to a\}$, $\mathcal{I}_{(\mathcal{DB})} \not\models \forall y\, P(a, y)$.*
- *However, $\{\Theta^{\mathbb{C}}_{\mathbb{X}} \mid \mathcal{I}^{(\mathbb{C})}_{(\mathcal{DB})}|_{\mathbb{P}_{\mathcal{DB}}} \models \widehat{\mathcal{Q}}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}\} = \{x \to a\}$.* □

5 Conditions for an Exact Safe-Range Reformulation

We have just seen the importance of getting an exact safe-range query reformulation. In this section we are going to study the conditions under which an exact safe-range query reformulation exists. While implicit definability is – as we already know – a sufficient condition for the existence of an exact reformulation, it does not guarantee alone the existence of a safe-range reformulation.

Example 3. *Let* $\mathbb{P} = \{A, B\}$, $\mathbb{P}_{\mathcal{DB}} = \{A\}$, $\mathcal{KB} = \{\forall x.\, B(x) \leftrightarrow \neg A(x)\}$, $\mathcal{Q} = B(x)$. *Then $\mathcal{Q}$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$, and every exact reformulation of $\mathcal{Q}$ over $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$ is not safe-range.* □

By looking at the example, it seems that the reason for the non-safe-range reformulation lies in the fact that the query returns non-database domain elements – due to the universal quantification. So, we try to restrict the queries to be $\mathcal{DB}$-relativised, namely they should return only database objects.

Definition 6 ($\mathcal{DB}$-relativised query). *An open query $\mathcal{Q}_{[\mathbb{X}]}$ is $\mathcal{DB}$-relativised under $\mathcal{KB}$, if in each model of $\mathcal{KB}$ the extension domain of $\mathcal{Q}_{[\mathbb{X}]}$ includes only domain elements which are among the interpretation of database predicates or constants from $\mathcal{KB}$ or $\mathcal{Q}_{[\mathbb{X}]}$. A closed query is $\mathcal{DB}$-relativised under any ontology.*

Example 4. *Let* $\mathbb{P} = \{A, B, C, D\}$, $\mathbb{P}_{\mathcal{DB}} = \{D\}$,
$\mathcal{KB} = \{\forall x.\, \neg A(x) \land B(x) \leftrightarrow D(x), \forall x.\, C(x) \to x = a\}$,
$\mathcal{Q}(x) = \neg A(x) \land B(x) \lor C(x)$.
Then, since $\mathcal{KB} \models \mathcal{Q}(x) \to D(x) \lor x = a$, D is a database predicate and a is constant in $\mathcal{KB}$, it is easy to see, that the query $\mathcal{Q}(x)$ is $\mathcal{DB}$-relativised under $\mathcal{KB}$. □

With $\mathcal{DB}$-relativised queries, the following theorem holds, giving the *semantic* (non-constructive) requirements for the existence of an exact reformulation.

Theorem 4 (Semantic). *Let $\mathbb{X} = \{x_1, \ldots, x_n\}$. If $\mathcal{Q}_{[\mathbb{X}]}$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$, ground domain independent, and $\mathcal{DB}$-relativised under $\mathcal{KB}$, and $\mathcal{KB}$ is domain independent, then there exists an exact reformulation $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ of $\mathcal{Q}_{[\mathbb{X}]}$ as a safe-range query in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ over $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$.*

Let us now discuss the conditions of the theorem.

The converse of theorem 4 in general does not hold.

In fact, if there exists a $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ exact safe-range reformulation $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ of $\mathcal{Q}_{[\mathbb{X}]}$ over $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$, then

- $\mathcal{Q}_{[\mathbb{X}]}$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$ given the Beth definability property of FOL.
- $\mathcal{Q}_{[\mathbb{X}]}$ is $\mathcal{DB}$-relativised under $\mathcal{KB}$ since $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ is $\mathcal{DB}$-relativised under $\mathcal{KB}$ and $\mathcal{Q}_{[\mathbb{X}]}$ is logically equivalent to $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ w.r.t. $\mathcal{KB}$.
- $\mathcal{Q}_{[\mathbb{X}]}$ may not be ground domain independent, and $\mathcal{KB}$ may be not domain independent.

Example 5. *Let* $\mathbb{P} = \{A, B, C\}$, $\mathbb{P}_{\mathcal{DB}} = \{B\}$, $\mathcal{KB} = \{\forall x.\, A(x) \wedge \forall y.\, C(y) \leftrightarrow B(x)\}$, $\mathcal{Q}(x) = A(x) \wedge \forall y.\, C(y)$. $\widehat{\mathcal{Q}}(x) = B(x)$ *is a safe-range exact reformulation of* $\mathcal{Q}(x)$ *over* $\mathbb{P}_{\mathcal{DB}}$ *under* $\mathcal{KB}$*. But* $\mathcal{Q}(x)$ *is not ground domain independent, and* $\mathcal{KB}$ *is not domain independent.* □

We cannot neglect the ground domain independence of the query.

Example 6. *Let* $\mathbb{P} = \{A, B\}$, $\mathbb{P}_{\mathcal{DB}} = \{B\}$,
$\mathcal{KB} = \{\forall x.\, A(x) \leftrightarrow B(x)\}$, $\mathcal{Q} = \forall x.\, A(x)$.

- *$\mathcal{KB}$ is domain independent;*
- *$\mathcal{Q}$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$, because it is explicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$: $\mathcal{KB} \models \forall x.\, A(x) \leftrightarrow \forall y.\, B(y)$;*
- *$\mathcal{Q}$ is $\mathcal{DB}$-relativised under $\mathcal{KB}$ because $\mathcal{Q}$ is closed;*
- *$\mathcal{Q}$ is not ground domain independent;*
- *All the exact reformulations of $\mathcal{Q}$ over $\mathbb{P}_{\mathcal{DB}}$ are logically equivalent w.r.t. $\mathcal{KB}$ to $\forall x.\, B(x)$, which is not domain independent. Therefore, they are not safe-range.* □

We cannot neglect domain independence of $\mathcal{KB}$.

Example 7. *Let* $\mathbb{P} = \{P, A\}$, $\mathbb{P}_{\mathcal{DB}} = \{A\}$,
$\mathcal{KB} = \{\exists x.\, P(x) \leftrightarrow \exists y.\neg A(y)\}$, $\mathcal{Q} = \exists x.\, P(x)$.

- *$\mathcal{Q}$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$ because the first sentence of $\mathcal{KB}$ gives an explicit definition of $\mathcal{Q}$;*
- *$\mathcal{Q}$ is ground domain independent;*
- *Since $\mathcal{Q}$ is closed, it is always $\mathcal{DB}$-relativised under $\mathcal{KB}$;*
- *$\mathcal{KB}$ is not domain independent;*
- *All the exact reformulations of $\mathcal{Q}$ over $\mathbb{P}_{\mathcal{DB}}$ are logically equivalent w.r.t. $\mathcal{KB}$ to $\exists y.\neg A(y)$, which is not domain independent. Therefore, they are not safe-range.* □

We can weaken domain independence of $\mathcal{KB}$.

Example 8. *Let* $\mathbb{P} = \{A, B, C\}$, $\mathbb{P}_{\mathcal{DB}} = \{C\}$,
$\mathcal{KB} = \{\forall x.\, A(x) \leftrightarrow C(x),\ \forall x.\, B(x)\}$, $\mathcal{Q}(x) = A(x) \wedge B(x)$.

- *$\mathcal{Q}(x)$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$ because the first sentence of $\mathcal{KB}$ gives an explicit definition of $\mathcal{Q}(x)$;*
- *$\mathcal{Q}$ is domain independent;*
- *$\mathcal{Q}(x)$ is $\mathcal{DB}$-relativised under $\mathcal{KB}$ because of the first sentence of the $\mathcal{KB}$.*
- *$\widehat{\mathcal{Q}}(x) = C(x)$ is an exact safe-range reformulation of $\mathcal{Q}(x)$ from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$.*

But $\mathcal{KB}$ is not domain independent. □

It is still open the problem to understand which part of $\mathcal{KB}$ is irrelevant for the reformulation and therefore does not need to be domain independent.

The above theorem shows us the semantic conditions to have an exact safe-range reformulation of a query, but it does not give us a method to compute such reformulation. The following *constructive theorem* gives us sufficient conditions for the existence of an exact safe-range reformulation in any decidable fragment of $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ where finite and unrestricted determinacy coincide, and compute it, if it exists.

Theorem 5 (Constructive)
Let $\mathbb{X} = \{x_1, \ldots, x_n\}$. *If*

1. $\mathcal{KB} \cup \widetilde{\mathcal{KB}} \models \forall \mathbb{X}.\, \mathcal{Q}_{[\mathbb{X}]} \leftrightarrow \widetilde{\mathcal{Q}}_{[\mathbb{X}]}$ *(i.e.,* $\mathcal{Q}_{[\mathbb{X}]}$ *is implicitly definable),*
2. $\mathcal{Q}_{[\mathbb{X}]}$ *is ground safe-range,*
3. $\mathcal{KB}$ *is safe-range,*
4. *there are* n *safe-range formulas* $\psi_1, \ldots, \psi_n$ *over* $\mathbb{P}_{\mathcal{DB}}$ *with constants from* $\mathcal{KB}$ *and* $\mathcal{Q}_{[\mathbb{X}]}$*, that we call* $\mathcal{DB}$*-sorts, such that* $\mathcal{KB} \models \forall \mathbb{X}.\, \mathcal{Q}_{[\mathbb{X}]} \to \psi_1(x_1) \wedge \ldots \wedge \psi_n(x_n)$*,*

then there exists an exact reformulation $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ *of* $\mathcal{Q}_{[\mathbb{X}]}$ *as a safe-range query in* $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ *over* $\mathbb{P}_{\mathcal{DB}}$ *under* $\mathcal{KB}$*, that can be obtained constructively, and each free variable* x_i *in* $\widehat{\mathcal{Q}}$ *appears in a* $\mathcal{DB}$*-sort* ψ_i *as a top level conjunct.*

In order to compute the exact safe-range query reformulation we use the tableau based method to find the Craig's interpolant (see [14,15]) to compute $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ from a validity proof of the implication $(\mathcal{KB} \wedge \mathcal{Q}_{[\mathbb{X}]}) \to (\widetilde{\mathcal{KB}} \to \widetilde{\mathcal{Q}}_{[\mathbb{X}]})$. Let us now consider a fully worked out example, adapted from [3].

Example 9. *Given:* $\mathbb{P} = \{R, V_1, V_2, V_3, A\}$, $\mathbb{P}_{\mathcal{DB}} = \{V_1, V_2, V_3, A\}$,
$\mathcal{KB} = \{\forall x, y.\, V_1(x,y) \leftrightarrow \exists z, v.\, R(z,x) \wedge R(z,v) \wedge R(v,y),$
$\forall x, y.\, V_2(x,y) \leftrightarrow \exists z.\, R(x,z) \wedge R(z,y),$
$\forall x, y.\, V_3(x,y) \leftrightarrow \exists z, v.\, R(x,z) \wedge R(z,v) \wedge R(v,y),$
$\forall x, y.\, R(x,y) \to A(y)\}.$
$\mathcal{Q}(x,y) = \exists z, v, u.\, R(z,x) \wedge R(z,v) \wedge R(v,u) \wedge R(u,y).$

The conditions of the constructive theorem are satisfied: $\mathcal{Q}(x,y)$ *is implicitly definable from* $\mathbb{P}_{\mathcal{DB}}$ *under* $\mathcal{KB}\mathcal{Q}(x,y)$ *is ground safe-range;* $\mathcal{KB}$ *is safe-range;* $\mathcal{Q}(x,y)$ *is* $\mathcal{DB}$*-relativised, and* $A(x)$ *and* $A(y)$ *are the* $\mathcal{DB}$*-sorts.*

Therefore, with the tableau method one finds the Craig's interpolant to compute $\widehat{\mathcal{Q}}(x,y)$ *from a validity proof of the implication* $(\mathcal{KB} \wedge \mathcal{Q}_{[\mathbb{X}]}) \to (\widetilde{\mathcal{KB}} \to \widetilde{\mathcal{Q}}_{[\mathbb{X}]})$ *and obtain* $\widehat{\mathcal{Q}}(x,y) = \exists z.\, V_1(x,z) \wedge \forall v.\, (V_2(v,z) \to V_3(v,y))$ *- an exact ground safe-range reformulation. Since* $\widehat{\mathcal{Q}}(x,y)$ *is exact and* $A(x), A(y)$ *are* $\mathcal{DB}$*-sorts,* $\mathcal{KB} \models \widehat{\mathcal{Q}}(x,y) \to A(x) \wedge A(y)$*. Then* $\mathcal{KB} \models \mathcal{Q}(x,y) \leftrightarrow \widehat{\mathcal{Q}}(x,y) \wedge A(x) \wedge A(y)$*. Therefore,* $\exists z.\, V_1(x,z) \wedge \forall v.\, (V_2(v,z) \to V_3(v,y)) \wedge A(x) \wedge A(y)$ *is an exact safe-range reformulation of* $\mathcal{Q}(x,y)$ *from* $\mathbb{P}_{\mathcal{DB}}$ *under* $\mathcal{KB}$*.* □

6 A Case Study: $\mathcal{SHOQ}$

$\mathcal{SHOQ}$ is an extension of the description logic $\mathcal{ALC}$ with transitive roles, role hierarchies, qualified number restrictions, and individuals; it is a fragment of first-order logic

Syntax	Semantics
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
$\exists R.C$	$\{x \mid \text{exists } y \text{ such that } (x, y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
$\forall R.C$	$\{x \mid \text{forall y } (x, y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$
$\{o\}$	$\{o\}^I \subseteq \Delta^I$
$\geq nR.C$	$\{x \mid \#(\{y \mid (x, y) \in R^{\mathcal{I}}\} \cap C^{\mathcal{I}}) \geq n\}$
$\leq nR.C$	$\{x \mid \#(\{y \mid (x, y) \in R^{\mathcal{I}}\} \cap C^{\mathcal{I}}) \leq n\}$

and of OWL2. The syntax and semantics of $\mathcal{SHOQ}$ is summarised in the next Table, where A is an atomic concept, C and D are concepts, o is an individual name and R is an atomic role. $\mathcal{SHOQ}$ is a rather standard description logic; for more details see, e.g., [16]. A TBox in $\mathcal{SHOQ}$ is a set of concept inclusion axioms $C \sqsubseteq D$, role inclusion axioms $R \sqsubseteq S$, and transitivity axioms $\mathit{Trans}(R)$ (where C, D are concepts and R, S are atomic roles) with the usual semantics. In this section, we present an application of our framework where the ontology is a TBox in $\mathcal{SHOQ}$, the query is a concept query – namely a concept expression denoting an open formula with one free variable – or a closed concept query – a concept query where the free variable has been substituted with a constant – and the database is a set of expressions of the form $A(a)$ and $R(a, b)$ (where A is an atomic concept and R is an atomic role) [10].

Finitely Controllable Determinacy. The constructive theorem holds in any fragment of $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ for a safe-range ontology $\mathcal{KB}$ and a $\mathcal{DB}$-relativised query $\mathcal{Q}_{[\mathbb{X}]}$ if the entailment $\mathcal{KB} \cup \widetilde{\mathcal{KB}} \models \forall \mathbb{X}.\ \mathcal{Q}_{[\mathbb{X}]} \leftrightarrow \widetilde{\mathcal{Q}}_{[\mathbb{X}]}$, that characterises implicit definability of $\mathcal{Q}_{[\mathbb{X}]}$, is finitely controllable. The finite controllability of this entailment in $\mathcal{SHOQ}$ is guaranteed because the entailment $\mathcal{KB} \cup \widetilde{\mathcal{KB}} \models \mathcal{Q} \equiv \widetilde{\mathcal{Q}}$ can be reduced in $\mathcal{SHOQ}$ to a concept satisfiability problem for an empty TBox, and $\mathcal{SHOQ}$ has the finite model property [17].

Safe-Range Ontology. We call any axiom (concept) in $\mathcal{SHOQ}$ (ground) safe-range, if the corresponding logically equivalent (open) formula in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ is (ground) safe-range. Role inclusion and transitivity axioms are always safe-range. For any concept C we denote the corresponding logically equivalent formula in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ with one free variable x as $C(x)$. Unfortunately concept inclusion axioms in $\mathcal{SHOQ}$ ontologies may not be safe-range: for example, the axiom $\neg$ `male` $\sqsubseteq$ `female` is not safe-range. It is easy to see that an axiom $C \sqsubseteq D$ is not safe-range if and only if $C(x)$ is not safe-range and $D(x)$ is safe-range: just observe that the axiom is logically equivalent to the formula $\neg \exists x.\ C(x) \wedge \neg D(x)$ in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$. The following proposition provides rules deciding whether a $\mathcal{SHOQ}$ concept is safe-range.

Proposition 3. *Let A be an atomic concept, C and D be $\mathcal{SHOQ}$ concepts. Then:*

1. $A(x), (\exists R.C)(x), \{o\}(x), (\geq nR.C)(x)$ *are safe-range;*
2. $(\forall R.C)(x), (\leq nR.C)(x)$ *are not safe-range;*
3. $(C \sqcap D)(x)$ *is safe-range if and only if* $C(x)$ *is safe-range or* $D(x)$ *is safe-range;*
4. $(C \sqcup D)(x)$ *is safe-range if and only if* $C(x)$ *is safe-range and* $D(x)$ *is safe-range;*

5. *$\neg C(x)$ is safe-range if and only if $C(x)$ is not safe-range.*

Proposition 4. *For any $\mathcal{SHOQ}$ concept C and constant a, $C(x)$ is ground safe-range, and $C(a)$ is safe-range.*

The presence of non-safe-range axioms in an ontology would prevent the application of our framework, but we argue that non-safe-range axioms should not appear in a cleanly designed $\mathcal{SHOQ}$ ontology, and, if present, they should be fixed. Indeed, the use of *absolute* negative information in the subsumee – such as in "a non-male is a female" ($\neg$ male $\sqsubseteq$ female) – should be discouraged by a clean design methodology, since the subsumer would include *all sorts* of objects in the universe (but the ones of the subsumee type) without any obvious control. Only *relativised* negative information in the subsumee should be allowed – such as in the axiom "a non-male person is a female" (person $\sqcap \neg$ male $\sqsubseteq$ female). This observations suggests a fix for non-safe-range axioms: for every non-safe-range axiom $C \sqsubseteq D$ users will be asked to replace it by the safe-range one $C \sqcap E \sqsubseteq D$ where E is an arbitrary safe-range concept. Therefore, the user is asked to make explicit the *type* of the subsumee, in a way to make it domain independent; note that the type could be also a fresh new atomic concept. We believe that the fix we are proposing for $\mathcal{SHOQ}$ is a reasonable one, and would make all $\mathcal{SHOQ}$ ontologies eligible to be used with our framework.

Ground safe-range and $\mathcal{DB}$-relativised query. Let $\mathcal{KB}$ be a $\mathcal{SHOQ}$ ontology, and $\mathcal{Q}(x)$ an implicitly definable query, which is a possibly complex concept in $\mathcal{SHOQ}$. In order to use our framework, a query should be ground safe-range and DB-relativised under the ontology. We already know that by Proposition 4 the query is ground safe-range. A query is DB-relativised if it returns only database objects; it may be strange for a user to issue a query which is not meant to return just database objects. Therefore, in the rare case a user is issuing a non-DB-relativised query, we would ask the user to conjoin the query with a safe-range concept composed only by database atomic concepts, which would become the *type* of the query. The type plays the role of the $\mathcal{DB}$-sort of theorem 5. We believe that also this fix for the queries is a reasonable one, and would make all queries eligible to be used with our framework.

7 Conclusion

We have introduced a framework to compute the exact reformulation of first-order queries to a database under ontologies. We have found the conditions which guarantee that a safe-range reformulation exists, and we show that it can be evaluated as a relational algebra query over the database to give the same answer as the original query under the ontology. A non-trivial case study has been presented in the field of description logics, with the $\mathcal{SHOQ}$ language.

We have also implemented a tool based on the *Prover9* theorem prover [18]: the tool manages an arbitrary first-order ontology, a database signature, and an arbitrary first-order query in TPTP syntax, it performs all the tests on them to check whether a reformulation can be computed, and it computes an optimal safe-range reformulation.

This framework, even with the limitation of atomic queries, is useful in data exchange-like scenarios, where the target database (made by determined relations) should be materialised as a proper database, over which arbitrary queries should be performed. This is not achieved in a context with non-exact rewritings preserving the certain answers. In our scenario rewritings are precomputed offline once. Our framework works also in the case of arbitrary queries, and our tool shows that this is possible in practice.

As a future work, we would like to study optimisations of reformulations. From the practical perspective, since there might be many rewritten queries from one original query, the problem of selecting an optimised one in terms of query evaluation is very important. In fact, one has to take into account which criteria should be used to optimise, such as: the size of the rewritings, the numbers of used predicates, the priority of predicates, the number of relational operators, and clever usage of duplicates. With the tool we plan to evaluate our proposed technique in a real context.

References

1. Etzioni, O., Golden, K., Weld, D.S.: Sound and efficient closed-world reasoning for planning. Artif. Intell. 89, 113–148 (1997)
2. Marx, M.: Queries determined by views: pack your views. In: Proceedings of the 26th ACM Symposium on Principles of Database Systems, PODS 2007, pp. 23–30 (2007)
3. Nash, A., Segoufin, L., Vianu, V.: Views and queries: Determinacy and rewriting. ACM Trans. Database Syst. 35, 21:1–21:41 (2010)
4. Fan, W., Geerts, F., Zheng, L.: View determinacy for preserving selected information in data transformations. Inf. Syst. 37, 1–12 (2012)
5. Seylan, İ., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over DBoxes. In: Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 923–925 (2009)
6. Franconi, E., Ibanez-Garcia, Y.A., Seylan, İ.: Query answering with DBoxes is hard. Electronic Notes in Theoretical Computer Science 278, 71–84 (2011)
7. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
8. Halevy, A.Y.: Answering queries using views: A survey. The VLDB Journal 10, 270–294 (2001)
9. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res (JAIR) 36, 1–69 (2009)
10. Franconi, E., Kerhet, V., Ngo, N.: Exact query reformulation over SHOQ DBoxes. In: Proc. of the 2012 International Workshop on Description Logics, DL 2012 (2012)
11. Avron, A.: Constructibility and decidability versus domain independence and absoluteness. Theor. Comput. Sci. 394, 144–158 (2008)
12. Beth, E.: On Padoa's method in the theory of definition. Indagationes Mathematicae 15, 330–339 (1953)
13. Gurevich, Y.: Toward logic tailored for computational complexity. In: Computation and Proof Theory, vol. 1104, pp. 175–216. Springer (1984)
14. Fitting, M.: First-order logic and automated theorem proving, 2nd edn. Springer (1996)
15. Borgida, A., de Bruijn, J., Franconi, E., Seylan, İ., Straccia, U., Toman, D., Weddell, G.E.: On finding query rewritings under expressive constraints. In: Proc. of the 18th Italian Symposium on Advanced Database Systems (SEBD 2010), pp. 426–437 (2010)
16. Horrocks, I., Sattler, U.: Ontology reasoning in the SHOQ(D) description logic. In: Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001), pp. 199–204 (2001)
17. Lutz, C., Areces, C., Horrocks, I., Sattler, U.: Keys, nominals, and concrete domains. J. Artif. Int. Res. 23, 667–726 (2005)
18. McCune, W.: Prover9 and Mace4 (2005-2011), `http://www.cs.unm.edu/~mccune/prover9`

A Selective Semantics for Logic Programs with Preferences

Alfredo Gabaldon

Center for Artificial Intelligence (CENTRIA)
Universidade Nova de Lisboa
a.gabaldon@fct.unl.pt

Abstract. Answer Set Programming (ASP), has become a prominent approach for Knowledge Representation and declarative problem solving. ASP has been extended with preferences and several semantics have been proposed. Among the available semantics, the so-called *selective* semantics have the property of always choosing preferred answer sets from among the standard answer sets, which is a desirable property in many applications. However, there exist programs which while having answer sets, have no preferred answer sets under the existing selective semantics. On the other hand, there are programs which, arguably, have too many preferred answer sets under existing semantics. We propose a new definition of preferred answer sets which is also selective, assigns at most one preferred answer set given a full prioritization of a program, and for negative-cycle-free programs the existence of a preferred answer set is guaranteed as in the case of standard logic programs.

1 Introduction

Reasoning with preferences is widely recognized as an important problem. Many knowledge representation formalisms have been extended to represent preferences as priorities between propositions in a knowledge base. In particular, prioritized versions of logic programming with answer set semantics [1, 2] have been studied and various semantics have been proposed [3–8]. Some of these approaches [6–8] are called *selective* because they use the preferences to choose from the answer sets of a logic program some preferred ones. Another characteristic of the selective approaches is that most of them extend logic programs with preferences without increasing computational complexity.

In this work we focus on selective approaches and propose a new definition of preferred answer sets of logic programs with preferences. The main motivation for introducing a new definition is that all of the existing selective approaches appear to be too strong in the sense that some programs have no preferred answer sets even though they have standard answer sets and no apparent conflict. At the same time, under these approaches there are programs where, even with a full set of priorities, it is not possible to choose only one answer set as the preferred one. In other words, the existing selective approaches sometimes choose too few and sometimes too many preferred answer sets. Under our proposed definition, a full set of priorities yields at most one preferred answer set. Moreover, for a class of propositional logic programs (called *negative-cycle-free* and *head-consistent*) a preferred answer set is guaranteed to exist.

It is standard practice in answer set programming (ASP) to combine programs that generate answer sets and then to use additional knowledge in the form of "constraints" to eliminate some answer sets [9]. Constraints are precisely a form of negative cycle. For a program outside the class of negative-cycle-free programs, it is possible under our definition that the program not have any preferred answer sets. This would reflect an inconsistency involving the priorities, the constraints and the rest of the rules in the program. This is consistent with standard practice in ASP, where problems are often solved by checking (non-)existence of answer sets. However, we show that programs that are negative-cycle-free, which always have standard answer sets, are also guaranteed to have preferred answer sets under our new definition. Furthermore, a program with a partial set of priorities may have multiple preferred answer sets, which is consistent with ASP practice where multiple answer sets correspond to multiple solutions of a problem. But a *full* set of priorities always yields at most one preferred answer set.

For comparison, we focus on the semantics introduced by Brewka and Eiter [6] (for brevity, the BE semantics). Among the selective semantics within the NP complexity class [6–8], the BE semantics produces a larger collection, in fact a superset, of preferred answer sets than the other the approaches do [10]. Since one of our goals is a larger class of programs with preferred answer sets, we take the BE semantics as the representative of the selective semantics and center our discussion and comparisons around it. Among other approaches for logic programming with preferences, there are [3, 5], which are in a higher complexity class, and [4] which is not selective. These approaches have their own advantages, but we will not discuss them further as we focus on selective semantics in the same complexity class as programs without preferences.

2 Prioritized Extended Logic Programs

2.1 Extended Logic Programs

We start with the syntax of Extended Logic Programs (elps) [1, 2]. A *literal* is an atom p or its negation $\neg p$. Literals of the same atom, e.g. p and $\neg p$, are called *contrary* and $\overline{l}$ denotes the literal contrary to l. If the language of an elp is not explicitly defined then it is understood to consist of all the literals of all atoms that appear in the program. Lit denotes the set of all literals in the language of an elp. A rule r is an expression of the form[1]

$$l_0 \leftarrow l_1, \ldots, l_n, not\, l_{n+1}, \ldots, not\, l_m \tag{1}$$

where $l_0, \ldots, l_m$ are literals and not denotes negation-as-failure (or *default negation*). Expressions $not\, l$ are called *extended literals*. For a rule r of the form (1), the head, l_0, is denoted by $head(r)$, the set of literals $\{l_1, \ldots, l_n\}$ by $body^+(r)$ and the set of literals $\{l_{n+1}, \ldots, l_m\}$ by $body^-(r)$. The *body* of a rule r is the union $body(r) = body^+(r) \cup body^-(r)$. For a set of rules P, $heads(P) = \{head(r) | r \in P\}$. An *extended logic program* is a finite set of rules.

[1] Instead of rules with empty head, we assume constraints are expressed as $p \leftarrow not\, p, \Gamma$.

A set of literals $S \subseteq Lit$ is called a *partial interpretation*. We say a literal l *defeats a rule* r if $l \in body^-(r)$. A partial interpretation S *defeats a rule* r if there is a literal $l \in S$ that defeats r. S *satisfies the body* of a rule r if $body^+(r) \subseteq S$ and S does not defeat r. S *satisfies* r if $head(r) \in S$ or S does not satisfy the body of r.

The answer sets of an elp whose rules do not contain not are defined as follows.

Definition 1. *Let P be an elp without default negation. A partial interpretation S is an answer set of P if S is minimal (wrt set inclusion) among the partial interpretations that satisfy the rules of P, and S is logically closed, i.e. if S contains contrary literals then S is Lit.*

For arbitrary programs, the definition is extended by introducing the Gelfond-Lifschitz reduct: let S be a partial interpretation and P be an elp. The *reduct*, P^S, of P relative to S is the set of rules $l_0 \leftarrow l_1, \ldots, l_n$ for all rules (1) in P that are not defeated by S.

Definition 2 (Answer Set). *A partial interpretation S is an answer set of an elp P if S is an answer set of P^S.*

2.2 Prioritized Extended Logic Programs

We now turn to *prioritized elps*, adapting the definitions from [6]. Let us start with the syntax. An elp rule r of the form (1) is called *prerequisite-free* if $body^+(r) = \emptyset$ and an elp P is *prerequisite-free* if all its rules are prerequisite-free.

Definition 3 (Prioritized elp). *A prioritized elp is a pair $\mathcal{P} = (P, <)$ where P is an elp and $<$ is a strict partial order on the rules of P.*

A simple example is the program

$$\begin{aligned} r_1 &: a \leftarrow not\, b. \\ r_2 &: b \leftarrow not\, a. \end{aligned}$$

with priority $r_1 < r_2$, i.e. rule r_1 has higher priority than r_2. Of the two answer sets $\{a\}$ and $\{b\}$, the intention is that $\{a\}$ would be the preferred answer set.

The answer sets of a prioritized elp $\mathcal{P} = (P, <)$ are defined as the answer sets of P and are denoted by $AS(\mathcal{P})$.

Definition 4. *A full prioritization of $(P, <)$ is any pair $(P, <')$ where $<'$ is a total order on P that is compatible with $<$, i.e., $r_1 < r_2$ implies $r_1 <' r_2$ for all r_1, r_2 in P.*

The total ordering in a fully prioritized elp induces an enumeration $r_1, r_2, \ldots$ of its rules with r_1 having the highest priority. Throughout the paper, we use such an enumeration in examples and write $r_i : l \leftarrow l_1, \ldots, l_n, not\, l_{n+1}, \ldots, not\, l_m$ to denote the ith rule in such an enumeration.

Let us next look at the BE preferred answer set semantics. We will refer to preferred answers sets under the BE semantics as BE-preferred answer sets. These definitions are simplified versions of those in [6] as we focus in this work on propositional programs.

Definition 5. *Let $\mathcal{P} = (P, <)$ be a fully prioritized elp where P is a set of n prerequisite-free rules and let S be a set of literals. The sequence of sets $S_0, S_1, \ldots, S_n$ is defined as follows: $S_0 = \emptyset$ and for $0 < i \leq n$,*

$$S_i = \begin{cases} S_{i-1}, & \text{if } r_i \text{ is defeated by } S_{i-1} \text{ or} \\ & \mathit{head}(r_i) \in S \text{ and } r_i \text{ is defeated by } S, \\ S_{i-1} \cup \{\mathit{head}(r_i)\}, & \text{otherwise.} \end{cases}$$

The set $C_{\mathcal{P}}(S)$ is defined as the smallest set of literals such that $S_n \subseteq C_{\mathcal{P}}(S)$ and $C_{\mathcal{P}}(S)$ is logically closed (consistent or equal to Lit).

Definition 6. *Let $\mathcal{P} = (P, <)$ be a fully prioritized elp with prerequisite-free P and let A be an answer set of P. Then A is the BE-preferred answer set of $\mathcal{P}$ iff $A = C_{\mathcal{P}}(A)$.*

For non prerequisite-free prioritized elps, a transformation is applied similar to the Gelfond-Lifschitz reduct but that produces rules without prerequisites.

Definition 7. *Let $\mathcal{P} = (P, <)$ be a fully prioritized elp and S be a set of literals. Define ${}^S\mathcal{P} = ({}^S P, {}^S{<})$ to be the fully prioritized elp such that ${}^S P$ is the set of rules obtained from P by*

1. *deleting every rule $r \in P$ s.t. body$^+(r) \not\subseteq S$, and*
2. *deleting body$^+(r)$ from every remaining rule r;*

and ${}^S{<}$ is inherited from $<$ by the mapping $f : {}^S P \mapsto P$ where $f(r')$ is the first rule in P wrt $<$ such that r' results from r by step (2) above. In other words, for every $r_1, r_2 \in P$, $r'_1 \, {}^S{<} \, r'_2$ iff $f(r'_1) < f(r'_2)$.

Definition 8. *A set A of literals is a BE-preferred answer set of a fully prioritized elp $\mathcal{P} = (P, <)$ iff A is a BE-preferred answer set of ${}^A\mathcal{P}$.*

Finally, for an arbitrary prioritized elp $\mathcal{P}$, A is a preferred answer set of $\mathcal{P}$ if A is a preferred answer set of some full prioritization of $\mathcal{P}$.

In the next section we present some examples illustrating this semantics, including programs without preferred answer sets and fully prioritized programs with multiple ones.

3 Some Examples

Here we consider some motivating examples. Some show programs that have no BE-preferred answer sets despite containing no apparent conflict and some show programs that are fully prioritized but, arguably, have too many BE-preferred answer sets. Most examples are in fact from [6] where they were recognized as problematic.

Example 1. Consider the program $\mathcal{P}_1$ with rules

$$\begin{aligned} r_1 &: c \leftarrow \mathit{not}\, b. \\ r_2 &: b \leftarrow \mathit{not}\, a. \end{aligned}$$

$\mathcal{P}_1$ has one answer set, $A = \{b\}$. Since $c \notin A$ nor is r_1 defeated by $\emptyset$, $c \in C_{\mathcal{P}_1}(A)$. Therefore this program has no BE-preferred answer sets.

Brewka and Eiter's approach to preferences is based on the view (which we follow as well) that preferences are introduced in order to "solve potential conflicts...to conclude more than in standard answer semantics" [6]. Since the rules in the above program show no apparent conflict between them and in fact the program has only one answer set, it seems reasonable that it should have a preferred answer set.

The following example shows this shortcoming even more directly, since one of the rules is a fact, i.e. does not involve defaults at all.

Example 2. Consider the program $\mathcal{P}_2$ with rules

$$\begin{aligned} r_1 &: a \leftarrow \mathit{not}\, b. \\ r_2 &: b. \end{aligned}$$

$\mathcal{P}_2$ has one answer set, $A = \{b\}$. By a similar argument as in the previous example, we have that $a \in C_{\mathcal{P}_2}(A)$ and so the program has no BE-preferred answer sets.

The above examples show that the semantics is in some sense too strong (this also holds for the other proposed selective semantics which are stronger). On the other hand, this semantics assigns multiple preferred answer sets to some programs that are fully prioritized. This means that a full prioritization is still not enough to solve all potential conflicts. Consider the following example.

Example 3. Consider the program $\mathcal{P}_3$ with rules

$$\begin{aligned} r_1 &: b \leftarrow \mathit{not}\, \neg b,\ a. \\ r_2 &: c \leftarrow \mathit{not}\, b. \\ r_3 &: a \leftarrow \mathit{not}\, c. \end{aligned}$$

This fully prioritized elp has two answer sets: $A_1 = \{c\}$ and $A_2 = \{a, b\}$. They are both BE-preferred answer sets. Consider A_1. Rule r_1 does not belong to the reduct of the program since prerequisite $a \notin A_1$. Then rule r_2 is not defeated by $\emptyset$ nor by A_1, so we get c which then defeats rule r_3 and we have $A_1 = C_{\mathcal{P}_3}(A_1)$. Now consider A_2. Prerequisite a is removed from r_1 in the reduct of the program. Then we have that rule r_1 is not defeated by $\emptyset$ nor by A_2, so we get b which then defeats rule r_2 allowing r_3 to fire. Thus we have $A_2 = C_{\mathcal{P}_3}(A_2)$.

Despite imposing a full prioritization, the conflict between rules r_1, r_3 and r_2 persists. In the next sections we develop an alternative definition that a) selects at most one preferred answer set when a full set of priorities is given and b) always assigns at least one preferred answer set to a program without negative cycles (constraints). Note that the programs in Ex. 1 and 2 do not contain constraints or any kind of conflict between rules and still do not have BE-preferred answer sets.

4 A New Definition of Preferred Answer Sets

Our proposed new definition is intuitively based on the view, following Brewka and Eiter, that priorities are used to resolve conflicts. Intuitively, it is also based on the idea of taking conflicts between rules somewhat more literally by appealing to a notion of "attack" that is to some degree inspired by argument attacks in argumentation. Here, by an attack of a rule on another we simply mean that if the attacking rule fires, it will defeat the other one. We then consider rules to be in conflict when they attack each other, as in the program:

$$\begin{aligned} a &\leftarrow \mathit{not}\, b. \\ b &\leftarrow \mathit{not}\, a. \end{aligned}$$

In the above simple program, the attacks between the rules are direct. But attacks can be indirect, involving a chain of other rules, as in the program:

$$\begin{aligned} a &\leftarrow c. \\ b &\leftarrow \mathit{not}\, a. \\ c &\leftarrow \mathit{not}\, b. \end{aligned}$$

Here, the second rule attacks the first indirectly through the third rule.

To simplify the development of our definition of preferred answer sets, we appeal to a well know unfolding operation which transforms the above program into the program:

$$\begin{aligned} a &\leftarrow \mathit{not}\, b. \\ b &\leftarrow \mathit{not}\, a. \\ c &\leftarrow \mathit{not}\, b. \end{aligned}$$

Attacks and conflicts between rules are much easier to define on unfolded programs and consequently simplifies our definition of preferred answer sets. However, we remark that for the purpose of implementation this is not a requirement and probably should be avoided in an efficient implementation. Although it would be possible to define the semantics without unfolding the program (e.g. by using argumentation inspired constructions as in [11]), this would make the definition of conflicts between rules and the overall semantics more cumbersome. That is why we opt to use unfoldings. Moreover, the answer set semantics satisfies the *Generalized Principle of Partial Evaluation* [12–14], which means that the unfolding transformation results in a program that has exactly the same answer sets as the original program. The formal definitions follow.

Definition 9 (Unfolding [12]). *Let P_i be an elp and r be a rule in P_i of the form $H \leftarrow L, \Gamma$ where L is a literal different from H and Γ is the rest of the rule's body. Suppose that $r_1, \ldots, r_k$ are all the rules in P_i s.t. each r_j is of the form $L \leftarrow \Gamma_j$ s.t. $L \notin body^+(r_j)$. Then*

$$P_{i+1} = (P_i \setminus \{r\}) \cup \{H \leftarrow \Gamma_j, \Gamma \,:\, 1 \leq j \leq k\}.$$

This operation is called unfolding *r in P_i and r is called the* unfolded *rule.*

Let us define the unfolding operation for prioritized elps. For our purposes it suffices to define it for fully prioritized elps.

Definition 10 (Unfolding fully prioritzed elps). *We say $\mathcal{P}_{i+1} = (P_{i+1}, <_{i+1})$ is the result of applying an* unfolding *on $\mathcal{P}_i = (P_i, <_i)$ if*

1. *P_{i+1} is the result of unfolding some $r \in P_i$ s.t. r is replaced by rules $r'_1, \dots, r'_k$,*
2. *for each rule r'_j obtained in the previous step, if $r'_j \in P_i$, i.e. an identical rule was already in the program, then*
   - *(a) if $r'_j <_i r$ then let $r'_j <_{i+1} r^*$ (resp. $r^* <_{i+1} r'_j$) for every rule r^* s.t. $r'_j <_i r^*$ (resp. $r^* <_i r'_j$), i.e. r'_j retains the same priority, since it has higher priority in $\mathcal{P}_i$ than the unfolded rule r.*
   - *(b) if $r <_i r'_j$ then let $r'_j <_{i+1} r^*$ (resp. $r^* <_{i+1} r'_j$) for every rule r^* s.t. $r <_i r^*$ (resp. $r^* <_i r$), i.e. r'_j now has the same priority r has in $\mathcal{P}_i$, since r had higher priority.*
3. *for each rule r'_j obtained in step (1) s.t. $r'_j \notin P_i$, i.e. it is a new rule, $<_{i+1}$ extends $<_i$ with the priorities $r'_j <_{i+1} r^*$ (resp. $r^* <_{i+1} r'_j$) if $r <_i r^*$ (resp. $r^* <_i r$), i.e. these new rules are assigned the same priority r has in $\mathcal{P}_i$.*

It is easy to see that applying an unfolding operation results in a fully prioritized elp.

Definition 11. *A transformation sequence is a sequence of fully prioritized elps $\mathcal{P}_0, \dots, \mathcal{P}_n$ such that each $\mathcal{P}_{i+1}$ is obtained by applying an unfolding operation on $\mathcal{P}_i$.*

Definition 12. *The unfolding of a fully prioritized elp $\mathcal{P}$, denoted $\overline{\mathcal{P}}$, is the fully prioritized elp $\mathcal{P}_n$ such that there is a transformation sequence $\mathcal{P}_0, \dots, \mathcal{P}_n$ where $\mathcal{P}_0 = \mathcal{P}$ and there is no rule in P_n that can be unfolded.*

Example 4. Consider again the program of Ex. 3. The unfolding of that program consists of the following rules:

$$\begin{array}{rcl} r'_1 : b & \leftarrow & not\, \neg b,\ not\, c. \\ r_2 : c & \leftarrow & not\, b. \\ r_3 : a & \leftarrow & not\, c. \end{array}$$

The unfolding helps to reveal more directly that there is a conflict between r_1, r_2: after unfolding, the head of one rule appears negated in the body of the other and vice-versa.

Let us now proceed with our definition of preferred answer sets, starting with unfolded, fully prioritized elps. We need some terminology.

Let P be an elp and X be a set of literals. We say a *literal l holds in X* if $l \in X$. An extended literal $not\, l$ is *defeated by X* if l holds in X. A *rule r is defeated by X* if there is a literal $l \in body^-(r)$ such that $not\, l$ is defeated by X. An extended literal *$not\, l$ holds in X* (wrt P) if $\bar{l}$ holds in X or every rule $r \in P$, if any, whose $head(r) = l$ is defeated by X. (Note it is possible for $not\, l$ not to hold nor be defeated in X.) For a rule r, the *body holds in X* if $body^+(r)$ holds in X and $\underline{not\, l}$ holds in X for each $l \in body^-(r)$. A rule r is *active in X* if neither $head(r)$ nor $\overline{head(r)}$ holds in, $body^+(r)$ holds in and r is not defeated by, X.(A similar notion of "active rule" is used in [10].)

We also need to define the closure of a set of literals X under an elp P: $close(X, P)$ is the smallest set X', $X \subseteq X'$, s.t. for every rule $r \in P$, if $body(r)$ holds in X' then $head(r) \in X'$. For instance, for the program P_1 of Ex. 1 we have $close(\emptyset, P_1) = \{b\}$.

We define attacks on a rule r wrt a set of literals as follows: a *rule r' attacks r wrt X* if r' is active in X and $head(r') \in body^-(r)$. We will omit the mention of X when clear from context and also use the notation $r' \rightarrow r$ for attacks.

Next we define a notion of "conflict" and how conflicts may be resolved by taking the priorities into account. A set of rules $\{r_1, \ldots, r_k\}$ is a *conflict* wrt X if there is a sequence of attacks $r_1 \rightarrow r_2$, $r_2 \rightarrow r_3$, $\ldots$, $r_{k-1} \rightarrow r_k$, $r_k \rightarrow r_1$. For a rule r in a conflict $F = \{r_1, \ldots, r_k\}$, we define two sets, $PR_F(r)$ and $CR_F(r)$, called *pro rules* and *con rules* resp., that partition F as follows: If $r \rightarrow r'$ then $r' \in CR_F(r)$. For all other attacks $r_i \rightarrow r_j$ of conflict F, if $r_i \in PR_F(r)$ then $r_j \in CR_F(r)$ and if $r_i \in CR_F(r)$ then $r_j \in PR_F(r)$. (We may omit the subscripts in PR_F, CR_F when the relevant conflict is clear.) For instance, for a conflict involving attacks $r_1 \rightarrow r_2$, $r_2 \rightarrow r_3$, $r_3 \rightarrow r_1$, the pro rules of r_1 are $PR(r_1) = \{r_1, r_3\}$ and the con rules are $CR(r_1) = \{r_1, r_2\}$. A conflict with an additional rule as follows: $r_1 \rightarrow r_2$, $r_2 \rightarrow r_3$, $r_3 \rightarrow r_4$, $r_4 \rightarrow r_1$, yields $PR(r_1) = \{r_1, r_3\}$ and $CR(r_1) = \{r_2, r_4\}$.

The following definition describes how to compute a sequence of sets that leads to a preferred answer set. It includes the conditions (item b) under which the preferences can be used to break a conflict in favor of one of the rules in it. Intuitively, the conditions are: the rule should be the highest priority rule in the conflict, if any of the pro rules is attacked, the attacking rule is also in the conflict, and the rule is not one of the con rules itself.

Definition 13. *Let $\mathcal{P} = (P, <)$ be an unfolded, fully prioritized elp. We define the sequence $X_0, X_1, \ldots$ as follows.*

1. $X_0 = \emptyset$.
2. X_{i+1} *is recursively defined as follows. Let $X' = close(X_i, P)$.*
   (a) *If $X' \neq X_i$ then $X_{i+1} = X'$.*
   (b) *If $X' = X_i$, then:*
   let r be the highest priority rule active in X_i such that there exists a conflict F satisfying the following conditions:
   i. $r \in F$;
   ii. $r < r'$ *for every* $r' \neq r$ *in* F;
   iii. *for every* $r^+ \in PR_F(r)$, *if* $r' \rightarrow r^+$ *then* $r' \in F$;
   iv. $r \notin CR_F(r)$.
   Then, $X_{i+1} = X_i \cup heads(PR_F(r))$.

Intuitively, this definition can be understood as an algorithm which in each iteration: computes the closure of X_i over P adding the heads of all rules that hold wrt X_i. If no new literals are added in this step, the algorithm then finds the highest priority rule r included in a conflict which also includes any rules attacking the pro rules. If it exists, it can be found by first looking for a conflict that includes all rules attacking r. If r is not a con rule itself (iv), we add the heads of the pro rules, which include r. Steps (a) and (b) can be done in the same iteration, but it simplifies the proofs to keep them separate.

Proposition 1. *There exists n s.t. for all $m > n$, $X_m = X_n$, i.e., the sequence reaches a fixpoint.*

Let $I_{\mathcal{P}}$ denote the fixpoint X_n if it does not contain contrary literals, and Lit otherwise.

Definition 14. *Let $\mathcal{P}$ be an unfolded, fully prioritized elp. The set $I_{\mathcal{P}}$ is a preferred answer set of $\mathcal{P}$ if $I_{\mathcal{P}}$ is an answer set of $\mathcal{P}$.*

It trivially follows from this definition that all preferred answer sets are answer sets. It is also easy to see that if $\mathcal{P}$ has a preferred answer set at all, it has exactly one: $I_{\mathcal{P}}$.

The computation of $I_{\mathcal{P}}$ may fail to produce one of the answer sets of the program, hence the need to test afterwards whether it is one. For some programs the computation may reach a fixpoint prematurely. Later we show that for negative-cycle-free, head-consistent programs, this computation is guaranteed to produce one of the answer sets. For this class of programs, after computing $I_{\mathcal{P}}$ it is not necessary to check if it is an answer set.

For an arbitrary prioritized elp, preferred answer sets are defined as follows.

Definition 15 (Preferred answer sets). *For an arbitrary prioritized elp $\mathcal{P}$, A is a preferred answer set of $\mathcal{P}$ if A is a preferred answer set of the unfolding $\overline{\mathcal{P}'}$ of one of the full prioritizations $\mathcal{P}'$ of $\mathcal{P}$.*

The set of all preferred answer sets of $\mathcal{P}$ will be denoted by $PAS(\mathcal{P})$.

Since, given a prioritized elp, we need to consider its full prioritizations, in our examples below we use fully prioritized programs.

Example 5. Consider the program P_1 from Ex. 1:

$$\begin{array}{l} r_1 : c \leftarrow \mathit{not}\, b. \\ r_2 : b \leftarrow \mathit{not}\, a. \end{array}$$

As described earlier, $\mathit{close}(X_0, P_1) = \{b\}$. So $X_1 = \{b\}$ which is the fixpoint. Since $\{b\}$ is also an answer set, it is the preferred answer set.

Example 6. Consider next the program P_3' from Ex. 3 and 4:

$$\begin{array}{l} r_1' : b \leftarrow \mathit{not}\, \neg b,\ \mathit{not}\, c. \\ r_2 : c \leftarrow \mathit{not}\, b. \\ r_3 : a \leftarrow \mathit{not}\, c. \end{array}$$

First note that $\mathit{close}(X_0, P_3') = \emptyset$. We then consider conflicts including r_1' and find $F = \{r_1', r_2\}$ with $PR(r_1') = \{r_1'\}$ and $CR(r_1') = \{r_2\}$. The only rule attacking r_1' is r_2 and $r_1' \notin CR(r_1')$. We obtain $X_1 = \{b\}$ and then $X_2 = \mathit{close}(X_1, P_3') = \{a, b\}$. This is the fixpoint and it is an answer set. Therefore it is the preferred answer set.

The following program includes a constraint and has no preferred answer sets under our definition nor under the other selective semantics.

Example 7. Consider the following program P_4:

$$\begin{array}{l} r_1 : p \leftarrow \mathit{not}\, p,\ \mathit{not}\, b. \\ r_2 : a \leftarrow \mathit{not}\, b. \\ r_3 : b \leftarrow \mathit{not}\, a. \end{array}$$

Clearly, $close(X_0, P_4) = \emptyset$. Rule r_1 is self attacking, i.e., $r_1 \in CR(r_1)$ in any conflict that includes it, so it is never considered in step (2b). In the case of r_2, there is conflict $F = \{r_2, r_3\}$ which includes r_2 and its only attacker r_3. We get $X_1 = \{a\}$, which is the fixpoint. However, $\{a\}$ is not an answer set and therefore not a preferred answer set. This program then has an answer set, $\{b\}$, but no preferred answer sets.

This program essentially has two parts, one part consisting of rules r_2, r_3 which intuitively generates a choice between a and b, and constraint r_1 which eliminates answer sets that contain a. But the priorities on the bottom two rules say to prefer a, which conflicts with the constraint. Note that if the priorities on r_2, r_3 are relaxed, i.e. the priority $r_2 < r_3$ is removed, then $\{b\}$ is a preferred answer set. It can be argued that the information encoded in the priorities and the constraint is in some sense contradictory. The priorities eliminate one answer set in favor of another and the constraint does the same for a different answer set. In the following section we show that prioritized elps without constraints always have preferred answer sets.

5 Properties

Here we present some formal properties of our definition of preferred answer sets. We start by showing that prioritized elps are a conservative extension of elps.

Given an elp P, a set of literals S is said to be *generated by* R if R is the set of all the rules $r \in P$ whose bodies are satisfied by S and $head(r) \in S$. We also say in this case that R is the set of *generating rules* of S.

Theorem 1. *Let $\mathcal{P} = (P, <)$ be a prioritized elp with empty $<$, i.e., without priorities. Then $AS(\mathcal{P}) = PAS(\mathcal{P})$.*

Example 7 shows a prioritized program that combines constraints and priorities with regular rules and has no preferred answer sets. It is well known in logic programming that for negative-cycle-free programs an answer set always exists. A reasonable question to ask is whether there is a similar class of prioritized programs for which a preferred answer set always exists. We show next that in fact negative-cycle-free prioritized programs, with the additional but less critical condition that the programs be head-consistent, are guaranteed to have a preferred answer set.

An elp P is said to be *head-consistent* if the set of literals $\{head(r) : r \in P\}$ is consistent, i.e. does not contain contrary literals. It is said to be *negative-cycle-free* if the dependency graph of P does not contain cycles with an odd number of negative edges.

Theorem 2. *Let $\mathcal{P} = (P, <)$ be a fully prioritized elp s.t. P is negative-cycle-free and head-consistent. Then $\mathcal{P}$ has a preferred answer set.*

Corollary 1. *If $\mathcal{P}$ is a fully prioritized elp with negative-cycle-free and head-consistent P, then the set $I_{\overline{\mathcal{P}}}$ is the preferred answer set.*

The main point of this corollary is that for a prioritized elp $\mathcal{P}$ with negative-cycle-free, head-consistent program, the computation of the sequence $X_0, \ldots, X_n$ of Def. 13 is guaranteed to produce an answer set, making the check stipulated in Def. 14 unnecessary.

In [6] one can find multiple examples illustrating how the BE semantics overcomes some of the shortcomings of previous approaches. Based on their study of these shortcomings and the development of their semantics, Brewka and Eiter proposed two principles that ought to be satisfied by any system based on prioritized defeasible rules. We reproduced them here in their particular form for prioritized elps.

The first principle specifies a requirement on the treatment of $<$ as a preference relation.

Principle 1. *Let A_1, A_2 be two answer sets of a prioritized elp $\mathcal{P} = (P, <)$ generated by the rules $R \cup \{r_1\}$ and $R \cup \{r_2\}$, respectively, where $r_1, r_2 \notin R$. If $r_1 < r_2$ then A_2 is not a preferred answer set of $\mathcal{P}$.*

The second principle is about relevance. It says that A should remain a preferred answer set after adding a rule with a prerequisite not in A while keeping all preferences intact.

Principle 2. *Let A be a preferred answer set of $(P, <)$ and r be a rule such that at least one prerequisite of r is not in A. Then A is a preferred answer set of $(P \cup \{r\}, <')$ whenever $<'$ agrees with $<$ on the rules in P.*

Next we show that our definition of preferred answer sets satisfies the first principle for unfolded programs.

Theorem 3. *The preferred answer set semantics based on Def. 15 satisfies Principle 1.*

Principle 2, which is about relevance, is not satisfied by our definition. This can be shown using the program from Ex. 3.

Example 8. Consider again the program $\mathcal{P}_3$ from Ex. 3:

$$\begin{array}{rl} r_1: & b \leftarrow \mathit{not}\,\neg b,\ a. \\ r_2: & c \leftarrow \mathit{not}\, b. \\ r_3: & a \leftarrow \mathit{not}\, c. \end{array}$$

Consider the program $\mathcal{P}_3'$ consisting only of the rules r_2, r_3 with $r_2 < r_3$. This program has one answer set $A_1 = \{c\}$ which is also preferred according to our semantics. The full program $\mathcal{P}_3$ has two answer sets, A_1 and $A_2 = \{a, b\}$ which is the preferred answer set. $\mathcal{P}_3'$ has no BE-preferred answer sets while for $\mathcal{P}_3$ both A_1, A_2 are BE-preferred.

According to Principle 2, $\{c\}$ should remain a preferred answer set because r_1 is not applicable in A_1 (not relevant). But in terms of attacks, r_1 seems relevant since it can attack r_2 and has higher priority. Moreover, consider replacing r_1 with its unfolded version $b \leftarrow \mathit{not}\,\neg b,\ \mathit{not}\, c$. The result is a program that is equivalent. However, in the case of adding this unfolded rule to $\mathcal{P}_3'$, Principle 2 no longer says anything about it. It does not apply. This is because it defines relevance in terms of prerequisites (literals in body^+). In other words, by applying the unfolding operation it is possible to sidestep Principle 2, even though unfolding does not affect a program's answer sets.

The example above also shows that satisfying Principle 2 necessarily requires that some programs have no preferred answer sets or have multiple ones. In the above example, $\mathcal{P}_3'$ has one answer set, $\{c\}$, which is not the same as the intuitively preferred answer set of $\mathcal{P}_3$. But Principle 2 requires that if $\{c\}$ is a preferred answer set, it must remain one after adding r_1. In fact, Brewka and Eiter show in [6] that any semantics that is consistent, i.e. that selects a preferred answer set whenever there are answer sets, is incompatible with Principle 2.

It is worth mentioning that Brewka and Eiter [6] define another semantics, called *weakly preferred semantics*, which assigns preferred answer sets to programs that do not have one under the BE semantics. However, this semantics is based on quantitative measures of relative satisfaction of the preferences, which is very different to the style of semantics we propose here and of the BE semantics. Furthermore, the weakly preferred semantics does not satisfy either of the two principles.

6 Conclusions

We have proposed a new definition of preferred answer sets. Under our definition, at most one preferred answer set is selected for programs that are fully prioritized. As in the case of standard answer sets, we showed that for negative-cycle-free programs the existence of preferred answer sets is guaranteed. We have also shown that our semantics captures the intended meaning of preferences as postulated by Principle 1 from [6].

In the literature about preferences in non-monotonic formalisms, the various approaches have been classified into two categories: *prescriptive* and *descriptive* (see e.g. [15, 16]). Roughly, prescriptive approaches take preferences as an indication of the order in which rules are to be applied, i.e. higher priority rules are to be applied before lower priority rules. On the other hand, descriptive approaches take preferences as specifying the "desirability" of rules. It is said that the descriptive approaches take a "global view" of preferences because they allow lower priority rules to be applied if doing so enables a higher priority rule that would otherwise be inapplicable.

In terms of these two categories, we find that our approach has features of both. Preferred answer sets are essentially computed by attempting to apply the rules in the order specified by the priorities. Also, similar to some prescriptive approaches, preferred answer sets here are not computed by first computing a standard answer set and then checking if it is preferred. On the other hand, similar to the descriptive approaches, a high priority rule may be suspended if it is attacked by a lower priority rule and it does not "counter-attack." If the attacker turns out to be defeated, the rule may fire after all. This kind of interaction between rules is the characteristic "global view" in descriptive approaches. Our approach also shares with descriptive approaches the characteristic of not satisfying Brewka and Eiter's Principle 2. However, as mentioned after Example 8, it seems that in the context of logic programming there are other reasons why one may not accept it in its current form. For a more thorough discussion of the descriptive vs prescriptive distinction, see [7, 15].

In future work we would like to investigate a semantics with the same properties as we have presented here, but which does not require programs to be unfolded. We intend to look for other classes of programs for which existence of preferred answer sets can be guaranteed, e.g. based on [17]. We also plan to compare our approach with that of [11]. It would also be interesting to look at the relationship between prioritized logic programs as described here and approaches to updating logic programs such as [18].

References

1. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9(3-4), 365–386 (1991)
2. Gelfond, M.: Answer sets. In: van Harmelen, F., Lifschitz, V., Porter, B. (eds.) Handbook of Knowledge Representation, pp. 285–316. Elsevier (2008)
3. Sakama, C., Inoue, K.: Prioritized logic programming and its application to commonsense reasoning. Artificial Intelligence 123(1-2), 185–222 (2000)
4. Gelfond, M., Son, T.C.: Reasoning with Prioritized Defaults. In: Dix, J., Moniz Pereira, L., Przymusinski, T.C. (eds.) LPKR 1997. LNCS (LNAI), vol. 1471, pp. 164–223. Springer, Heidelberg (1998)
5. Zhang, Y., Foo, N.Y.: Answer sets for prioritized logic programs. In: Maluszynski, J. (ed.) International Symposium on Logic Programming (ILPS 1997), pp. 69–83 (1997)
6. Brewka, G., Eiter, T.: Preferred answer sets for extended logic programs. Artificial Intelligence 109(1-2), 297–356 (1999)
7. Delgrande, J.P., Schaub, T., Tompits, H.: Logic programs with compiled preferences. In: Horn, W. (ed.) 14th European Conference on Artificial Intelligence (ECAI 2000), pp. 464–468 (2000)
8. Wang, K., Zhou, L.-z., Lin, F.: Alternating Fixpoint Theory for Logic Programs with Priority. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS (LNAI), vol. 1861, pp. 164–178. Springer, Heidelberg (2000)
9. Baral, C.: Knowledge representation, reasoning and declarative problem solving. Cambridge University Press (2003)
10. Schaub, T., Wang, K.: A comparative study of logic programs with preference. In: Nebel, B. (ed.) 17th International Joint Conference on Artificial Intelligence (IJCAI 2001), pp. 597–602 (2001)
11. Sefranek, J., Simko, A.: Warranted derivations of preferred answer sets. In: 25th Workshop on Logic Programming (2011)
12. Aravindan, C., Dung, P.M.: On the correctness of unfold/fold transformation of normal and extended logic programs. Journal of Logic Programming 24(3), 201–217 (1995)
13. Dix, J.: A classification theory of semantics of normal logic programs: II. Weak properties. Fundamenta Informaticae 22(3), 257–288 (1995)
14. Brass, S., Dix, J.: Semantics of (disjunctive) logic programs based on partial evaluation. Journal of Logic Programming 40(1), 1–46 (1999)
15. Delgrande, J.P., Schaub, T., Tompits, H., Wang, K.: A classification and survey of preference handling approaches in nonmonotonic reasoning. Computational Intelligence 20(2), 308–334 (2004)
16. Brewka, G.: Adding Priorities and Specificity to Default Logic. In: MacNish, C., Moniz Pereira, L., Pearce, D.J. (eds.) JELIA 1994. LNCS (LNAI), vol. 838, pp. 247–260. Springer, Heidelberg (1994)
17. Costantini, S.: On the existence of stable models of non-stratified logic programs. Theory and Practice of Logic Programming 6(1-2), 169–212 (2006)
18. Alferes, J.J., Brogi, A., Leite, J., Moniz Pereira, L.: Evolving Logic Programs. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 50–62. Springer, Heidelberg (2002)

A Minimal Model Semantics for Nonmonotonic Reasoning

Laura Giordano[1], Valentina Gliozzi[2], Nicola Olivetti[3], and Gian Luca Pozzato[2]

[1] DISIT, Universitá del Piemonte Orientale, Alessandria, Italy, Alessandria, Italy
laura@mfn.unipmn.it
[2] Dipartimento di Informatica, Università degli Studi di Torino, Italy
{gliozzi,pozzato}@di.unito.it
[3] Aix-Marseille Université, CNRS, LSIS UMR 7296, 13397, Marseille, France
nicola.olivetti@univ-cezanne.fr

Abstract. This paper provides a general semantic framework for nonmonotonic reasoning, based on a minimal models semantics on the top of KLM systems for nonmonotonic reasoning. This general framework can be instantiated in order to provide a semantic reconstruction within modal logic of the notion of rational closure, introduced by Lehmann and Magidor. We give two characterizations of rational closure: the first one in terms of minimal models where propositional interpretations associated to worlds are fixed along minimization, the second one where they are allowed to vary. In both cases a knowledge base must be expanded with a suitable set of consistency assumptions, represented by negated conditionals. The correspondence between rational closure and minimal model semantics suggests the possibility of defining variants of rational closure by changing either the underlying modal logic or the comparison relation on models.

1 Introduction

In a seminal work Kraus Lehmann and Magidor [7] (henceforth KLM) proposed an axiomatic approach to nonmonotonic reasoning. Plausible inferences are represented by nonmonotonic conditionals of the form $A \mid\sim B$, to be read as "typically or normally A entails B": for instance $monday \mid\sim go\_work$, "normally on Monday I go to work". The conditional is nonmonotonic since from $A \mid\sim B$ one cannot derive $A \wedge C \mid\sim B$, in our example, one cannot derive $monday \wedge ill \mid\sim go\_work$. KLM proposed a hierarchy of four systems, from the weakest to the strongest: cumulative logic **C**, loop-cumulative logic **CL**, preferential logic **P** and rational logic **R**. Each system is characterized by a set of postulates expressing natural properties of nonmonotonic inference. We present below an axiomatization of the two stronger logics **P** and **R** (**C** and **CL** being too weak to be taken as an axiomatic base for nonmonotonic reasoning). But before presenting the axiomatization, let us clarify one point: in the original presentation of KLM systems, [7] a conditional $A \mid\sim B$ is considered as a consequence relation between a pair of formulas A and B, so that their systems provide a set of "postulates" (or closure conditions) that the intended consequence relations must satisfy. Alternatively, these postulates may be seen as *rules* to derive new conditionals from given ones. We take a slightly different viewpoint, shared among others by Halpern and Friedman [4] (see

Section 8) and Boutilier [2] who proposed a modal interpretation of KLM systems **P** and **R**: in our understanding these systems are ordinary logical systems in which a conditional $A \mid\!\sim B$ is a propositional formula belonging to the object language. Whenever we restrict our consideration, as done by Kraus Lehmann and Magidor, to the entailment of a conditional from a set of conditionals, the two viewpoints *coincide*: a conditional is a logical consequence in logic **P/R** of a set of conditionals if and only if it belongs to all preferential/rational consequence relations extending that set of conditionals, or (in semantic terms), it is valid in all preferential/rational models (as defined by KLM) of that set. Here is the axiomatization of logics **P** and **R**, in our presentation KLM postulates/rules are just *axioms*. We use $\vdash_{PC}$ (resp. $\models_{PC}$) to denote provability (resp. validity) in the propositional calculus.

All axioms and rules of propositional logic	
$A \mid\!\sim A$	(REF)
if $\vdash_{PC} A \leftrightarrow B$ then $(A \mid\!\sim C) \rightarrow (B \mid\!\sim C)$	(LLE)
if $\vdash_{PC} A \rightarrow B$ then $(C \mid\!\sim A) \rightarrow (C \mid\!\sim B)$	(RW)
$((A \mid\!\sim B) \wedge (A \mid\!\sim C)) \rightarrow (A \wedge B \mid\!\sim C)$	(CM)
$((A \mid\!\sim B) \wedge (A \mid\!\sim C)) \rightarrow (A \mid\!\sim B \wedge C)$	(AND)
$((A \mid\!\sim C) \wedge (B \mid\!\sim C)) \rightarrow (A \vee B \mid\!\sim C)$	(OR)
$((A \mid\!\sim B) \wedge \neg(A \mid\!\sim \neg C)) \rightarrow (A \wedge C) \mid\!\sim B)$	(RM)

The axiom (CM) is called cumulative monotony and it is characteristic of all KLM logics, axiom (RM) is called rational monotony and it characterizes the logic of rational entailment **R**. The weaker logic of preferential entailment **P** contains all axioms, but (RM). **P** and **R** seem to capture the core properties of nonmonotonic reasoning, as shown in [4] they are quite ubiquitous being characterized by different semantics (all of them being instances of so-called plausibility structures).

Logics **P** and **R** enjoy a very simple modal semantics, actually it turns out that they are the flat fragment of some well-known conditional logics. For **P** the modal semantics is defined by considering a set of worlds W equipped by an accessibility (or preference) relation $<$ assumed to be transitive, irreflexive, and satisfying the so-called Smoothness Condition. For the stronger **R** $<$ is further assumed to be modular. Intuitively the meaning of $x < y$ is that x is more typical/more normal/less exceptional than y. We say that $A \mid\!\sim B$ is true in a model if B holds in all most normal worlds where A is true, i.e. in all $<$-minimal worlds satisfying A.

KLM systems formalize desired properties of nonmonotonic inference. However, they are too weak to perform useful nonmonotonic inferences. For instance KLM systems cannot handle irrelevant information in conditionals: from $monday \mid\!\sim go\_work$, there is no way of concluding $monday \ \wedge \ shines \mid\!\sim go\_work$ in any one of KLM systems. Lehmann and Magidor in [8] look for a plausible definition of the set of conditional assertions entailed by a conditional knowledge base. They argue that such a set of assertions must be rational and they propose a true nonmonotonic construction on the top of preferential logic called *rational closure*. Rational closure, besides satisfying the postulates of **R**, allows one to perform some truthful nonmonotonic inferences, like the one just mentioned ($monday \wedge shines \mid\!\sim go\_work$).[1] The authors has given a syntactic procedure to calculate the set of conditionals entailed by the rational closure as well as a quite complex semantic construction. It is worth noticing that a strongly related construction has been proposed by Pearl [9] with his notion of 1-entailment, motivated by a probabilistic interpretation of conditionals.

In this work we tackle the problem of giving a purely semantic characterization of rational closure, stemming directly from the modal semantics of logic **R**. Notice that we restrict our attention to finite knowledge bases. More precisely, we try to answer to the following question: given the fact that logic **R** is characterized by a specific class of Kripke models, how can we characterize the Kripke models of the rational closure of a set of positive conditionals?

The characterization we propose may be seen as an instance of a general recipe for defining nonmonotonic inference: (i) fix an underlying modal semantics for conditionals (such as the one of **P** or **R**), (ii) obtain nonmonotonic inference by restricting semantic consequence to a class of "minimal" models according to some preference relation on models. The preference relation in itself is defined independently from the *language* and from the *set of conditionals* (knowledge base) whose nonmonotonic consequences we want to determine. In this respect our approach is similar in spirit to "minimal models" approaches to nonmonotonic reasoning, such as circumscription.

The general recipe for defining nonmonotonic inference we have sketched may have a wider interest than that of capturing rational closure. First of all, we may think of studying variants of rational closure based on other modal logics and/or on other comparison relations on models. Secondly, being a purely semantic approach (the preference relation is independent from the language), our semantics can cope with a larger language than the one considered in KLM framework. To this regard, already in this paper, we consider a richer language allowing boolean combinations of conditionals[2]. In the future, we may think of applying our semantics to Nonmonotonic Description Logics, where an extension of rational closure has been recently considered [3].

In any case, the quest of a modal characterization of rational closure turns out to be harder than expected. Our semantic account is based on the minimization of the *height* of worlds in models, where the height of a world is defined in terms of length of the $<$-chains starting from the world. Intuitively, the lower the height of a world, the more normal (or less exceptional) is the world and our minimization corresponds intuitively to the idea of minimizing less-normal or less-plausible worlds (or maximizing most plausible ones). We begin by considering the nonmonotonic inference relation determined by restricting considerations to models which minimize the *height of worlds*. In this first characterization we keep fixed the propositional interpretation associated to worlds. The

[1] From a technical point of view, as shown in [8] , it turns out that the intersection of all rational consequence relations satisfying a set of conditionals coincides with the least *preferential* consequence relation satisfying that set, so that (i) the axiom/rule (RM) does not add anything and (ii) such relation in itself *fails* to satisfy (RM). The notion of rational closure provides a solution to both problems and can be seen as the "minimal" (in some sense) rational consequence completing a set of conditionals.

[2] An extension of rational closure to knowledge bases comprising both positive and negative conditionals has been proposed in [1].

consequence relation makes sense in its own, but as we show it is *strictly weaker* than rational closure. We can obtain nonetheless a first characterization of rational closure if we further restrict attention to minimal *canonical models* that is to say, to models that contain all propositional interpretations compatible with the knowledge base K (i.e. all propositional interpretations except those that satisfy some formulas inconsistent with the knowledge base K). Restricting attention to canonical models amounts to expanding K by all formulas $\neg(A \mid\sim \bot)$ (read as "A is possible", as it encodes S5 $\Diamond A$) for all formulas A such that $K \not\models_R A \mid\sim \bot$. We thus obtain a simple and neat characterization of rational closure, but at the price of an *exponential* increase of the knowledge base K.

We then propose a second characterization that does not entail this exponential blow up. In analogy with circumscription, we consider a stronger form of minimization where we minimize the height of worlds, but *we allow to vary* the propositional interpretation associated to worlds. Again the resulting minimal consequence relation makes sense in its own, but as we show it still does not correspond to rational closure. In order to capture rational closure, we must basically add the assumption that there are "enough" worlds to satisfy all conditionals consistent with the knowledge base K. This amounts to adding a *small* set of consistency assumptions (represented by negative conditionals). In this way we capture exactly rational closure, without an exponential increase of K.

Due to space limitations, we put the main proofs in the accompanying paper [6].

2 General Semantics

In KLM framework the language of both logics **P** and **R** consists only of conditionals $A \mid\sim B$. We consider here a richer language allowing boolean combinations of conditionals (and propositional formulas). Our language $\mathcal{L}$ is defined from a set of propositional variables ATM. We use $A, B, C, \ldots$ to denote propositional formulas (not containing $\mid\sim$), and $F, G, \ldots$ to denote arbitrary formulas. More precisely, the formulas of $\mathcal{L}$ are defined as follows: if A and B are propositional formulas, $A \mid\sim B \in \mathcal{L}$; if F is a boolean combination of formulas of $\mathcal{L}$, $F \in \mathcal{L}$. A knowledge base K is any set of formulas in $\mathcal{L}$: as already mentioned, in this work we restrict our attention to finite knowledge bases.

The semantics of **P** and **R** is defined respectively in terms of preferential and rational[3] models, that are possible world structures equipped with a preference relation $<$, intuitively $x < y$ means that the world/individual x is *more normal/ more typical* than the world/individual y. The intuitive idea is that $A \mid\sim B$ holds in a model if the most typical/normal worlds/individuals satisfying A satisfy also B. Preferential models presented in [7] characterize the system **P**, whereas the more restricted class of rational models characterizes the system **R** [8].

Definition 1. *A* preferential *model is a triple $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$ where: $\mathcal{W}$ is a non-empty set of items; $<$ is an irreflexive, transitive relation on $\mathcal{W}$ satisfying the Smoothness relation defined below; V is a function $V : \mathcal{W} \longmapsto 2^{ATM}$, which assigns to every world w the set of atoms holding in that world. If F is a boolean combination of formulas, its truth conditions ($\mathcal{M}, w \models F$) are defined as for propositional logic.*

[3] We use the expression "rational model" rather than "ranked model" which is also used in the literature in order to avoid any confusion with the notion of rank used in rational closure.

Let A be a propositional formula; we define $Min^{\mathcal{M}}_{<}(A) = \{w \in \mathcal{W} \mid \mathcal{M}, w \models A$ and $\forall w'$, $w' < w$ implies $\mathcal{M}, w' \not\models A\}$. We also define $\mathcal{M}, w \models A \mathrel{|\!\sim} B$ if for all w', if $w' \in Min^{\mathcal{M}}_{<}(A)$ then $\mathcal{M}, w' \models B$. Last we define the Smoothness Condition*: if $\mathcal{M}, w \models A$, then $w \in Min^{\mathcal{M}}_{<}(A)$ or there is $w' \in Min^{\mathcal{M}}_{<}(A)$ such that $w' < w$. Validity and satisfiability of a formula are defined as usual. Given a set of formulas K of $\mathcal{L}$ and a model $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$, we say that $\mathcal{M}$ is a model of K, written $\mathcal{M} \models K$, if, for every $F \in K$, and every $w \in \mathcal{W}$, we have that $\mathcal{M}, w \models F$. K preferentially entails a formula F, written $K \models_P F$ if F is valid in all preferential models of K.*

Since we limit our attention to finite knowledge bases, we can restrict our attention to finite models, as the logic enjoys the finite model property (observe that in this case the smoothness condition is ensured trivially by the irreflexivity of the preference relation). From Definition 1, we have that the truth condition of $A \mathrel{|\!\sim} B$ is "global" in a model $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$: given a world w, we have that $\mathcal{M}, w \models A \mathrel{|\!\sim} B$ if, for all w', if $w' \in Min^{\mathcal{M}}_{<}(A)$ then $\mathcal{M}, w' \models B$. It immediately follows that $A \mathrel{|\!\sim} B$ holds in w if only if $A \mathrel{|\!\sim} B$ is valid in a model, i.e. it holds that $\mathcal{M}, w' \models A \mathrel{|\!\sim} B$ for all w' in $\mathcal{W}$; for this reason we will often write $\mathcal{M} \models A \mathrel{|\!\sim} B$. Moreover, when the reference to the model $\mathcal{M}$ is unambiguous, we will simply write $Min_{<}(A)$ instead of $Min^{\mathcal{M}}_{<}(A)$.

Definition 2. *A* rational *model is a preferential model in which $<$ is further assumed to be* modular*: for all $x, y, z \in \mathcal{W}$, if $x < y$ then either $x < z$ or $z < y$. K rationally entails a formula F, written $K \models_R F$ if F is valid in all rational models of K.*

When the logic is clear from the context we shall write $K \models F$ instead of $K \models_P F$ or $K \models_R F$. From now on, we restrict our attention to *rational models*.

Definition 3. *The height $k_{\mathcal{M}}(w)$ of a world w in $\mathcal{M}$ is the length of the longest chain $w_0 < \ldots < w$ from w to a w_0 such that for no w' it holds that $w' < w_0$* [4].

Notice that finite Rational models can be equivalently defined by postulating the existence of a function $k : \mathcal{W} \to \mathbb{N}$, and then letting $x < y$ iff $k(x) < k(y)$.

Definition 4. *The height $k_{\mathcal{M}}(F)$ of a formula F is $i = min\{k_{\mathcal{M}}(w) : \mathcal{M}, w \models F\}$. If there is no $w : \mathcal{M}, w \models F$, F has no height.*

Proposition 1. *For any $\mathcal{M} = \langle \mathcal{W}, V, < \rangle$ and any $w \in \mathcal{W}$, we have $\mathcal{M} \models A \mathrel{|\!\sim} B$ iff $k_{\mathcal{M}}(A \wedge B) < k_{\mathcal{M}}(A \wedge \neg B)$.*

As already mentioned, although the operator $\mathrel{|\!\sim}$ is *nonmonotonic*, the notion of logical entailment just defined is itself *monotonic*: if $K \models_P F$ and $K \subseteq K'$ then also $K' \models_P F$ (the same holds for $\models_R$). In order to define a nonmonotonic entailment we introduce our second ingredient of minimal models. The underlying idea is to restrict attention to models that minimize *the height of worlds*. Informally, given two models of K, one in which a given x has height 2 (because for instance $z < y < x$) , and another in which it has height 1 (because only $y < x$), we would prefer the latter, as in this model x is "more normal" than in the former.

[4] In the literature the function $k_{\mathcal{M}}$ is usually called *ranking*, but we call it *height* in order to avoid any confusion with the different notions of *ranking* as defined by Lehmann and Magidor and used in this paper as well. Our notion of ranking is similar to the one originally introduced by Spohn [11] and to the one introduced by Pearl [9]. Observe that the definition of height given above also works for preferential models.

In analogy with circumscription, there are mainly two ways of comparing models with the same domain: 1) by keeping the valuation function fixed (only comparing $\mathcal{M}$ and $\mathcal{M}'$ if V and V' in the two models respectively coincide); 2) by also comparing $\mathcal{M}$ and $\mathcal{M}'$ in case $V \neq V'$. We consider the two semantics resulting from these alternatives. The first one is a *fixed interpretations minimal semantics*, for short *FIMS*.

Definition 5 (*FIMS*)**.** *Given $\mathcal{M} = \langle \mathcal{W}, <, V\rangle$ and $\mathcal{M}' = \langle \mathcal{W}', <', V'\rangle$ we say that $\mathcal{M}$ is preferred to $\mathcal{M}'$ with respect to the fixed interpretations minimal semantics ($\mathcal{M} <_{FIMS} \mathcal{M}'$) if $\mathcal{W} = \mathcal{W}'$, $V = V'$, and for all x, $k_{\mathcal{M}}(x) \leq k_{\mathcal{M}'}(x)$ whereas there exists x' : $k_{\mathcal{M}}(x') < k_{\mathcal{M}'}(x')$. We say that $\mathcal{M}$ is minimal with respect to $<_{FIMS}$ in case there is no $\mathcal{M}'$ such that $\mathcal{M}' <_{FIMS} \mathcal{M}$. We say that K minimally entails a formula $F \in \mathcal{L}$ with respect to FIMS, and we write $K \models_{FIMS} F$, if F is valid in all models of K which are minimal with respect to $<_{FIMS}$.*

The following theorem shows that, for KBs that are sets of conditionals, we can characterize minimal models with fixed interpretations in terms of conditionals that are falsified by a world. Intuitively minimal models are those where the worlds of height 0 satisfy all conditionals, and the height (> 0) of a world x is determined by the height $k_{\mathcal{M}}(C)$ of the antecedents C of conditionals *falsified* by x. Given a model $\mathcal{M} = \langle \mathcal{W}, <, V\rangle$ and $x \in \mathcal{W}$, we define $S_x = \{C \mid\!\sim D \in K \mid \mathcal{M}, x \models C \wedge \neg D\}$.

Proposition 2. *Let K be a knowledge base and $\mathcal{M}$ a model, then $\mathcal{M} \models K$ if and only if $\mathcal{M}$ satisfies the following, for every $x \in \mathcal{W}$: 1. if $k_{\mathcal{M}}(x) = 0$ then $S_x = \emptyset$; 2. if $S_x \neq \emptyset$, then $k_{\mathcal{M}}(x) > k_{\mathcal{M}}(C)$ for every $C \mid\!\sim D \in S_x$.*

Observe that condition 1 is a consequence of condition 2; we have explicitly mentioned it for clarity (see the subsequent proposition and theorem).

Proposition 3. *Let K be a knowledge base and let $\mathcal{M}$ be a* minimal *model of K with respect to FIMS; then $\mathcal{M}$ satisfies for every $x \in \mathcal{W}$: 1. if $S_x = \emptyset$ then $k_{\mathcal{M}}(x) = 0$; 2. if $S_x \neq \emptyset$, then $k_{\mathcal{M}}(x) = 1 + max\{k_{\mathcal{M}}(C) \mid C \mid\!\sim D \in S_x\}$.*

Theorem 1. *Let K be a knowledge base and let $\mathcal{M}$ be any model, then $\mathcal{M}$ is a FIMS minimal model of K if and only if $\mathcal{M}$ satisfies for every $x \in \mathcal{W}$: 1. $S_x = \emptyset$ iff $k_{\mathcal{M}}(x) = 0$; 2. if $S_x \neq \emptyset$, then $k_{\mathcal{M}}(x) = 1 + max\{k_{\mathcal{M}}(C) \mid C \mid\!\sim D \in S_x\}$.*

In our second semantics, we let the interpretations vary. The semantics is called variable interpretations minimal semantics, for short *VIMS*.

Definition 6 (*VIMS*)**.** *Given $\mathcal{M} = \langle \mathcal{W}, <, V\rangle$ and $\mathcal{M}' = \langle \mathcal{W}', <', V'\rangle$ we say that $\mathcal{M}$ is preferred to $\mathcal{M}'$ with respect to the variable interpretations minimal semantics, and write $\mathcal{M} <_{VIMS} \mathcal{M}'$, if $\mathcal{W} = \mathcal{W}'$, and for all x, $k_{\mathcal{M}}(x) \leq k_{\mathcal{M}'}(x)$ whereas there exists x' : $k_{\mathcal{M}}(x') < k_{\mathcal{M}'}(x')$. We say that $\mathcal{M}$ is minimal with respect to $<_{VIMS}$ in case there is no $\mathcal{M}'$ such that $\mathcal{M}' <_{VIMS} \mathcal{M}$. We say that K minimally entails (with respect to VIMS) F, and write $K \models_{VIMS} F$, if F is valid in all models of K which are minimal with respect to $<_{VIMS}$.*

It is easy to realize that the two semantics, *FIMS* and *VIMS*, define different sets of minimal models. This is illustrated by the following example.

Example 1. Let $K = \{penguin \mid\sim bird, penguin \mid\sim \neg fly, bird \mid\sim fly\}$. We derive that $K \not\models_{FIMS} penguin \wedge black \mid\sim \neg fly$. Indeed in *FIMS* there can be a model $\mathcal{M}$ in which $\mathcal{W} = \{x, y, z\}$, $V(x) = \{penguin, bird, fly, black\}$, $V(y) = \{penguin, bird\}$, $V(z) = \{bird, fly\}$, and $z < y < x$. $\mathcal{M}$ is a model of K, and it is minimal with respect to *FIMS* (indeed once fixed $V(x), V(y), V(z)$ as above, it is not possible to lower the height of x nor of y nor of z unless we falsify K). Furthermore, in $\mathcal{M}$ x is a typical black penguin (since there is no other black penguin preferred to it) that flies. Therefore, $K \not\models_{FIMS} penguin \wedge black \mid\sim \neg fly$. On the other hand, $\mathcal{M}$ is not minimal with respect to *VIMS*. Indeed, consider the model $\mathcal{M}' = \langle \mathcal{W}, <', V' \rangle$ obtained from $\mathcal{M}$ by letting $V'(x) = \{penguin, bird, black\}$, $V'(y) = V(y)$, $V'(z) = V(z)$ and by defining $<'$ as: $z <' y$ and $z <' x$. Clearly $\mathcal{M}' \models K$, and $\mathcal{M}' <_{VIMS} \mathcal{M}$, since $k_{\mathcal{M}'}(x) < k_{\mathcal{M}}(x)$, while $k_{\mathcal{M}'} = k_{\mathcal{M}}$ for all other worlds.

The example above shows that *FIMS* and *VIMS* lead to different sets of minimal models for a given K. Notice however that the model $\mathcal{M}'$ we have used to illustrate this fact is not a minimal model for K in *VIMS*. A minimal model in *VIMS* for K that can be defined on the domain $\mathcal{W}$ is given by $V(x) = V(y) = V(z) = \{bird, fly\}$, and the empty relation $<$. This is quite a degenerate model of K in which there are no penguins. This illustrates the strength of *VIMS*: in case of knowledge bases that only contain positive conditionals, logical entailment in *VIMS* collapses into classical logic entailment. This feature corresponds to a similar feature of the non-monotonic logic $\mathbf{P}_{min}$ in [5], and can be proven in the same way.

Proposition 4. *Let K be a set of positive conditionals. Let us replace all formulas of the form $A \mid\sim B$ in K with $A \rightarrow B$, and call K' the resulting set of formulas. We have that $K \models_{VIMS} A \mid\sim B$ if and only if $K' \models_{PC} A \rightarrow B$.*

As for $\mathbf{P}_{min}$ this strong feature of *VIMS* can be avoided when considering knowledge bases that include existence assertions: these are negated conditionals, in the example for instance we could add $\neg(penguin \mid\sim \bot)$ to force us to consider non trivial models in which penguins exist. In the next section, we will use *VIMS* in this kind of way, by always considering knowledge bases that include existence assertions (expressed by negated conditionals).

3 A Semantical Reconstruction of Rational Closure

We provide a semantic characterization of the well known rational closure, described in [8] within the two semantics described in the previous section. More precisely, we can give two semantic characterizations of rational closure, the first based on *FIMS*, the second based on *VIMS*. Since in rational closure no boolean combinations of conditionals are allowed, in the following, the knowledge base K is just a finite set of positive conditional assertions. We recall the notion rational closure, giving its syntactical definition in terms of *rank* of a formula.

Definition 7. *Let K be a finite set of positive conditional assertions and A a propositional formula. A is said to be* exceptional *for K iff $K \models_R \top \mid\sim \neg A$*[5].

[5] In [8], $\models_P$ is used instead of $\models_R$. However when K contains only positive conditionals the two notions coincide (see footnote 1), then we use $\models_R$ here since we consider rational models.

A conditional formula $A \mid\!\sim B$ is exceptional for K if its antecedent A is exceptional for K. The set of conditional formulas which are exceptional for K will be denoted as $E(K)$. It is possible to define a non-decreasing sequence of subsets of K $C_0 \supseteq C_1, \ldots$ by letting $C_0 = K$ and, for $i > 0$, $C_i = E(C_{i-1})$. Observe that, being K finite, there is a $n \geq 0$ such that for all $m > n, C_m = C_n$ or $C_m = \emptyset$.

Definition 8. *A propositional formula A has* rank *i for K iff i is the least natural number for which A is not exceptional for C_i. If A is exceptional for all C_i then A has no rank.*

The notion of rank of a formula allows to define the rational closure of K.

Definition 9. *Let K be a conditional knowledge base. The rational closure $\bar{K}$ of K is the set of all $A \mid\!\sim B$ s.t. either (1) the rank of A is strictly less than the rank of $A \land \neg B$ (this includes the case A has a rank and $A \land \neg B$ has none), or (2) A has no rank.*

The rational closure of a knowledge base K seemingly contains all conditional assertions that, in the analysis of nonmonotonic reasoning provided in [8], one rationally wants to derive from K. For a full discussion, see [8].

Can we capture rational closure within our semantics? A first conjecture might be that the *FIMS* of Definition 5 could capture rational closure. However, we are soon forced to recognize that this is not the case. For instance, Example 1 above illustrates that $\{penguin \mid\!\sim bird, penguin \mid\!\sim \neg fly, bird \mid\!\sim fly\} \not\models_{FIMS} penguin \land black \mid\!\sim \neg fly$. On the contrary, it can be verified that $penguin \land black \mid\!\sim \neg fly$ is in the rational closure of $\{penguin \mid\!\sim bird, penguin \mid\!\sim \neg fly, bird \mid\!\sim fly\}$. Therefore, *FIMS* as it is does not allow us to define a semantics corresponding to rational closure. Things change if we consider *FIMS* applied to models that contain *all possible valuations "compatible" with a given knowledge base K*. We call these models *canonical models*.

Example 2. Consider Example 1 above. If we restrict our attention to models that also contain a w with $V(w) = \{penguin, bird, black\}$ which is a black penguin that does not fly and can therefore be assumed to be a typical penguin, we are able to conclude that typically black penguins do not fly, as in rational closure. Indeed, in all minimal models of K that also contain w with $V(w) = \{penguin, bird, black\}$, it holds that $penguin \land black \mid\!\sim \neg fly$.

We are led to the conjecture that *FIMS* restricted to canonical models could be the right semantics for rational closure. Fix a propositional language $\mathcal{L}_{Prop}$ comprising a finite set of propositional variables ATM, a propositional interpretation $v : ATM \longrightarrow \{true, false\}$ is *compatible* with K, if there is no formula $A \in \mathcal{L}_{Prop}$ such that $v(A) = true$ and $K \models_R A \mid\!\sim \bot$.

Definition 10. *A model $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$ satisfying a knowledge base K is said to be* canonical *if it contains (at least) a world associated to each propositional interpretation compatible with K, that is to say: if v is compatible with K, then there exists a world w in $\mathcal{W}$, such that for all propositional formulas B $\mathcal{M}, w \models B$ iff $v(B) = true$.*

Theorem 2. *For a given domain $\mathcal{W}$, there exists a unique canonical model $\mathcal{M}$ for K over $\mathcal{W}$ such that, for all other canonical models $\mathcal{M}'$ over $\mathcal{W}$, we have $\mathcal{M} <_{FIMS} \mathcal{M}'$.*

In the following, we show that the canonical models that are minimal with respect to *FIMS* are an adequate semantic counterpart of rational closure.

To prove the correspondence between the rational closure of a knowledge base K and the fixed interpretation minimal model semantics of K, we need to prove some propositions. The next one is a restatement for rational models of Lemma 5.18 in [8], and it can be proved in a similar way. Note that, as a difference, point 2 in Lemma 5.18 is an "if and only if" rather than an "if" statement.

Proposition 5. *Let $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$ be a rational model of K. Let $\mathcal{M}_0 = \mathcal{M}$ and, for all i, let $\mathcal{M}_i = \langle \mathcal{W}_i, <_i, V_i \rangle$ be the rational model obtained from $\mathcal{M}$ by removing all the worlds w with $k_{\mathcal{M}}(w) < i$, i.e., $\mathcal{W}_i = \{w \in \mathcal{W} : \ k_{\mathcal{M}}(w) \geq i\}$. For any propositional formula A, if $rank(A) \geq i$, then: (1) $k_{\mathcal{M}}(A) \geq i$; (2) If $C_i \models_R A \mathrel{|\!\sim} B$ then $\mathcal{M}_i \models A \mathrel{|\!\sim} B$.*

Proposition 6. *Let $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$ be a canonical model of K, minimal with respect to $<_{FIMS}$. For all $w \in \mathcal{W}$ it holds that: if $\mathcal{M}, w \models A \rightarrow B$ for all $A \mathrel{|\!\sim} B$ in C_i, then $k_{\mathcal{M}}(w) \leq i$.*

Proposition 7. *Let $\mathcal{M}$ be a canonical model of K minimal with respect to $<_{FIMS}$. Then, $rank(A) = i$ iff $k_{\mathcal{M}}(A) = i$.*

Theorem 3. *Let K be a knowledge base and $\mathcal{M}$ be a canonical model of K minimal with respect to $<_{FIMS}$. For all conditionals $A \mathrel{|\!\sim} B$ we have $\mathcal{M} \models A \mathrel{|\!\sim} B$ if and only if $A \mathrel{|\!\sim} B \in \overline{K}$, where $\overline{K}$ is the rational closure of K.*

In Theorem 3 we have shown a correspondence between rational closure and minimal models with fixed interpretations, *on the proviso that* we restrict our attention to minimal *canonical* models. We can obtain the same effect by extending K into K' by adding negated conditionals: $K' = K \cup \{\neg(C \mathrel{|\!\sim} \bot) \mid K \not\models_R (C \mathrel{|\!\sim} \bot)\}$. Indeed it can be easily verified that all models of K' are canonical, hence restricting *FIMS* to canonical models on the one hand and considering the extension of K as K' on the other hand amounts to the same effect. We can therefore restate Theorem 3 above as follows:

Theorem 4. *Let K be a knowledge base and let $K' = K \cup \{\neg(C \mathrel{|\!\sim} \bot) \mid K \not\models_R (C \mathrel{|\!\sim} \bot)\}$. For all conditionals $A \mathrel{|\!\sim} B$ we have $K' \models_{FIMS} A \mathrel{|\!\sim} B$ if and only if $A \mathrel{|\!\sim} B \in \overline{K}$, where $\overline{K}$ is the rational closure of K.*

Notice that the size of K' is exponential in that of K. Can we lift the restriction to canonical models by adopting a semantics based on variable valuations? In the general case, the answer is negative. We have already mentioned that if we consider knowledge bases consisting only positive conditionals logical entailment in *VIMS* collapses into classical logic entailment. To avoid this collapse, we can require that, when we are checking for entailment of a conditional $A \mathrel{|\!\sim} B$ from a K, at least an $A \wedge B$ world and an $A \wedge \neg B$ world are present in K. This can be obtained by adding to K the conditionals $\neg(A \wedge B \mathrel{|\!\sim} \bot)$ and $\neg(A \wedge \neg B \mathrel{|\!\sim} \bot)$. Also in this case, however, we cannot give a positive answer to the above question. In fact, it is possible to build a model of K, minimal with respect to *VIMS*, which falsifies a conditional $A \mathrel{|\!\sim} B$ which on the contrary is satisfied in all the canonical minimal models of K under *FIMS*. This is shown by the following example.

Example 3. Let K be the following: $\{T \mid\sim S$, $S \mid\sim \neg D$, $L \mid\sim P$, $R \mid\sim Q$, $E \mid\sim F$, $H \mid\sim G$, $D \mid\sim \neg P \wedge \neg Q \wedge \neg F \wedge \neg G$, $S \mid\sim \neg(L \wedge R)$, $S \mid\sim \neg(L \wedge E)$, $S \mid\sim \neg(L \wedge H)$, $S \mid\sim \neg(R \wedge E)$, $S \mid\sim \neg(R \wedge H)$, $S \mid\sim \neg(E \wedge H)\}$. Let $A = D \wedge S \wedge R \wedge L \wedge E \wedge H$,, $B = \neg Q \wedge \neg P \wedge \neg F \wedge \neg G$ and let $K' = K \cup \{\neg(A \wedge B \mid\sim \bot)$, $\neg(A \wedge \neg B \mid\sim \bot)\}$. We define a model $\mathcal{M} = (\mathcal{W}, <, V)$ of K', which is minimal with respect to *VIMS*, as follows: $\mathcal{W} = \{x, w, y_1.y_2, y_3\}$, where: $V(y_1) = \{S, \neg D, \neg R, \neg L, \neg E, \neg H, P, Q, F, G\}$, $V(y_2) = \{\neg S, \neg D, R, L, E, H, P, Q, F, G\}$, $V(y_3) = \{\neg S, \neg D, R, L, E, H, P, Q, F, G\}$, $V(x) = \{D, S, R, L, E, H, \neg Q, \neg P, \neg F, \neg G\}$, $V(w) = \{D, S, R, L, E, H, Q, \neg P, \neg F, \neg G\}$, with $k_{\mathcal{M}}(y_1) = 0$, $k_{\mathcal{M}}(y_2) = 1$, $k_{\mathcal{M}}(y_3) = 1$, $k_{\mathcal{M}}(x) = 2$ and $k_{\mathcal{M}}(w) = 2$. Observe that: x is an $A \wedge B$ minimal world; w is an $A \wedge \neg B$ minimal world; y_1 is an S minimal world; y_2 is a minimal world for R, L, E and H; and y_3 is a D minimal world.

$\mathcal{M}$ is a model of K which is minimal with respect to *VIMS*. Also, $A \mid\sim B$ is falsified in $\mathcal{M}$, while, on the contrary, $A \mid\sim B$ holds in all the canonical models minimal with respect to *FIMS*. Indeed, in all such models the height of $k(A \wedge B) = 2$ while $k(A \wedge \neg B) = 3$. However, it is not possible to construct a model $\mathcal{M}'$ with 5 worlds so that $\mathcal{M}' <_{VIMS} \mathcal{M}$. In particular, assigning to x or w height 1 would require the introduction of minimal worlds for R, L, E and H with height 0. But world y_2 cannot be given height 0, as it does not satisfy the conditionals with antecedent S. In canonical models there are distinct minimal R worlds, L worlds, E worlds and H worlds height 0 (which are also minimal S worlds).

As suggested by this example, in order to characterize rational closure in terms of *VIMS*, we should restrict our consideration to models which contain "enough" worlds. In the following, as in Theorem 4, we enrich K with negated conditionals but, as a difference with K' of Theorem 4, we only need to add to K a polynomial number of negated conditionals (instead of an exponential number). The purpose of the addition is that of restricting our attention to models minimal with respect to $<_{VIMS}$ that have a domain large enough to have, in principle, a distinct most-preferred world for each antecedent of conditional in K. For this reason, we add for each antecedent C of K a new corresponding atom ϕ_C. If the problem to be addressed is that of knowing whether $A \mid\sim B$ is logically entailed by K, we also introduce $\phi_{A \wedge B}$ and $\phi_{A \wedge \neg B}$, and we define K' as follows.

Definition 11. *We define* $A_{K, A \mid\sim B} = \{C \mid$ *either for some* D, $C \mid\sim D \in K$ *or* $C = A \wedge B$ *or* $C = A \wedge \neg B$, *and* $K \not\vdash_R C \mid\sim \bot\}$ *and* $K' = K \cup \{\neg(C \wedge \phi_C \mid\sim \bot) \mid C \in A_{K, A \mid\sim B}\} \cup \{(\phi_{C_i} \wedge \phi_{C_j} \mid\sim \bot) \mid C_i, C_j \in A_{K, A \mid\sim B}\}$.

We here establish a correspondence between *FIMS* and *VIMS*. By virtue of Theorem 3, this allows us to establish a correspondence between rational closure and *VIMS*, as stated by Theorem 6.

Theorem 5. *Let* $\mathcal{M}$ *be a canonical model of* K*, minimal with respect to FIMS, and let* K' *be the extension of* K *defined as in Definition 11. We have that* $\mathcal{M} \models A \mid\sim B$ *iff* $K' \models_{VIMS} A \mid\sim B$.

From Theorem 3 and Theorem 5 just shown, it follows that:

Theorem 6. $A \mid\sim B \in \bar{K}$ *iff* $K' \models_{VIMS} A \mid\sim B$ *for* K' *of Definition 11.*

4 Relation with $\mathbf{P}_{min}$ and Pearl's System Z

In [5] an alternative nonmonotonic extension of preferential logic **P** called $\mathbf{P}_{min}$ is proposed. Similarly to the semantics presented in this work, $\mathbf{P}_{min}$ is based on a minimal modal semantics. However the preference relation among models is defined in a different way. Intuitively, in $\mathbf{P}_{min}$ the fact that a world x is a minimal A-world is expressed by the fact that x satisfies $A \wedge \Box\neg A$, where $\Box$ is defined with respect to the inverse of the preference relation (i.e. with respect to the accessibility relation given by Ruv iff $v < u$). The idea is that preferred models are those that minimize the set of worlds where $\neg\Box\neg A$ holds, that is A-worlds which are not minimal. As a difference from the approach presented in this work, the semantics of $\mathbf{P}_{min}$ is defined starting from preferential models of Definition 1, in which the relation $<$ is irreflexive and transitive (thus, no longer modular). $\mathbf{P}_{min}$ is a nonmonotonic logic considering only **P** models that, intuitively, minimize the non-typical worlds. More precisely, given a set of formulas K, a model $\mathcal{M} = \langle \mathcal{W}_{\mathcal{M}}, <_{\mathcal{M}}, V_{\mathcal{M}} \rangle$ of K and a model $\mathcal{N} = \langle \mathcal{W}_{\mathcal{N}}, <_{\mathcal{N}}, V_{\mathcal{N}} \rangle$ of K, we say that $\mathcal{M}$ is preferred to $\mathcal{N}$ if $\mathcal{W}_{\mathcal{M}} = \mathcal{W}_{\mathcal{N}}$, and the set of pairs $(w, \neg\Box\neg A)$ such that $\mathcal{M}, w \models \neg\Box\neg A$ is strictly included in the corresponding set for $\mathcal{N}$. A model $\mathcal{M}$ is a *minimal model* for K if it is a model of K and there is no a model $\mathcal{M}'$ of K which is preferred to $\mathcal{M}$. Entailment in $\mathbf{P}_{min}$ is restricted to minimal models of a given set of formulas K. In Section 3 of [5] it is observed that the logic $\mathbf{P}_{min}$ turns out to be quite strong. In general, if we only consider knowledge bases containing only positive conditionals, we get the same trivialization result (part of Proposition 1 in [5]) as the one contained in Proposition 4 for VIMS.

This does not hold for rational closure. This is the reason why we have introduced the additional assumptions of Definition 11 in order to obtain an equivalence with rational closure. Similarly, in order to tackle this trivialization in $\mathbf{P}_{min}$, Section 3 in [5] is focused on the so called *well-behaved knowledge bases*, that explicitly include that A is possible ($\neg(A \mid\sim \bot)$) for all conditional assertions $A \mid\sim B$ in the knowledge base.

We can now wonder whether $\mathbf{P}_{min}$ is equivalent to VIMS, which is the semantics to which it resembles the most. Or whether VIMS is equivalent to a stronger version of $\mathbf{P}_{min}$ obtained by replacing **P** with **R** as the underlying logic. We call $\mathbf{R}_{min}$ this stronger version of $\mathbf{P}_{min}$.

Example 4. Let $K = \{\mathit{PhD} \mid\sim \neg \mathit{worker}, \mathit{PhD} \mid\sim \mathit{adult}, \mathit{adult} \mid\sim \mathit{worker}, \mathit{italian} \mid\sim \mathit{house\_owner}, \mathit{PhD} \mid\sim \neg \mathit{house\_owner}\}$. What do we derive in $\mathbf{P}_{min}$ and $\mathbf{R}_{min}$, and what in VIMS? By what said above, since K only contains positive conditionals, both in $\mathbf{P}_{min}$ and $\mathbf{R}_{min}$, on the one side, and in VIMS, on the other side, we derive that $\mathit{italian} \wedge \mathit{PhD} \mid\sim \bot$. So let's add to K the constraint that people who are italian and have a PhD do exist by introducing in K a conditional $\neg(\mathit{italian} \wedge \mathit{PhD} \mid\sim \bot)$, thus obtaining: $K' = \{\mathit{PhD} \mid\sim \neg \mathit{worker}, \mathit{PhD} \mid\sim \mathit{adult}, \mathit{adult} \mid\sim \mathit{worker}, \mathit{italian} \mid\sim \mathit{house\_owner}, \mathit{PhD} \mid\sim \neg \mathit{house\_owner}, \neg(\mathit{italian} \wedge \mathit{PhD} \mid\sim \bot)\}$.
Notice that since $\neg(\mathit{italian} \wedge \mathit{PhD} \mid\sim \bot)$ entails both that $\neg(\mathit{italian} \mid\sim \bot)$ and that $\neg(\mathit{PhD} \mid\sim \bot)$, and that this in turn entails $\neg(\mathit{adult} \mid\sim \bot)$, K' is also well-behaved.
It can be verified that the logical consequences of K' in $\mathbf{P}_{min}$, $\mathbf{R}_{min}$, and VIMS differ. In both $\mathbf{P}_{min}$ and $\mathbf{R}_{min}$, for instance, we derive neither that $\mathit{italian} \wedge \mathit{PhD} \mid\sim \mathit{house\_owner}$ nor that $\mathit{italian} \wedge \mathit{PhD} \mid\sim \neg \mathit{house\_owner}$: the two alternatives are equivalent. On the other hand, in VIMS we derive that $\mathit{italian} \wedge \mathit{PhD} \mid\sim \neg \mathit{house\_owner}$.

The previous example shows that in some cases $VIMS$ is stronger than both $\mathbf{P}_{min}$ and $\mathbf{R}_{min}$. The following one shows that the two approaches are incomparable, since there are also consequences that hold for both $\mathbf{P}_{min}$ and $\mathbf{R}_{min}$ but not for $VIMS$.

Example 5. Let $K = \{PhD \mid\!\sim adult, adult \mid\!\sim work, PhD \mid\!\sim \neg work, italian \mid\!\sim house\_owner\}$. What do we derive about typical $italian \wedge PhD \wedge work$, for instance? Do they inherit the property of typical italians of being $house\_owner$? Again, in order to prevent the entailment of $italian \wedge PhD \wedge work \mid\!\sim \bot$ from K both in $VIMS$ and in $\mathbf{P}_{min}$ and $\mathbf{R}_{min}$, we add to K the constraint that italians with a PhD who work exist, henceforth they also have typical instances. Therefore we expand K into $K' = \{PhD \mid\!\sim adult, adult \mid\!\sim work, PhD \mid\!\sim \neg work, italian \mid\!\sim house\_owner, \neg(italian \wedge PhD \wedge work \mid\!\sim \bot)\}$. By reasoning as in Example 4 we can show that K' is a well-behaved knowledge base. Now it can be shown that $italian \wedge PhD \wedge work \mid\!\sim house\_owner$ is entailed in $\mathbf{P}_{min}$ and $\mathbf{R}_{min}$, whereas nothing is entailed in $VIMS$. This difference can be explained intuitively as follows. The set of properties for which an individual is atypical matters in $\mathbf{P}_{min}$ and $\mathbf{R}_{min}$ where one has to minimize the set of distinct $\neg\Box\neg C$: even if an $italian \wedge PhD \wedge work$ is an atypical PhD, $\mathbf{P}_{min}$ and $\mathbf{R}_{min}$ still maximize its typicality as an italian, and therefore entail that it is a house_owner, as all typical italians. As a difference, in $VIMS$, what matters is *the set of individuals which are more typical* than a given x, rather than *the set of properties* with respect to which they are more typical. As a consequence, since an x which is $italian \wedge PhD \wedge work$ is an atypical PhD, there is no need to maximize its typicality as an italian, as long as this does not increase the set of individuals more typical than x.

In [9] Pearl has introduced two notions of 0-entailment and 1-entailment to perform nonmonotonic reasoning. We recall here the semantic definition of both and then we remark upon their relation with our semantics and rational closure. A model $\mathcal{M}$ for a finite knowledge base K has the form $\mathcal{M} = (\{true, false\}^{ATM}, k_{\mathcal{M}})$ where $\{true, false\}^{ATM}$ is the set of propositional interpretations for, say, a fixed finite propositional language, and $k_{\mathcal{M}}$ is our height function mapping propositional interpretations to $\mathbb{N}$, the definition of height $k_{\mathcal{M}}(A)$ of a formula is the same as in our semantic. A conditional $A \mid\!\sim B$ is true in a model $\mathcal{M}$ if $k_{\mathcal{M}}(A \wedge B) < k_{\mathcal{M}}(A \wedge \neg B)$. Then the two entailments relations are defined as follows:

$K \models_{0-ent} A \mid\!\sim B$ if $A \mid\!\sim B$ is true in all models of K
$K \models_{1-ent} A \mid\!\sim B$ if $A \mid\!\sim B$ is true in the (unique) model $\mathcal{M}$ of K which is *minimal* with respect to $k_{\mathcal{M}}$.

(minimal with respect to $k_{\mathcal{M}}$ means that no other model $\mathcal{M}'$ assigns a lower value $k_{\mathcal{M}'}$ to any propositional interpretation). First, observe that Pearl's semantics (both 0 and 1 entailment) cannot cope with conditionals having an inconsistent antecedent. This limitation is deliberate and is motivated by a probabilistic interpretation of conditionals: in asserting $A \mid\!\sim B$, A must not be impossible, no matter how it is unlikely. For this reason, a knowledge base such as $K = \{A \mid\!\sim P, A \mid\!\sim \neg P, B \mid\!\sim Q\}$ is out of the scope of Pearl's semantics, and nothing can be said about its consequences. As a difference with respect to Pearl's approach we are able to consider such K, we just derive that A is impossible, without concluding that K is inconsistent or trivial, in the sense that everything follows from it. Moreover both 0-entailment and 1-entailment fail to validate:

$$\emptyset \models_{0-ent/1-ent} A \mid\sim \bot \text{ whenever } \vdash_{PC} \neg A$$

which is valid in any KLM logic, whence in rational closure (as well as in our semantics). However two definitions should make apparent the relations with our semantics and rational closure. If we consider a K such that $\forall A \mid\sim B \in K, K \not\models_R A \mid\sim \bot$, we get an obvious correspondence between our *canonical* models (which will contain worlds for very possible propositional interpretation) and models of Pearl's semantics. The correspondence preserves $FIMS$ minimality, so that we get immediately:

Proposition 8. *$K \models_{1-ent} A \mid\sim B$ iff $A \mid\sim B$ holds in any FIMS-minimal* canonical *model of K.*

By Theorem 3, we therefore obtain $K \models_{1-ent} A \mid\sim B$ iff $A \mid\sim B \in \bar{K}$. This is not a surprise, the correspondence between 1-entailment and rational closure was already observed by Pearl in [9,10]. However, it only works for knowledge bases with the strong consistency assumption as above.

5 Conclusions and Future Works

We have provided a semantic reconstruction of the known rational closure within modal logic. We have provided two minimal model semantics, based on the idea that preferred rational models are those one in which the height of the worlds is minimized. We have then shown that adding suitable possibility assumptions to a knowledge base, these two minimal model semantics correspond to rational closure.

The correspondence between the proposed minimal model semantics and rational closure suggests the possibility of defining variants of rational closure by varying the three ingredients underlying our approach, namely: (i) the properties of the preference relation $<$: for instance just preorder, or multi-linear [5], or weakly-connected (observe that $\mathbf{P}$ is complete with respect to any of the three classes); (ii) the comparison relation on models: for instance based on the heights of the worlds or on the inclusion between the relations $<$, or on negated boxed formulas satisfied by a world, as in the logic $\mathbf{P}_{min}$; (iii) the choice between fixed or variable interpretations. The systems obtained by various combinations of the three ingredients are largely unexplored and may give rise to useful nonmonotonic logics. We finally intend to extend our approach to richer languages, notably in the context of nonmonotonic description logics.

References

1. Booth, R., Paris, J.B.: A note on the rational closure of knowledge bases with both positive and negative knowledge. Journal of Logic, Language and Information 7, 165–190 (1998)
2. Boutilier, C.: Conditional logics of normality: a modal approach. Artificial Intelligence 68(1), 87–154 (1994)
3. Casini, G., Straccia, U.: Rational Closure for Defeasible Description Logics. In: Janhunen, T., Niemelä, I. (eds.) JELIA 2010. LNCS, vol. 6341, pp. 77–90. Springer, Heidelberg (2010)
4. Friedman, N., Halpern, J.Y.: Plausibility measures and default reasoning. Journal of the ACM 48(4), 648–685 (2001)
5. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A Nonmonotonic Extension of KLM Preferential Logic **P**. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR-17. LNCS, vol. 6397, pp. 317–332. Springer, Heidelberg (2010)

6. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A minimal model semantics for rational closure. In: Rosati, R., Woltran, S. (eds.) 14th International Workshop on Non-Monotonic Reasoning, NMR 2012, Roma, Italy (2012)
7. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
8. Lehmann, D., Magidor, M.: What does a conditional knowledge base entail? Artificial Intelligence 55(1), 1–60 (1992)
9. Pearl, J.: System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning. In: Parikh, R. (ed.) TARK (3rd Conference on Theoretical Aspects of Reasoning about Knowledge), pp. 121–135. Morgan Kaufmann, Pacific Grove (1990)
10. Pearl, J., Goldszmidt, M.: On the relation between rational closure and System-Z. In: NMR 1990 (3rd Int. W. on Non-Monotonic Reasoning), South Lake Tahoe, California, USA (1990)
11. Spohn, W.: Ordinal conditional functions: A dynamic theory of epistemic states. In: Harper, W.L., Skyrms, B. (eds.) Causation in Decision, Belief Change, and Statistics, pp. 105–134. Kluwer, Dordrecht (1988)

Extending a Temporal Defeasible Argumentation Framework with Possibilistic Weights

Lluís Godo[1], Enrico Marchioni[1], and Pere Pardo[1,2]

[1] Artificial Intelligence Research Institute, IIIA
Spanish National Research Council, CSIC
Campus UAB, 08193 Bellaterra, Spain
{godo,enrico,pardo}@iiia.csic.es

[2] Dept. Lògica, Història i Filosofia de la Ciència
Universitat de Barcelona, Montalegre 6, 08001 Barcelona, Spain

Abstract. Recently, a temporal extension of the argumentation defeasible reasoning system DeLP has been proposed. This system, called t-DeLP, allows to reason defeasibly about changes and persistence over time but does not offer the possibility of ranking defeasible rules according to criteria of preference or certainty (in the sense of belief). In this contribution we extend t-DeLP by allowing to attach uncertainty weights to defeasible temporal rules and hence stratifying the set of defeasible rules in a program. Technically speaking, weights are modelled as necessity degrees within the frame of possibility theory, a qualitative model of uncertainty.

1 Introduction and Motivation

The system DeLP [13] provides a defeasible logic programming-based argumentation framework, based on the use of dialectical trees, upon which several extensions have been built. In particular, the system t-DeLP [17] makes it possible to express defeasible reasoning in a discrete temporal framework as well as changes and persistence over time. t-DeLP, however, does not offer the possibility of ranking defeasible rules according to criteria of preference or certainty (in the sense of belief).

In addition, in t-DeLP, the application of rules does not take into account any possible uncertainty in the occurrence of temporal events. This issue has been already tackled in another extension of DeLP, the system P-DeLP [3,2,8], that allows the handling of possibilistic uncertainty (of a qualitative, ordinal nature) by attaching defeasible rules and arguments with necessity degrees.

To bridge this gap, in this paper we introduce an extended argumentation framework, called pt-DeLP, where it is possible to express uncertainty on temporal rules and events, and how this uncertainty may change over time. This new system presents the possibility of formalizing arguments and defeat relations among them that combine both temporal criteria of t-DeLP and the belief strength criteria from P-DeLP. In fact, preferred arguments in t-DeLP are those with comparably more (temporally basic) information, or comparably less use of persistence rules (which simply assume that some fact will not change). This allows to express certain non-monotonic temporal phenomena (extinction of facts vs. persistence) in an economic way. On the other hand, P-DeLP establishes a preference for arguments that support their conclusions with comparably higher weights. Then, a natural way of combining these two kinds of preference (defeat) relations is through the definition of a lexicographic preference relation (see e.g. [4]) that assigns more relevance to temporal criteria or vice versa. We will show that, under any of these two lexicographic combinations, pt-DeLP is a conservative extension of t-DeLP, while this is not the case with P-DeLP.

The paper is structured as follows. In Section 2 we present the logic underlying the proposed logic programming framework, which is introduced in Section 3. The relationship between t-DeLP and P-DeLP to the new framework pt-DeLP is studied in Section 4. Then, an illustrative example is developed in Section 5 and we finish with a brief discussion on related work and conclusions.

2 Language and Semantics of the Base Logic of pt-DeLP

The logic upon which pt-DeLP is based on consists of (Boolean combinations of) temporal formulas, encoding the occurrence of an event at a given time, equipped with a weight that represents the degree of uncertainty attached to the formula. Here, we describe how pt-DeLP is defined within such a syntactic framework.

Given a finite set of propositional variables $\mathsf{Var} = \{p, q, \dots\}$, let us denote by Lit the set of literals built from Var, i.e. $\mathsf{Lit} = \{p, \neg p \mid p \in \mathsf{Var}\}$. By $\neg\ell$ we will denote $\neg p$ if $\ell = p$, or p if $\ell = \neg p$, with $p \in \mathsf{Var}$.

The set ATForm is defined as the set of all pairs (ℓ, t) such that $\ell \in \mathsf{Lit}$ and $t \in \mathbb{N}$. Every formula in ATForm is called an *atomic temporal formula.* The set TForm of *temporal formulas* is built, as usual, from the set of atomic temporal formulas with the classical Boolean connectives $\wedge, \vee, \neg, \longrightarrow$.

A temporal interpretation for TForm is a mapping $w : \mathsf{ATForm} \times \mathbb{N} \to \{0, 1\}$ such that, for each $t \in \mathbb{N}$, $w((\neg p, t)) = 1 - w((p, t))$. The interpretation w extends to formulas TForm as usual using the classical truth-functions for the Boolean connectives. Notice that the above extra condition ensures that, in fact, temporal formulas $(\neg\ell, t)$ and $\neg(\ell, t)$ are considered as logically equivalent.

We will denote by Ω the set of temporal interpretations over TForm. An interpretation $w \in \Omega$ is called a *model* of a temporal formula Φ, denoted by $w \models \Phi$, whenever $w(\Phi) = 1$.

In what follows, we expand the temporal language by introducing weights and provide a suitable semantics in terms of possibilistic uncertainty.[1]

Definition 1. *A* possibilistic model *over* TForm *is a possibility distribution* $\pi : \Omega \to [0, 1]$ *such that* $\max_{w \in \Omega} \pi(w) = 1$. *The possibility distribution* π *induces a necessity measure* N_π *on* TForm *in the usual way, i.e.:* $N_\pi(\Phi) = \inf\{1 - \pi(w) \mid w \in \Omega, w \not\models \Phi\}$.

[1] For all the details on possibility theory and possibilistic logic the reader is referred e.g. to [11] and the references therein.

A *weighted temporal formula* (wt-formula) is an expression of the form $\langle \Phi \rangle_r$, with $\Phi \in \mathsf{TForm}$, and $r \in [0,1]$, which is to be interpreted as a lower bound for the necessity degree of Φ. Note that a formula like $\langle (\ell_1, t) \longrightarrow (\ell_2, t) \rangle_r$ is a formula from TForm, while $\langle (\ell_1 \longrightarrow \ell_2, t) \rangle_r$ is not. The set of all wt-formulas is denoted WTForm.

Definition 2. *A* possibilistic model π *over* TForm *satisfies a weighted temporal formula* $\langle \Phi \rangle_r$*, denoted* $\pi \models_{pos} \langle \Phi \rangle_r$*, whenever* $N_\pi(\Phi) \geq r$*. As usual, we say that* π *satisfies a set of wt-formulas* P *whenever* π *satisfies every* $\langle \Phi \rangle_r \in P$*. Moreover, the induced consequence relation is defined as follows:* $P \models_{pos} \langle \Phi \rangle_r$ *iff every* π *satisfying* P *also satisfies* $\langle \Phi \rangle_r$*.*

We now define the language of pt-DeLP as a fragment of WTForm, replacing $\longrightarrow$ by $\longleftarrow$ as costumary in logic programming languages. Let

$$\mathsf{WTLit} = \{\langle (\ell, t) \rangle_r \mid \ell \in \mathsf{Lit}, t \in \mathbb{N}, r \in [0,1]\}$$

be the set of all wt-formulas $\langle \Phi \rangle_r$ where Φ is in ATForm. Each formula of this kind is called a *weighted temporal literal* [2] (wt-literal, for short). Moreover, let

$$\begin{aligned}\mathsf{WTRule} = \{\langle (\ell, t) \longleftarrow (\ell_1, t_1) \wedge \cdots \wedge (\ell_n, t_n) \rangle_r \mid (\ell, t), \ldots, (\ell_n, t_n) \in \mathsf{ATForm};\\ t \geq \max\{t_1, \ldots, t_n\}, r \in [0,1]\}\end{aligned}$$

be the set of all wt-formulas $\langle \Phi \rangle_r$ where Φ is a temporal formula of the form $(\ell, t) \longleftarrow (\ell_1, t_1) \wedge \cdots \wedge (\ell_n, t_n)$. Each formula contained in WTRule is called a *weighted temporal rule* (wt-rule, for short).[3] If $\delta = \langle (\ell, t) \longleftarrow (\ell_1, t_1) \wedge \cdots \wedge (\ell_n, t_n) \rangle_r$, we write $\mathsf{head}(\delta) = (\ell, t)$, $\mathsf{body}(\delta) = \{(\ell_1, t_1), \ldots, (\ell_n, t_n)\}$, and $\mathsf{lit}(\delta) = \{\mathsf{head}(\delta)\} \cup \mathsf{body}(\delta)$. If $r = 1$, δ is called a *strict* rule, and *defeasible* otherwise (i.e. when $0 < r < 1$). The language of pt-DeLP corresponds to the fragment of WTForm consisting of the set of formulas $\mathsf{WTLit} \cup \mathsf{WTRule}$.

Note that rules from WTRule are forward temporal rules (i.e rules in which the time of the occurrence of the literal in the conclusion follows the time of the occurrences of the premises), and so we keep from t-DeLP the idea that temporal rules represent causal relationships. For instance, a wt-rule like $\langle (\mathsf{dead}(\text{Lars}), t) \longleftarrow (\mathsf{bitten}(\text{Lars}), t_1) \wedge (\mathsf{antidote}(\text{Lars}), t_2)) \rangle_\alpha$ means: it is α-plausible that Lars dies at t when poisoned at t_1 and given an antidote at t_2.

It turns out that in many situations the weight attached to a temporal formula $(\ell, t) \longleftarrow (\ell_1, t_1) \wedge \cdots \wedge (\ell_n, t_n)$ does not actually depend on the absolute time values but on the temporal distances $t - t_1, \ldots, t - t_n$ between the time of occurrence of the head of the rule and of the different premises. For this reason we introduce the following convenient and schematic notation for specifying a given temporal evolution of the weights in a wt-rule. In a sense, such schematic representations assume that the weights are invariant under uniform time translations: a *schematic* weighted temporal rule is an expression δ_ν of the form

$$\langle \ell \longleftarrow (\ell_1, \mathsf{d}_1) \wedge \cdots \wedge (\ell_n, \mathsf{d}_n) \rangle_{\nu(\mathsf{d}_1, \ldots, \mathsf{d}_n)},$$

[2] In the rest of the work, wt-literals $\langle (\ell, t) \rangle_r$ will be written in the simpler form $(\ell, t)_r$.
[3] The notation $\langle (\ell, t) \longleftarrow (\ell_1, t_1), \ldots, (\ell_n, t_n) \rangle_r$ will also be used later for wt-rules.

where $\mathtt{d}_1, \ldots, \mathtt{d}_n$ are variables taking values in $\mathbb{N}$ and $\nu : \mathbb{N}^n \to [0,1]$. This schematic rule compactly encodes the set of the following instantiated *wt*-rules:

$$\delta = \langle (\ell, t) \longleftarrow (\ell_1, t - d_1) \wedge \cdots \wedge (\ell_n, t - d_n) \rangle_{\nu(d_1, \ldots, d_n)}$$

for each $d_1, \ldots, d_n \in \mathbb{N}$ and for each $t \geq \max\{d_1, \ldots, d_n\}$. The function ν is called a *weight distribution* and assigns to each instantiated rule δ a weight $\nu(d_1, \ldots, d_n)$ that depends on the temporal distances $d_1, \ldots, d_n$ of the head of the rule ℓ with respect to each premise ℓ_i in the body.

As a particular case, we can represent weighted versions of t-DeLP persistence rules δ_ℓ for selected literals ℓ. These are of the form $\langle \ell \longleftarrow (\ell, \mathtt{d}) \rangle_{\nu(\mathtt{d})}$ which state that a literal ℓ holding at t will still hold at $t + d$ with degree $\nu(d)$, for any d.

The notion of derivability in pt-DeLP is the natural extension of that of t-DeLP with possibilistic weights by means of a weighted version of modus ponens (which is sound with respect to the above possibilistic semantics, see e.g. [11]).

Definition 3 (Derivability). *Given a set of wt-literals and wt-rules* $P \subseteq \mathsf{WTLit} \cup \mathsf{WTRule}$*, we say that a wt-literal* $(\ell, t)_r$ *is* derivable *from* P*, denoted* $(\ell, t)_r \in Cn(P)$*, iff*

1. $(\ell, t)_{r'} \in P$ *with* $r \leq r'$*, or*
2. *there exist a set of wt-literals* $(\ell_1, t_1)_{r_1}, \ldots, (\ell_n, t_n)_{r_n} \in Cn(P)$ *and a wt-rule*

$$\langle (\ell, t) \longleftarrow (\ell_1, t_1) \wedge \cdots \wedge (\ell_n, t_n) \rangle_s, \qquad \textit{such that } r \leq \min\{s, r_1, \ldots, r_n\}.$$

3 An Argumentation System for pt-DeLP

Logic-based argumentation systems aim at providing computational tools to reason under conflicting or inconsistent information. Therefore, it is crucial to have a clear notion of what an inconsistent set of pt-DeLP formulas is.

Definition 4. *A set* $P \subseteq \mathsf{WTLit} \cup \mathsf{WTRule}$ *is said to be pt-DeLP-inconsistent if there exist* $(\ell, t)_r, (\neg\ell, t)_s \in Cn(P)$ *with* $\min(r, s) > 0$*. Otherwise, we say that* P *is pt-DeLP-consistent.*

Note that if P is inconsistent, P is not satisfiable by any possibilistic model since $\min(N_\pi((\ell, t)), N_\pi((\neg\ell, t))) = 0$ in any possibilistic model π. The converse is not true, since the notion of derivability in pt-DeLP is obviously weaker than the possibilistic logical consequence $\models_{pos}$. In other words, if $(\ell, t)_r \in Cn(P)$ then $P \models_{pos} (\ell, t)_r$, but the opposite does not always hold.

Definition 5 (Program). *A* pt-DeLP program *is a pair* (Π, Δ) *such that* Π *is a consistent finite set of strict wt-rules and* Δ *is a finite set defeasible wt-rules.*

Definition 6 (Argument). *Given a pt-DeLP program* $P = (\Pi, \Delta)$*, an* argument *for* $(\ell, t)_r$ *is a set* $\mathcal{A} = \mathcal{A}_\Pi \cup \mathcal{A}_\Delta$*, with* $\mathcal{A}_\Pi \subseteq \Pi$ *and* $\mathcal{A}_\Delta \subseteq \Delta$*, such that:*

1. $\Pi \cup \mathcal{A}_\Delta$ *is pt-DeLP-consistent,*

2. $r = \max\{s \in [0,1] \mid (\ell, t)_s \in Cn(\mathcal{A}_\Delta \cup \mathcal{A}_\Pi)\}$,
3. *Both* $\mathcal{A}_\Delta$ *and* $\mathcal{A}_\Pi$ *are minimal w.r.t. inclusion, i.e.: there are no* $\mathcal{A}'_\Delta \subset \mathcal{A}_\Delta$ *and* $\mathcal{A}'_\Pi \subset \mathcal{A}_\Pi$ *such that* $(\ell, t)_r \in Cn(\mathcal{A}'_\Delta \cup \mathcal{A}'_\Pi)$.

Notice that, given a pt-DeLP program (Π, Δ) there may exist different arguments for $(\ell, t)_r$, with different r. In fact, there might be different sets $\mathcal{A} = \mathcal{A}_\Pi \cup \mathcal{A}_\Delta$ and $\mathcal{A}' = \mathcal{A}'_\Pi \cup \mathcal{A}'_\Delta$, with $\mathcal{A}_\Pi \not\subseteq \mathcal{A}'_\Pi$ and $\mathcal{A}_\Delta \not\subseteq \mathcal{A}'_\Delta$, such that $\mathcal{A}$ is an argument for $(\ell, t)_s$, while $\mathcal{A}'$ is an argument for $(\ell, t)_{s'}$, with $s \neq s'$.

Definition 7. *Given a pt-DeLP program* (Π, Δ) *and an argument* $\mathcal{A} = \mathcal{A}_\Pi \cup \mathcal{A}_\Delta$ *for* $(\ell, t)_r$*, we define:*

1. $\mathsf{concl}(\mathcal{A}) = (\ell, t)_r$;
2. $\mathsf{base}(\mathcal{A}) = \{(\ell', t) \mid \exists \delta \in \mathcal{A}, (\ell', t) \in \mathsf{body}(\delta),$ *and* $\nexists \delta \in \mathcal{A}, (\ell', t) \in \mathsf{head}(\delta)\}$;
3. $\mathsf{TLit}(\mathcal{A}) = \bigcup_{\delta \in \mathcal{A}} \mathsf{lit}(\delta)$.

Definition 8 (Sub-argument). *Let* (Π, Δ) *be a pt-DeLP program and let* $\mathcal{A} = \mathcal{A}_\Pi \cup \mathcal{A}_\Delta$ *be an argument for* $(\ell, t)_r$ *in* (Π, Δ)*. Given some* $(\ell_0, t_0) \in \mathsf{TLit}(\mathcal{A})$*, a* sub-argument *for* $(\ell_0, t_0)_s$ *(for some* $s \in (0, 1]$*) is a set* $\mathcal{B} = \mathcal{B}_\Pi \cup \mathcal{B}_\Delta$*, with* $\mathcal{B}_\Delta \subseteq \mathcal{A}_\Delta$ *and* $\mathcal{B}_\Pi \subseteq \mathcal{A}_\Pi$*, such that* $\mathcal{B}$ *is an argument for* $(\ell_0, t_0)_s$*. The sub-argument of* $\mathcal{A}$ *induced by* (ℓ_0, t_0) *will be denoted* $\mathcal{A}(\ell_0, t_0)$*.*

We now define the notion of attack between arguments as a natural extension of those for DeLP and t-DeLP.

Definition 9 (Attack). *Given a pt-DeLP program* (Π, Δ)*, let* $\mathcal{A}_0$ *and* $\mathcal{A}_1$ *be arguments for* $(\ell_0, t_0)_{r_0}$ *and* $(\ell_1, t_1)_{r_1}$*, respectively. We say that* $\mathcal{A}_1$ attacks $\mathcal{A}_0$ *iff there exists a subargument* $\mathcal{B}_0$ *of* $\mathcal{A}_0$ *for a weighted temporal literal* $(\neg \ell_1, t_1)_s$ *for some* $s > 0$*. In this case,* $\mathcal{A}_1$ *is said to attack* $\mathcal{A}_0$ *at* $\mathcal{B}_0$*.*

Given the notion of attack, we can now consider several notions of defeat. On the one hand, following the idea proposed in P-DeLP (Possibilsitic DeLP), the presence of weights makes it reasonable to use such weights in deciding whether an attacking argument defeats another argument. Note that the next definition is a direct extension to pt-DeLP of the notion of defeater in P-DeLP.

Definition 10 (Possibilistic Defeater). *Let an argument* $\mathcal{A}_1$ *attack* $\mathcal{A}_0$ *at* $\mathcal{B}_0$*, where we have* $\mathsf{concl}(\mathcal{A}_1) = (\ell, t)_r$ *and* $\mathsf{concl}(\mathcal{B}_0) = (\neg \ell, t)_s$*. We say that* $\mathcal{A}_1$ *is a* proper possibilistic defeater *for* $\mathcal{A}_0$*, denoted* $\mathcal{A}_1 \succ_\pi \mathcal{A}_0$*, whenever* $r > s$*; and we say that* $\mathcal{A}_1$ *is a* blocking possibilistic defeater *for* $\mathcal{A}_0$*, denoted* $\mathcal{A}_1 \preceq\succeq_\pi \mathcal{A}_0$*, whenever* $s = r$*.*

Still, time and information specificity plays a fundamental role in reasoning in pt-DeLP, and so the definition given in [17] of a *temporal* defeater for t-DeLP should be also taken into account. In the following definition $\Delta_{\neg \ell}$ denotes a suitable set of instances of the schematic persistence rule $\langle \neg \ell \longleftarrow (\neg \ell, \mathsf{d}) \rangle_{\nu(\mathsf{d})}$.

Definition 11 (Temporal Defeater). *Let an argument* $\mathcal{A}_1$ *attack* $\mathcal{A}_0$ *at* $\mathcal{B}_0$*, where* $\mathsf{concl}(\mathcal{A}_1) = (\ell, t)_r$ *and* $\mathsf{concl}(\mathcal{B}_0) = (\neg \ell, t)_s$*. We say that* $\mathcal{A}_1$ *is a* proper temporal defeater *for* $\mathcal{A}_0$*, denoted* $\mathcal{A}_1 \succ_\tau \mathcal{A}_0$ *iff*

1. $\mathsf{base}(\mathcal{A}_1) \not\supsetneq \mathsf{base}(\mathcal{B}_0)$, *or*
2. *there exists* $t' < t$ *such that* $\mathcal{B}_0 = \mathcal{A}_1(\ell, t') \cup \Delta_{\neg\ell}$

We say $\mathcal{A}_1$ *is a* blocking temporal defeater *for* $\mathcal{A}_0$*, denoted* $\mathcal{A}_1 \preceq\succeq_\tau \mathcal{A}_0$ *iff neither* $\mathcal{A}_1$ *nor* $\mathcal{B}_0$ *are proper temporal defeaters for each other.*

The fact that pt-DeLP is built upon the presence of time and weights, formalized in terms of a necessity measures, suggests that both the concepts of possibilistic and a temporal defeaters should be taken into account to define a proper notion of defeat.

Definition 12 (($\pi \times \tau$)-defeater, ($\tau \times \pi$)-defeater). *Let an argument* $\mathcal{A}_1$ *attack* $\mathcal{A}_0$ *at* $\mathcal{B}_0$*, where* $\mathsf{concl}(\mathcal{A}_1) = (\ell, t)_r$ *and* $\mathsf{concl}(\mathcal{B}_0) = (\neg\ell, t)_s$*. We say that* $\mathcal{A}_1$ *is a* proper ($\pi \times \tau$)-defeater *for* $\mathcal{A}_0$*, denoted* $\mathcal{A}_1 \succ_{\pi\times\tau} \mathcal{A}_0$ *iff*

1. $\mathcal{A}_1 \succ_\pi \mathcal{A}_0$*, i.e.* $\mathcal{A}_1$ *is a* proper possibilistic defeater*, or*
2. $\mathcal{A}_1 \preceq\succeq_\pi \mathcal{A}_0$*, i.e.* $\mathcal{A}_1$ *is a* blocking possibilistic defeater *for* $\mathcal{A}_0$*, and* $\mathcal{A}_1 \succ_\tau \mathcal{A}_0$*, i.e.* $\mathcal{A}_1$ *is a* proper temporal defeater *for* $\mathcal{A}_0$*.*

We say that $\mathcal{A}_1$ *is a* blocking ($\pi \times \tau$)-defeater *for* $\mathcal{A}_0$*, denoted* $\mathcal{A}_1 \preceq\succeq_{\pi\times\tau} \mathcal{A}_0$ *iff* $\mathcal{A}_1 \preceq\succeq_\pi \mathcal{A}_0$ *and* $\mathcal{A}_1 \preceq\succeq_\tau \mathcal{A}_0$*. The definition of proper and blocking* ($\tau \times \pi$)*-defeater is completely analogous and is obtained by replacing* τ *with* π *and vice versa.*

The argumentation semantics for pt-DeLP inherits the one of DeLP (as well as those of t-DeLP and P-DeLP) based on dialectical trees, with slight modifications in each case. The following definitions actually are parametric with respect to the notion of proper and blocking defeater, so they can be instantiated in any of the ($\tau \times \pi$) or ($\pi \times \tau$) criteria. Of course, depending on whether we prioritize the temporal defeat criteria or the comparison among weights, we will obtain distinct notions of warrant.

Definition 13 (Argumentation Line, Dialectical Tree, Marking). *Let* $\mathcal{A}_1$ *be an argument in* (Π, Δ)*. An* argumentation line *for* $\mathcal{A}_1$ *is a sequence* $\Lambda = [\mathcal{A}_1, \mathcal{A}_2, \ldots]$ *where:*

(i) *supporting arguments, i.e. those in odd positions* $\mathcal{A}_{2i+1} \in \Lambda$ *are jointly consistent with* Π*, and similarly for interfering arguments* $\mathcal{A}_{2i} \in \Lambda$ *;*
(ii) *a supporting (resp. interfering) argument is different from the attacked sub-arguments of previous supporting (resp. interfering) arguments:* $\mathcal{A}_{i+2k} \neq \mathcal{A}_i(\neg\mathsf{concl}(\mathcal{A}_{i+1}))$*;*
(iii) $\mathcal{A}_{i+1}$ *is a proper defeater for* $\mathcal{A}_i$ *if* $\mathcal{A}_i$ *is a blocking defeater for* $\mathcal{A}_{i-1}$*.*

An argumentation line $[\mathcal{A}_1, \ldots, \mathcal{A}_n]$ *for* $\mathcal{A}_1$ *is* maximal *if there is no other argument* $\mathcal{A}_{n+1}$ *such that* $[\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}_{n+1}]$ *is an argumentation line for* $\mathcal{A}_1$*.*

The dialectical tree *for* $\mathcal{A}_1$ *is the set of maximal argumentation lines rooted in* $\mathcal{A}_1$ *arranged in the form of a tree, and is denoted* $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ *(see [13] for more details). The bottom-up* marking *procedure of a dialectical tree* $\mathcal{T}$ *is as follows:*

(1) mark all terminal nodes of $\mathcal{T}$ with a U (for undefeated);
(2) mark a node $\mathcal{B}$ with a D (for defeated) if it has a children node marked U;
(3) mark $\mathcal{B}$ with U if all its children nodes are marked D.

Finally, the notion of warranted literal is defined as follows. Notice that in the presence of weights it makes sense to actually consider two notions of warrant, the usual one and another one witnessing the highest weight with which a literal can be warranted, see e.g. [3,8].

Definition 14 (Warrant, Strong Warrant). *Given a pt-DeLP program (Π, Δ) and a query temporal literal (ℓ, t), we say (ℓ, t) is warranted in (Π, Δ), denoted $(\ell, t) \in \mathsf{warr}(\Pi, \Delta)$, if, for some $r > 0$, there exists an argument $\mathcal{A}$ for $(\ell, t)_r$ such that $\mathcal{A}$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$. In case there is no value $s > r$ such that $(\ell, t)_s$ is also warranted, we say that $(\ell, t)_r$ is strongly warranted in (Π, Δ), denoted $(\ell, t)_r \in \mathsf{swarr}(\Pi, \Delta)$.*

It should be noted that, as expected, the set $\mathsf{warr}(\Pi, \Delta)$ is always consistent, even with the $(\tau \times \pi)$ defeat relation, but its consistency with the strict part of the program Π cannot be guaranteed.

4 Relating pt-DeLP to t-DeLP and P-DeLP

Now we proceed to study how pt-DeLP relates to each of the frameworks t-DeLP and P-DeLP. To this end, we first propose a translation between the respective languages.

Definition 15. *We define the translation maps f and g from, respectively, the language of t-DeLP and P-DeLP into that of pt-DeLP. These maps are the following:*

f :	t-DeLP		pt-DeLP
fact	(ℓ, t)	$\longmapsto$	$(\ell, t)_1$
rule	$(\ell, t) \longleftarrow (\ell_1, t_1), \ldots, (\ell_n, t_n)$	$\longmapsto$	$\langle (\ell, t) \longleftarrow (\ell_1, t_1), \ldots, (\ell_n, t_n) \rangle_r$
			where $\begin{cases} r = 1 & \text{if } \delta \in \Pi \\ r = .5 & \text{if } \delta \in \Delta \end{cases}$
g :	P-DeLP	$\longmapsto$	pt-DeLP
fact	$(\ell)_r$	$\longmapsto$	$(\ell, 0)_r$
rule	$\langle \ell \longleftarrow \ell_1, \ldots, \ell_n \rangle_r$	$\longmapsto$	$\langle (\ell, 0) \longleftarrow (\ell_1, 0), \ldots, (\ell_n, 0) \rangle_r$

If P is a set of t-DeLP formulas, we will denote its translation by f by $f[P]$ or simply fP, and analogously with g. Note that both mappings f and g preserve the strict or defeasible character of facts and rules (in the case of g the weight is also preserved).

Lemma 1. *For arbitrary arguments $\mathcal{A}, \mathcal{B}$ in a t-DeLP program P, we have $\mathcal{A} \succ_\tau \mathcal{B}$ iff $f[\mathcal{A}] \succ_\tau f[\mathcal{B}]$ iff $f[\mathcal{A}] \succ_{\tau\times\pi} f[\mathcal{B}]$ iff $f[\mathcal{A}] \succ_{\pi\times\tau} f[\mathcal{B}]$. The case of $\preceq\succeq_\tau$, $\preceq\succeq_{\tau\times\pi}$ and $\preceq\succeq_{\pi\times\tau}$ is similar.*

$$\begin{tikzcd}
 & \mathsf{warr}(\Pi,\Delta) \arrow[dr, "f"] & \\
(\Pi,\Delta) \arrow[ur, "\text{t-DeLP}"] \arrow[dr, "f"'] & & f[\mathsf{warr}(\Pi,\Delta)] = \mathsf{warr}(f\Pi,f\Delta) \\
 & (f\Pi,f\Delta) \arrow[ur, "\text{PT-DeLP}"'] &
\end{tikzcd}$$

Fig. 1. A representation of pt-DeLP being a conservative extension of t-DeLP under the translation map f, and for any of the two criteria $(\tau \times \pi)$ and $(\pi \times \tau)$

Proof. (Sketch) The idea is to use the fact that conflicting arguments are nonetheless consistent with Π, so they must both be of degree .5. This makes the degrees collapse and hence makes the $\preceq\succeq_\pi$ always true. Thus, the result is determined purely by $\succ_\tau$ and $\preceq\succeq_\tau$ relations.

Lemma 1 fails for the mapping g between P-DeLP and pt-DeLP, as shown in Example 1 below. From now on, the results focus then on the relation between t-DeLP and pt-DeLP. Indeed, if we ignore the notational differences between

– t-DeLP (being strictly vs. defeasibly derivable), and
– pt-DeLP with set of degrees $\{0, .5, 1\}$ (i.e. being derivable with degree .5 vs. with degree 1)

we can define the conditions for pt-DeLP to be a conservative extension of t-DeLP as follows: *the* $\mathsf{warr}(\cdot,\cdot)$ *operator commutes with* f; see Figure 1.

Proposition 1. *pt-DeLP is a conservative extension of t-DeLP, under both lexicographic orderings* $(\tau \times \pi)$ *or* $(\pi \times \tau)$.

Proof. (Sketch) The proof proceeds by using Lemma 1 and showing by induction that the dialectical trees $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$ and $\mathcal{T}_{(f\Pi,f\Delta)}(f[\mathcal{A}])$ are isomorphic.

In contrast to this preservation result for t-DeLP and pt-DeLP, the mapping g fails to guarantee the warrant literals from P-DeLP to pt-DeLP for both lexicographic criteria, as the following example shows.

Example 1. Consider the P-DeLP-program (Π,Δ) containing only the literals and rules mentioned in the arguments below.

$$\mathcal{A} = \{(p)_1; \langle q \longleftarrow p\rangle_{.8}\} \text{ with conclusion } (q)_{.8}$$
$$\mathcal{B} = \{(p)_1; (r)_1; \langle \neg q \longleftarrow p, r\rangle_{.8}\} \text{ with conclusion } (\neg q)_{.8}$$

Clearly, $\mathcal{A}$ and $\mathcal{B}$ attack each other, and are blocking defeaters for each other. The dialectical tree for $\mathcal{A}$ is $[\mathcal{A},\mathcal{B}]$ which makes $\mathcal{A}$ defeated. Since this is the only argument for $(q)_{.8}$, we have $(q)_{.8} \notin \mathsf{warr}(\Pi,\Delta)$. The tree for $\mathcal{B}$ is $[\mathcal{B},\mathcal{A}]$, and we also have that $(\neg q)_{.8} \notin \mathsf{warr}(\Pi,\Delta)$. On the other hand, if we translate them into pt-DeLP as $\mathcal{A}' = g[\mathcal{A}]$ and $\mathcal{B}' = g[\mathcal{B}]$, we find that:

1. Case ($\tau \times \pi$): From $\mathcal{B}' \succ_\tau \mathcal{A}'$, we infer $\mathcal{B}' \succ_{\tau\times\pi} \mathcal{A}'$, so $\mathcal{T}_{(g\Pi, g\Delta)}(\mathcal{B}') = [\mathcal{B}']$ and $g(\mathsf{concl}(\mathcal{B})) = (\neg q, 0)_{.8} \in \mathsf{warr}(g\Pi, g\Delta)$.
2. Case ($\pi \times \tau$): Since $\mathcal{A}' \preceq\succeq_\pi \mathcal{B}'$, we try the temporal criteria which as before gives $\mathcal{B}' \succ_{\pi\times\tau} \mathcal{A}'$ from $\mathcal{B}' \succ_\tau \mathcal{A}'$. As before, this makes $\mathcal{B}'$ undefeated and $\mathcal{A}'$ defeated (in the corresponding trees), so $(\neg q, 0)_{.8} \in \mathsf{warr}(g\Pi, g\Delta)$.

Thus, for both criteria, pt-DeLP is not a conservative extension of P-DeLP.

5 An Example

Consider the following scenario:

Lars, a tourist visiting the Snake Forest, just got bitten *by a poisonous snake. Normally, the bite of this kind of snake increases the likelihood of* fast poisoning*, whose maximum degree is reached 10 hours after the bite. On the other hand, Lars (who, as an experienced tropical tourist, has been bitten a few times before) has developed some resistance to the poison. So this poison increases instead his likelihood of* slow poisoning*, the maximum degree of likelihood for this being reached 20 hours after the bite. In any case, once bitten, the actual occurrence of slow or fast poisoning causes* death *immediately. Assuming Lars arrives alive to the* hospital *and he is given an antidote, the likelihood that Lars survives the next k hours depends on how much time passed between the bite and the antidote.*

So we decide to take Lars to the nearest hospital, which normally takes 8 hours. But, the radio just announced we will find a traffic jam*, causing a delay of 4 more hours. Thus, most plausibly it will take us 12 hours to reach the hospital. The problem is then to compute at time* 0 *(now) how strong is our belief of whether Lars will die (or not) at some later time t in this scenario.*

This example extends a similar scenario considered in [17], where only the amount and relevance of temporal information was considered. The possibilistic degrees allow for more smooth descriptions of the causal relations considered.

The scenario is formalized in Figure 2. The strict facts denote the factual knowledge about the initial state (where the reasoning takes place). The strict rules describe the effect of two possible ways in which the bite can lead to death by poisoning (one way would be faster than the other). In either case, the immediate effect is death. A faster poisoning does occur in normal people, while certain resistance can be acquired with experience, giving some extra temporary resistance (slow poisoning). The defeasible rules δ_2, δ_3 describe the likelihood of, resp., faster and slower poisoning across time, provided Lars has been bitten (resp. and experienced). This likelihood is represented in Figure 3 (Top). The X-axis represents time (from now, 0, to 20 hours into the future). The Y-axis describes the degree for derivability of slow and faster poisoning (i.e. death) at each time t. The arguments built in this program are also shown in Figure 4. The first and third arguments are those built from δ_2 (in grey, denoting defeated) and δ_3 (white, denoting undefeated). In order to prevent conclusions that are too pessimistic (about Lars' chances) using δ_2, a canceling rule δ_4 is introduced. So the argument for δ_3 can impose the (correct) degree $\alpha_3(\mathsf{d}_0)(= \alpha_3(t-0) = \alpha_3(t))$,

Π − facts

$$\left\{\begin{matrix}(\textsf{@forest}(\mathrm{Lars}),0)_1, & (\textsf{bitten}(\mathrm{Lars}),0)_1, \\ (\neg\textsf{dead}(\mathrm{Lars}),0)_1, & (\textsf{exp}(\mathrm{Lars}),0)_1\end{matrix}\right\}$$

Π − rules

$$\langle \textsf{dead}(\mathrm{Lars}) \longleftarrow \textsf{fast.poisoning}(\mathrm{Lars}),0)\rangle_1 \qquad \delta_0$$

$$\langle \textsf{dead}(\mathrm{Lars}) \longleftarrow \textsf{slow.poisoning}(\mathrm{Lars}),0)\rangle_1 \qquad \delta_1$$

Δ

$$\langle \textsf{fast.poisoning}(\mathrm{Lars}) \longleftarrow ((\textsf{bitten}(\mathrm{Lars}),\mathtt{d}_0))\rangle_{\alpha_2(\mathtt{d}_0)} \qquad \delta_2$$

where $\alpha_2(\mathtt{d}_0)$ is as in Fig. 3(Top).

$$\langle \textsf{slow.poisoning}(\mathrm{Lars}) \longleftarrow ((\textsf{bitten}(\mathrm{Lars}),\mathtt{d}_0)),(\textsf{exp}(\mathrm{Lars}),\mathtt{d}_1)\rangle_{\alpha_3(\mathtt{d}_0)} \qquad \delta_3$$

where $\alpha_3(\mathtt{d}_0)$ is as in Fig. 3(Top).

$$\langle \neg\textsf{fast.poisoning}(\mathrm{Lars}) \longleftarrow ((\textsf{bitten}(\mathrm{Lars}),\mathtt{d}_0)),(\textsf{exp}(\mathrm{Lars}),\mathtt{d}_1)\rangle_{\alpha_4(\mathtt{d}_0)} \qquad \delta_4$$

where $\alpha_4(\mathtt{d}_0) = \alpha_3(\mathtt{d}_0) + .001$

$$\langle \textsf{@hospital}(\mathrm{Lars}) \longleftarrow (\textsf{bitten}(\mathrm{Lars}),\mathtt{d}_0),(\textsf{@forest}(\mathrm{Lars}),\mathtt{d}_1)\rangle_{\alpha_5(\mathtt{d}_1)} \qquad \delta_5$$

$$\text{where } \alpha_5(\mathtt{d}_1) = \begin{cases} .9 & \text{if } \mathtt{d}_1 = 8 \\ 0 & \text{otherwise}\end{cases}$$

$$\langle \neg\textsf{@hospital}(\mathrm{Lars}) \longleftarrow (\textsf{bitten}(\mathrm{Lars}),\mathtt{d}_0),(\textsf{@forest}(\mathrm{Lars}),\mathtt{d}_1),(\textsf{traffic.jam},\mathtt{d}_2)\rangle_{\alpha_6(\mathtt{d}_1)} \qquad \delta_6$$

$$\text{where } \alpha_6(\mathtt{d}_1,\mathtt{d}_2) = \begin{cases} .95 & \text{if } \mathtt{d}_1 = 8 = \mathtt{d}_2 \\ 0 & \text{otherwise}\end{cases}$$

$$\langle \neg\textsf{slow.poisoning}(\mathrm{Lars}) \longleftarrow (\textsf{bitten}(\mathrm{Lars}),\mathtt{d}_0),\textsf{@hospital}(\mathrm{Lars}),\mathtt{d}_1),(\textsf{exp}(\mathrm{Lars}),\mathtt{d}_2)\rangle_{\alpha_7} \quad \delta_7$$

$$\text{where } \alpha_7(\mathtt{d}_0,\mathtt{d}_1) = \begin{cases} \alpha_3(\mathtt{d}_0) + .001 & \text{if } \mathtt{d}_1 < \mathtt{d}_0 \\ 0 & \text{otherwise}\end{cases}$$

$$\langle \textsf{@hospital}(\mathrm{Lars}) \longleftarrow (\textsf{bitten}(\mathrm{Lars}),\mathtt{d}_0),(\textsf{@forest}(\mathrm{Lars}),\mathtt{d}_1),(\textsf{traffic.jam},\mathtt{d}_2)\rangle_{\alpha_6(\mathtt{d}_1,\mathtt{d}_2)} \qquad \delta_8$$

$$\text{where } \alpha_6(\mathtt{d}_1,\mathtt{d}_2) = \begin{cases} .95 & \text{if } \mathtt{d}_1 = 12 = \mathtt{d}_2 \\ 0 & \text{otherwise}\end{cases}$$

$$\langle(\textsf{dead}(\mathrm{Lars}) \longleftarrow (\textsf{bitten}(\mathrm{Lars}),\mathtt{d}_0),(\textsf{@hospital}(\mathrm{Lars}),\mathtt{d}_1),(\textsf{exp}(\mathrm{Lars},\mathtt{d}_2))\rangle_{\alpha_9(\mathtt{d}_0,\mathtt{d}_1)} \qquad \delta_9$$

$$\text{where } \alpha_9(\mathtt{d}_0,\mathtt{d}_1) = \begin{cases} \alpha_3(\mathtt{d}_1 - |\mathtt{d}_0 - \mathtt{d}_1|) & \text{if } \mathtt{d}_1 < \mathtt{d}_0 \\ 0 & \text{otherwise}\end{cases}$$

Fig. 2. The list of strict facts and rules, and defeasible rules for the example

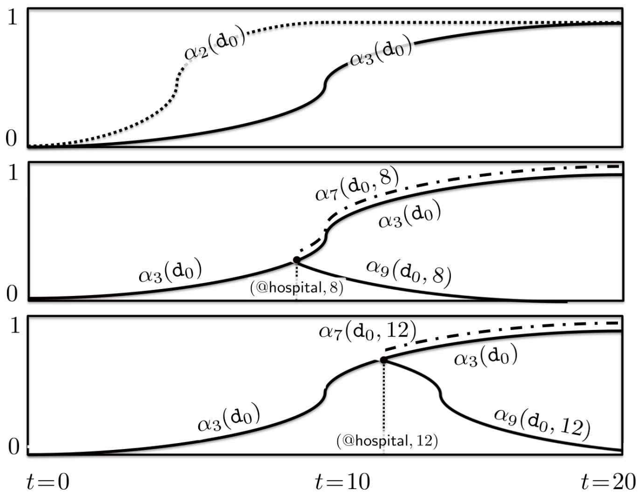


Fig. 3. The Snake-Bites-Lars example. (Top) A comparison between the likelihood of fast-poisoning (resp. slow-poisoning) in dashed arrow (resp. solid arrow). (Mid) The likelihood of Lars dying at t, if we reach the hospital at time 8; this is described by α_3 -before reaching the hospital-, and by α_9 -afterwards. (Bottom) Similarly for the case we reach the hospital at time 12, which is the actual case.

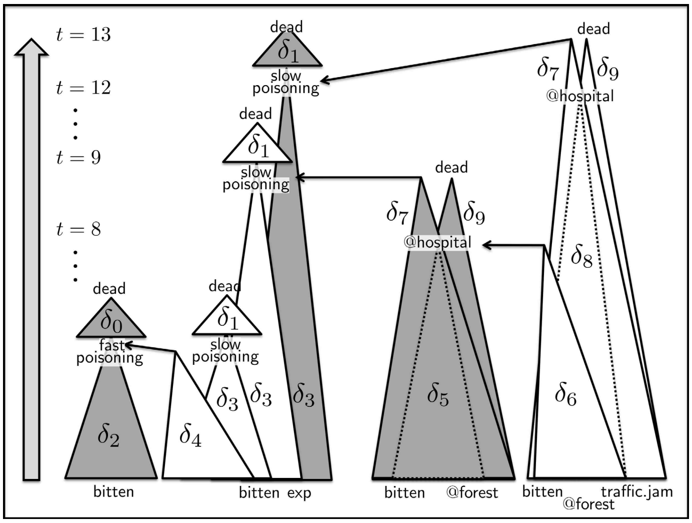


Fig. 4. An illustration of the example from Section 5 in terms of defeaters. Triangles represent arguments, with the conclusion on top, and the base on bottom. White triangles are undefeated arguments, grey arguments are defeated. The arrows denote the defeat relations. Conclusions attacking their negations are left blank. For both criteria $(\tau \times \pi)$ and $(\pi \times \tau)$, the output $\mathsf{swarr}(\Pi, \Delta)$ includes $(\mathsf{dead}(\mathrm{Lars}), t)_{\alpha_3}$ for $t < 12$ and $(\mathsf{dead}(\mathrm{Lars}), t)_{\alpha_9}$, for $t \geq 12$, as well as $(\mathsf{@hospital}(\mathrm{Lars}), t)_{.95}$.

correct at least up to the point where we reach the hospital. See in Figure 4 how the counter-argument based on δ_4 acts against δ_2 at a particular time point, to let δ_3-arguments succeed.

After we reach the hospital, the previously correct α_3 becomes too pessimistic, since it does not take into account the antidote, and prevents the correct degrees to surface (the lower degrees described by α_9). To this end, δ_3-based arguments are to be counter-argued by arguments with δ_7. Now, arguments containing δ_9 or δ_7 depend of course on the time t we reach the hospital: we have arguments for $t = 8$ and for $t = 12$, though the latter is better supported. See Figure 3 (Mid) for the $t = 8$ case and (Bottom) for the (correct) case $t = 12$. Thus, the output $\mathsf{swarr}(\mathit{\Pi}, \mathit{\Delta})$ includes $(\mathsf{dead}(\text{Lars}), t)_{\alpha_3(t)}$ for $t < 12$ and $(\mathsf{dead}(\text{Lars}), t)_{\alpha_9(t)}$, for $t \geq 12$, as well as $(@\mathsf{hospital}(\text{Lars}), t)_{.95}$. These output results from either criteria $(\tau \times \pi)$ and $(\pi \times \tau)$. So we could tell Lars that he will not be in much danger most of the time.

6 Related Work and Conclusions

We presented a new argumentation-based logic programming formalism that combines previous work on temporal reasoning, on the one side, and possibilistic reasoning on the other. This is done by taking the lexicographic product of the defeat criteria of these two systems. We have studied two notions of warrant (with and without degrees). As a result, we have shown that the combined system extend the temporal defeasible framework but not the possibilistic one.

Many different approaches exist in the area of temporal reasoning. Within the more specific field of defeasible logics (for non-monotonic reasoning), one can point to the initially proposed rule-based systems [6,16], recently extended in [14]. Since [12], though, much work has been done on argumentation-based systems, for deliberative agents who have reasons for and against claims. Most works in this area that address temporal argumentation, though, do so by associating time intervals to literals and arguments [15,5,9]. The present pointwise approach, based on [17], makes t-DeLP (or pt-DeLP) simpler than these while keeping enough expressive power. On the other hand, several works on possibilistic defeasible logic programming can be mentioned, e.g. [2,3,8]. We have assumed part of this work in the present possibilistic temporal approach. Although as far as we know, combining possibilistic and temporal defeasible argumentation is novel, the issue of combining time and possibilistic logic had already been addressed in [10].

As for future work, we would consider the *rationality postulates* of [7], which specify reasonable constraints on the logical behavior of the warrant operator. We expect that some partial solution to this problem would come by trying to extend the results for the temporal case in [17]. A more radical option would be to adopt a recursive semantics (instead of the current based on dialectical trees) like in [1] that would ensure the indirect consistency postulate.

Acknowledgments. The authors are grateful to the anonymous reviewers for their insightful comments. They also acknowledge partial support by the Spanish projects *Agreement Technologies* (CONSOLIDER CSD2007-0022, INGENIO

2010), ARINF (TIN2009-14704-C03-03) and TASSAT (TIN2010- 20967-C04-01), as well as by the Marie Curie IRSES Project MaToMuVI (FP7-PEOPLE-2009).

References

1. Alsinet, T., Béjar, R., Godo, L.: A characterization of collective conflict for defeasible argumentation. In: Proc. of COMMA 2010, pp. 27–38 (2010)
2. Alsinet, T., Chesñevar, C.I., Godo, L., Sandri, S., Simari, G.R.: Formalizing argumentative reasoning in a possibilistic logic programming setting with fuzzy unification. Int. J. Approx. Reasoning 48(3), 711–729 (2008)
3. Alsinet, T., Chesñevar, C.I., Godo, L., Simari, G.R.: A Logic Programming Framework for Possibilistic Argumentation: Formalization and Logical Properties. Fuzzy Sets and Systems 159, 1208–1228 (2008)
4. Andréka, H., Ryan, M., Schobbens, P.Y.: Operators and laws for combining preference relations. J. of Logic and Computation 12(1), 13–53 (2002)
5. Augusto, J.: Simari G. R. Temporal Defeasible Reasoning. Knowledge and Information Systems 3, 287–318 (2001)
6. Billington, D.: Defeasible logic is stable. J. of Logic and Computation 3, 379–400 (1993)
7. Caminada, M., Amgoud, L.: On the evaluation of argumentation formalisms. Artificial Intelligence 171, 286–310 (2007)
8. Capobianco, M., Simari, G.R.: A Proposal for Making Argumentation Computationally Capable of Handling Large Repositories of Uncertain Data. In: Godo, L., Pugliese, A. (eds.) SUM 2009. LNCS, vol. 5785, pp. 95–110. Springer, Heidelberg (2009)
9. Cobo, L., Martínez, D., Simari, G.R.: On Admissibility in Timed Abstract Argumentation Frameworks. In: Proc. of ECAI 2010, pp. 1007–1008. IOS Press (2010)
10. Dubois, D., Lang, J., Prade, H.: Timed possibilistic logic. Fundamenta Informaticae 15(3-4), 211–234 (1991)
11. Dubois, D., Prade, H.: Possibilistic logic: a retrospective and prospective view. Fuzzy Sets and Systems 144(1), 3–23 (2004)
12. Dung, P.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77(2), 321–357 (1995)
13. García, A., Simari, G.R.: Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Programming 4(1+2), 95–138 (2004)
14. Governatori, G., Terenziani, P.: Temporal Extensions to Defeasible Logic. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 476–485. Springer, Heidelberg (2007)
15. Mann, N., Hunter, A.: Argumentation Using Temporal Knowledge. In: Proc. of COMMA 2008, pp. 204–215. IOS Press (2008)
16. Nute, D.: Defeasible logic. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3, pp. 353–395. Oxford University Press (1994)
17. Pardo, P., Godo, L.: t-DeLP: A Temporal Extension of the Defeasible Logic Programming Argumentative Framework. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 489–503. Springer, Heidelberg (2011)

On Decidability of a Logic for Order of Magnitude Qualitative Reasoning with Bidirectional Negligibility

Joanna Golińska-Pilarek

Institute of Philosophy, University of Warsaw, Poland
j.golinska@uw.edu.pl
http://www.joannagolinska.com

Abstract. Qualitative Reasoning (QR) is an area of research within Artificial Intelligence that automates reasoning and problem solving about the physical world. QR research aims to deal with representation and reasoning about continuous aspects of entities without the kind of precise quantitative information needed by conventional numerical analysis techniques. Order-of-magnitude Reasoning (OMR) is an approach in QR concerned with the analysis of physical systems in terms of relative magnitudes. In this paper we consider the logic $\mathsf{OMR_N}$ for order-of-magnitude reasoning with the bidirectional negligibility relation. It is a multi-modal logic given by a Hilbert-style axiomatization that reflects properties and interactions of two basic accessibility relations (strict linear order and bidirectional negligibility). Although the logic was studied in many papers, nothing was known about its decidability. In the paper we prove decidability of $\mathsf{OMR_N}$ by showing that the logic has the strong finite model property.

Keywords: multi-modal logic, qualitative reasoning, order-of-magnitude reasoning, bidirectional negligibility, knowledge representation, decidability.

1 Introduction

The problem of knowledge representation and reasoning is recognized as one of the central topics of Artificial Intelligence. Qualitative Reasoning (QR) has emerged as a novel alternative for the representation of knowledge in AI thirty years ago to deal with representation and reasoning about continuous aspects of entities and systems in a symbolic, but human-like manner. One of the greatest advantages of qualitative approach is that, given only vague or incomplete qualitative information at any of the represented levels of abstraction, a qualitative approach will infer as much as possible according to distinctions within the model. In contrast, a numerical approach requires information to be expressed as unit values, and hence the processing capability within a numerical approach is limited if the quantities are not available. Qualitative reasoning methods have been foremost applied to modeling problems in cognitive science, information technology, and engineering domains. Further possible and existing areas of applications include: natural language semantics and processing, linguistics, economics and decision support systems, ecology and bioinformatics, robotics, and education.

Order-of-magnitude Reasoning (OMR) is an approach in the field of qualitative reasoning. The order-of-magnitude approach enables us to reason in terms of relative magnitudes of variables obtained by comparisons of the sizes of quantities ([9]). OMR methods of reasoning are situated midway between numerical methods and purely qualitative formalisms. There are two main types of OMR approaches: Absolute Order of Magnitude (AOM) and Relative Order of Magnitude (ROM) models. The differences among both types of OMR approaches are as follows: AOM is based on a partition of the real line, in which each element of the real line belongs to a qualitative class, whereas ROM reasoning introduces a family of binary order of magnitude relations, among which are comparability, negligibility, and closeness relations. The first multi-modal logic for order-of-magnitude reasoning, with comparability relation based on Absolute Order of Magnitude, was introduced in [1]. Later on, this logic was extended to cope other magnitude relations such as negligibility ([2]), bidirectional negligibility ([3]), and non-closeness and distance ([4]). For all these logics dual tableau systems have been constructed as shown in papers [6] and [7] (cf. [8]). Some of these systems have been implemented in PROLOG (see [5]).

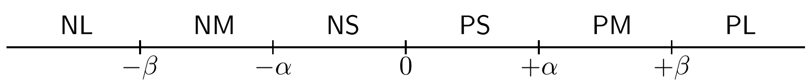
Dual tableau systems for order-of-magnitude logics in question provide a means for almost automated reasoning, but they do not provide decision procedures. In fact the problem whether the logics for order-of-magnitude reasoning are decidable is still open. The aim of the paper is to give a partial solution to this problem. We focus on the logic for order-of-magnitude reasoning with bidirectional negligibility, $\mathsf{OMR_N}$, introduced in [3]. We prove that this logic has the finite model property, hence it is decidable.

The paper is organized as follows. In Section 2 we present the logic $\mathsf{OMR_N}$, its syntax, axiomatization, and semantics. In Section 3 we prove its decidability by showing that it has the strong finite model property. Final remarks and prospects of future work are described in Section 4.

2 Logic $\mathsf{OMR_N}$

In this section we present the logic $\mathsf{OMR_N}$ for order-of-magnitude qualitative reasoning with bidirectional negligibility, its Hilbert-style axiomatization, and semantics. In the presentation of the logic we follow [3].

The logic $\mathsf{OMR_N}$ is based on the Absolute Order of Magnitude model AOM(5) on which the new binary relation of *bidirectional negligibility* is defined. Recall that in AOM(5) the real line is divided into seven equivalences classes with five landmarks:



where α, β are two positive real numbers such that $\alpha < \beta$, for the usual ordering $<$ on real numbers. There are seven equivalences classes above: NL, NM, NS, $[0]$, PS, PM, and PL. The labels correspond to 'negative large', 'negative medium', 'negative small', 'zero', 'positive small', 'positive medium', and 'positive large', respectively. Given positive real numbers α and β such that $0 < \alpha < \beta$, an element x is said to be *negligible* with respect to y (xNy for short) if and only if either $x = 0$ or both $x \in \mathsf{NS} \cup \mathsf{PS}$ and $y \in \mathsf{NL} \cup \mathsf{PL}$. Thus, 0 is negligible with respect to any real number and each number sufficiently small is negligible with respect to any number sufficiently large, independently of their signs. It follows that the relation N is not a restriction of $<$, hence the relation N is referred to as *bidirectional*.

The logic $\mathsf{OMR_N}$ is the multi-modal logic over two basic accessibility relations together with their converses. The two basic relations are: the strict linear ordering and bidirectional negligibility. More precisely, the vocabulary of the language of $\mathsf{OMR_N}$ consists of the following pairwise disjoint sets symbols:

- $\mathbb{V}$ - a countably infinite set of propositional variables
- $\mathbb{C} = \{c_1, \ldots, c_5\}$ - the set of propositional constants
- $\{\neg, \vee, \wedge, \rightarrow\}$ - the set of classical propositional operations of negation, disjunction, conjunction, and implication, respectively
- $\{[R], [\overline{R}], [N], [\overline{N}]\}$ - the set of unary modal propositional operations of necessity with respect to the accessibility relation R, converse of R, the negligibility relation N, and its converse, respectively.

As usual in modal logics, the set of $\mathsf{OMR_N}$-*formulas* is defined as the smallest set that includes all the propositional variables and constants and is closed on the $\mathsf{OMR_N}$-operations. We will write $\langle T \rangle$ as abbreviations of $\neg[T]\neg$, for any $T \in \{R, \overline{R}, N, \overline{N}\}$. The propositional constants c_1, c_2, c_3, c_4, c_5 are logical counterparts of the distinguished landmarks $-\beta, -\alpha, 0, +\alpha, +\beta$, respectively, from $\mathsf{AOM}(5)$-model. The intuitive meanings of formulas with the modal connectives $[N]$ and $[\overline{N}]$ are as follows:

- $[N]\varphi$ means φ holds in all states with respect to which the current one is negligible.
- $[\overline{N}]$ means φ holds in all states which are negligible with respect to the current one.

The Hilbert-style axiomatization of $\mathsf{OMR_N}$ consists of all the tautologies of the classical propositional logic together with the axiom schemas and rules listed below.

Axioms for propositional constants, for $i \in \{1, \ldots, 5\}$, $j \in \{1, \ldots, 4\}$:

(C1) $\langle \overline{R} \rangle c_i \vee c_i \vee \langle R \rangle c_i$
(C2) $c_i \rightarrow ([\overline{R}]\neg c_i \wedge [R]\neg c_i)$
(C3) $c_j \rightarrow \langle R \rangle c_{j+1}$

The axioms (C1)-(C3) reflect the existence, uniqueness, and ordering of landmarks.

Axioms for modal operations:

(Ax1) $[T](\varphi \to \psi) \to ([T]\varphi \to [T]\psi)$, for $T \in \{R, N\}$

(Ax2) $\varphi \to [T]\langle T' \rangle\varphi$, for $T \in \{R, \overline{R}, N, \overline{N}\}$ and $T' = \begin{cases} \overline{R}, & \text{if } T = R \\ R, & \text{if } T = \overline{R} \\ \overline{N}, & \text{if } T = N \\ N, & \text{if } T = \overline{N} \end{cases}$

(Ax3) $[R]\varphi \to [R][R]\varphi$

(Ax4) $[[R](\varphi \vee \psi) \wedge [R]([R]\varphi \vee \psi) \wedge [R](\varphi \vee [R]\psi)] \to ([R]\varphi \vee [R]\psi)$

(Ax5) $([\overline{R}]\varphi \wedge \varphi \wedge [R]\varphi) \to [N]\varphi$

(Ax6) $(\langle R \rangle c_2 \vee \langle \overline{R} \rangle c_4) \to [N](\varphi \wedge \neg\varphi)$

(Ax7) $c_3 \to ([N]\varphi \to ([\overline{R}]\varphi \wedge \varphi \wedge [R]\varphi))$

(Ax8) $(\neg c_3 \wedge (c_2 \vee (\langle \overline{R} \rangle c_2 \wedge \langle R \rangle c_4) \vee c_4)) \to [N](\langle R \rangle c_1 \vee \langle \overline{R} \rangle c_5)$

(Ax9) $(\neg c_3 \wedge (c_2 \vee (\langle \overline{R} \rangle c_2 \wedge \langle R \rangle c_4) \vee c_4)) \to$
$([N]\varphi \to ([\overline{R}](\langle R \rangle c_1 \to \varphi) \wedge [R](\langle \overline{R} \rangle c_5 \to \varphi)))$

Axiom (Ax1) is the usual modal axiom. Axiom (Ax2) expresses that $\overline{T}$ is the converse of T. Axiom (Ax3) reflects transitivity of R, while (Ax4) is related to its connectedness. Axiom (Ax5) says that N is the restriction of $R \cup \overline{R}$. Axiom (Ax6) expresses that neither large nor medium elements are negligible with respect to any element. Axiom (Ax7) means that 0 is negligible with respect to any element. Axioms (Ax8) and (Ax9) correspond to the property: $x \neq 0$ is negligible with respect to y if and only if x is small and y is large.

Rules of inference:

(MP) If $\vdash \varphi \to \psi$ and $\vdash \varphi$, then $\vdash \psi$.

(G1) If $\vdash \varphi$, then $\vdash [R]\varphi$.

(G2) If $\vdash \varphi$, then $\vdash [\overline{R}]\varphi$.

An $\mathsf{OMR_N}$-formula is said to be $\mathsf{OMR_N}$-*provable* whenever there exists its proof in the system described above. A sound and complete dual tableau system for the relational logic associated with $\mathsf{OMR_N}$ have been constructed in [7]. The dual tableau system can be used to verify whether a formula is $\mathsf{OMR_N}$-provable, but the system does not provide a decision procedure for $\mathsf{OMR_N}$, as it may generate infinite trees.

Now, we present the semantics for the logic $\mathsf{OMR_N}$. An $\mathsf{OMR_N}$-model is a structure $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$, where:

- U is a non-empty set of *states*
- R is a strict linear order on U, i.e., for all $x, y, z \in U$ the following hold:
  - (irref) $(x, x) \notin R$
  - (tran) If $(x, y) \in R$ and $(y, z) \in R$, then $(x, z) \in R$
  - (con) Either $(x, y) \in R$ or $(y, x) \in R$ or $x = y$
- $\overline{R}$ is the converse of R
- $m(p) \subseteq U$, for every propositional variable $p \in \mathbb{V}$
- $m(c_i) \in U$ and $(m(c_j), m(c_{j+1})) \in R$, for every $i \in \{1, \ldots, 5\}$ and for every $j \in \{1, \ldots, 4\}$

- N is a binary relation on U defined as $N \overset{\text{df}}{=} G_1 \cup G_2$:
  $G_1 = \{(x, y) \in U \times U \mid x = m(c_3)\}$
  $G_2 = \{(x, y) \in U \times U \mid (\lambda \text{ or } \mu) \text{ and } (\gamma \text{ or } \delta) \text{ and } (\zeta \text{ or } \eta)\}$, where
  $\lambda := ((m(c_2), x) \in R$, $\mu := (x = m(c_2))$,
  $\gamma := (((x, m(c_4)) \in R)$, $\delta := (x = m(c_4))$,
  $\zeta := ((y, m(c_1)) \in R)$ and $\eta := ((m(c_5), y) \in R)$
- $\overline{N}$ is the converse of N.

Let φ be an $\mathsf{OMR_N}$-formula and let $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$ be an $\mathsf{OMR_N}$-model. The *satisfaction of* φ *in* $\mathcal{M}$ *by a state* $s \in U$, ($\mathcal{M}, s \models \varphi$ for short), is defined as usual in modal logic. Recall that for modal operations it is defined as follows:

For $T \in \{R, \overline{R}, N, \overline{N}\}$,

- $\mathcal{M}, s \models [T]\varphi$ iff for all $t \in U$, $(s, t) \in R$ implies $\mathcal{M}, t \models \varphi$.

An $\mathsf{OMR_N}$-formula φ is *true* in an $\mathsf{OMR_N}$-model $\mathcal{M}$ whenever it is satisfied in $\mathcal{M}$ for all $s \in U$. An $\mathsf{OMR_N}$-formula φ is $\mathsf{OMR_N}$-*valid* whenever it is true in all $\mathsf{OMR_N}$-models. The following is stated in [3] (for the proof see [2]):

Theorem 1 (Soundness and Completeness). *For every* $\mathsf{OMR_N}$*-formula* φ*, the following conditions are equivalent:*

1. φ *is* $\mathsf{OMR_N}$*-valid.*
2. φ *is* $\mathsf{OMR_N}$*-provable.*

We finish this section with the small example of application of the logic $\mathsf{OMR_N}$ mentioned in [3]. For further detailed discussions on properties and application of the logic for order-of-magnitude reasoning with bidirectional negligibility see, for instance [2] and [3].

Example
Suppose we want to specify the behavior of a device to control automatically the temperature in a building so that some specific conditions hold. Let c_3 represents the temperature with value 0. If the temperature in a building must be closed to some limit t, then for practical purposes we may assume that any value of the interval $[t - \epsilon, t + \epsilon]$, for small ϵ, is admissible. The extreme points of this interval can be seen as the landmarks c_2 and c_4, respectively. Assume also that if the temperature is out of this interval, it is necessary to use 'normal' *heating* or *cooling* systems. Furthermore, we have another interval $[t - \lambda, t + \lambda]$ such that if the temperature is not in this interval, extra heating or extra cooling systems must be used, as the normal systems will not suffice. The extreme points of this interval can be seen as the landmarks c_1 and c_5, respectively. We assume also that there are two systems of humidification, normal and extra. The first system operates when normal systems of heating or cooling are used, while the second one in the case of extra heating or extra cooling. The intervals NL, NM, $\mathsf{NS} \cup [0] \cup \mathsf{PS}$, PM, PL can be interpreted as VERY-COLD, COLD, OK, HOT, VERY-HOT, respectively. Axioms that specify the general behavior of the whole system are:

OK $\to$ *off*	
COLD $\to$ *heat*	HOT $\to$ *cool*
VERY-COLD $\to$ *X-heat*	VERY-HOT $\to$ *X-cool*
(COLD $\vee$ HOT) $\to$ *hum*	(VERY-COLD $\vee$ VERY-HOT) $\to$ *X-hum*

Relations among the actions are expressed as:

$X\text{-}heat \to (\neg heat\ \wedge \neg off\ \neg cool\ \wedge \neg X\text{-}cool\ \wedge \neg X\text{-}hum\ )$
$heat \to (hum\ \wedge \neg X\text{-}cool\ \wedge \neg cool\ \wedge \neg off)$
$off \to (\neg hum\ \wedge \neg X\text{-}hum\ \wedge \neg X\text{-}cool\ \wedge \neg cool)$
$cool \to (hum\ \wedge \neg X\text{-}cool)$
$X\text{-}cool \to X\text{-}hum$
$hum \to (cool\ \vee heat)$
$X\text{-}hum \to \neg hum$

In the above *off* means that the system is off, and *cool* (resp. *heat*, *X-cool*, *X-heat*, *hum*, *X-hum*) means that the cooling (resp. heating, extra cooling, extra heating, humidification, extra humidification) system is used. The above specification together with axioms of $\mathsf{OMR_N}$ has the following consequences, among others:

$\langle\overline{R}\rangle c_2 \to (hum \vee X\text{-}hum)$
$(cool \wedge \langle\overline{R}\rangle c_2) \to [R](\neg\langle\overline{R}\rangle c_5 \to hum)$
$(off \wedge \neg c_3) \to [N]X\text{-}hum$
$X\text{-}hum \to [\overline{N}]off$

Now, let p represents 'the temperature is incremented (positively or negatively) with respect to the actual value'. As argued in [3] the following formula is then provable: $(\text{OK} \wedge \langle N\rangle p) \to (hum \vee X\text{-}hum)$. This formula says: if the temperature is OK and it is incremented in some value with respect to which the actual value is negligible, then humidification or extra humidification system must operate. Hence, the logic $\mathsf{OMR_N}$ can be used to verify some further properties of the whole system, which are not given explicitly by the specification conditions or axioms.

3 Decidability of $\mathsf{OMR_N}$

In this section we prove decidability of the logic $\mathsf{OMR_N}$ which so far was an open problem. The idea of the proof is as follows. First, we show that the logic $\mathsf{OMR_N}$ is sound and complete with respect to the class of weaker $\mathsf{OMR_N}$-models, which are called here quasi $\mathsf{OMR_N}$-models. Then, applying the filtration method, we show that each formula satisfiable in a quasi $\mathsf{OMR_N}$-model, is satisfiable in a finite quasi $\mathsf{OMR_N}$-model. This will yield the finite model property of logic $\mathsf{OMR_N}$, which by the classical results will imply its decidability. First, we introduce some useful notions and show some basic facts.

Let $\mathcal{K}$ be a class of Kripke models of the form $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$, where U is a non-empty set of states, R and N are any binary relations on U, $\overline{R}$ and $\overline{N}$ are converses of relations R and N, respectively, and m is the meaning function such that $m(p) \subseteq U$ and $m(c_i) \in U$, for every propositional variable p and for every $i \in \{1, \ldots, 5\}$. The satisfaction and the truth of an $\mathsf{OMR_N}$-formula are defined as in $\mathsf{OMR_N}$-models. We say that an $\mathsf{OMR_N}$-formula is $\mathcal{K}$-valid whenever it is true in all structures of $\mathcal{K}$.

Proposition 1 (Correspondence).

1. *If $\mathcal{K}$ is the class of models $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$ such that R is transitive, $(m(c_i), m(c_i)) \notin R$, and $(m(c_j), m(c_{j+1})) \in R$, for all $i \in \{1, \ldots, 5\}$ and $j \in \{1, \ldots, 4\}$, then the axioms* (C1), (C2), *and* (C3) *are $\mathcal{K}$-valid.*
2. *For every class $\mathcal{K}$, the axioms* (Ax1) *and* (Ax2) *are $\mathcal{K}$-valid.*
3. *If $\mathcal{K}$ is the class of structures $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$ with a transitive relation R, then the axiom* (Ax3) *is $\mathcal{K}$-valid.*
4. *If $\mathcal{K}$ is the class of structures $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$ with a connected relation R, then the axioms* (Ax4) *and* (Ax5) *are $\mathcal{K}$-valid.*
5. *If $\mathcal{K}$ is the class of structures $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$ such that R is transitive, N is defined as in* $\mathsf{OMR_N}$*-models, $(m(c_i), m(c_i)) \notin R$ and, in addition, $(m(c_j), m(c_{j+1})) \in R$, for all $i \in \{1, \ldots, 5\}$, $j \in \{1, \ldots, 4\}$, then the axiom* (Ax6) *is $\mathcal{K}$-valid.*
6. *If $\mathcal{K}$ is the class of structures $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$ such that N is defined as in* $\mathsf{OMR_N}$*-models, then the axioms* (Ax7), (Ax8), *and* (Ax9) *are $\mathcal{K}$-valid.*

For space reasons, we omit the proof of the above proposition.

A *quasi* $\mathsf{OMR_N}$*-model* is a structure $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$, where U is a non-empty set of states, R is a transitive and connected relation on U, $\overline{R}$, N, and $\overline{N}$ are defined as in $\mathsf{OMR_N}$-models, $m(p) \subseteq U$, for every propositional variable p, $m(c_i) \in U$, $(m(c_i), m(c_i)) \notin R$, and $(m(c_j), m(c_{j+1})) \in R$, for all $i \in \{1, \ldots, 5\}$ and $j \in \{1, \ldots, 4\}$. Quasi $\mathsf{OMR_N}$-models are referred to as $\mathsf{OMR_N^*}$-models. The satisfaction and the truth of a formula in an $\mathsf{OMR_N^*}$-model is defined as in $\mathsf{OMR_N}$-models. An $\mathsf{OMR_N}$-formula φ is said to be $\mathsf{OMR_N^*}$*-valid* whenever it is true in all $\mathsf{OMR_N^*}$-models. It is easy to see that every $\mathsf{OMR_N}$-model is an $\mathsf{OMR_N^*}$-model. Hence, we have:

Proposition 2. *If an* $\mathsf{OMR_N}$*-formula φ is* $\mathsf{OMR_N^*}$*-valid, then it is* $\mathsf{OMR_N}$*-valid.*

Now, we prove soundness of $\mathsf{OMR_N}$ with respect to all $\mathsf{OMR_N^*}$-models.

Proposition 3. *For every* $\mathsf{OMR_N}$*-formula φ, if φ is* $\mathsf{OMR_N}$*-provable, then it is* $\mathsf{OMR^*}$*-valid.*

Proof. We need to show that all the axioms are $\mathsf{OMR_N^*}$-valid and all the rules preserve $\mathsf{OMR_N^*}$-validity.

Note that in every $\mathsf{OMR_N^*}$-model $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$ the relation R is transitive and connected, the relations $\overline{R}$, N, $\overline{N}$ are defined as in $\mathsf{OMR_N}$-models, $(m(c_i), m(c_i)) \notin R$ and $(m(c_j), m(c_{j+1})) \in R$, for all $i \in \{1, \ldots, 5\}$ and $j \in \{1, \ldots, 4\}$. Thus, by Proposition 1, all the axioms are $\mathsf{OMR_N^*}$-valid. The proof of preserving $\mathsf{OMR_N^*}$-validity of the rules is straightforward. □

By Theorem 1, Proposition 2, and Proposition 3, we obtain:

Theorem 2. *For every* $\mathsf{OMR_N}$*-formula* φ*, the following conditions are equivalent:*

1. φ *is* $\mathsf{OMR_N}$*-valid.*
2. φ *is* $\mathsf{OMR_N^*}$*-valid.*
3. φ *is* $\mathsf{OMR_N}$*-provable.*

Therefore, the class of $\mathsf{OMR_N^*}$-models forms an alternative semantics for the logic $\mathsf{OMR_N}$. Now, we show that every $\mathsf{OMR_N^*}$-satisfiable formula φ is satisfiable in a finite $\mathsf{OMR_N^*}$-model. In the construction of a finite $\mathsf{OMR_N^*}$-model, we will use the filtration method.

Let φ be an $\mathsf{OMR_N^*}$-satisfiable formula. Define $\varphi^* \overset{\text{df}}{=} \varphi \wedge \bigwedge_{i\in\{1,\ldots,5\}}(c_i \to [R]\neg c_i)$. It is easy to see that the formula $c_i \to [R]\neg c_i$ is valid in all $\mathsf{OMR_N^*}$-models, for every $i \in \{1,\ldots,5\}$, as $(m(c_i), m(c_i)) \notin R$. Therefore, φ is $\mathsf{OMR_N^*}$-satisfiable if and only if φ^* is $\mathsf{OMR_N^*}$-satisfiable.

Let Γ^φ be the set of all subformulas of φ^*. Since φ^* is $\mathsf{OMR_N^*}$-satisfiable, there exist an $\mathsf{OMR_N^*}$-model $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$ and $a \in U$ such that $\mathcal{M}, a \models \varphi^*$. Given $s \in U$, let Γ^φ_s be the set of all subformulas ψ of φ^* which are satisfied in s, that is $\Gamma^\varphi_s \overset{\text{df}}{=} \{\psi \in \Gamma^\varphi : \mathcal{M}, s \models \psi\}$. Now, let us define a binary relation $\sim_{\Gamma^\varphi}$ on U as:

$$s \sim_{\Gamma^\varphi} t \text{ if and only if } \Gamma^\varphi_s = \Gamma^\varphi_t.$$

Fact 3. *The relation* $\sim_{\Gamma^\varphi}$ *is an equivalence relation on* U*.*

Now, we will define a filtration model $\mathcal{M}_{\Gamma^\varphi}$ determined by the relation $\sim_{\Gamma^\varphi}$ and the model $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$. A filtration model is a structure of the form $\mathcal{M}_{\Gamma^\varphi} = (U_{\Gamma^\varphi}, R_{\Gamma^\varphi}, \overline{R}_{\Gamma^\varphi}, N_{\Gamma^\varphi}, \overline{N}_{\Gamma^\varphi}, m_{\Gamma^\varphi})$, where:

– $U_{\Gamma^\varphi} = \{|s| : s \in U\}$, where $|s|$ is an equivalence class of s determined by the relation $\sim_{\Gamma^\varphi}$
– $m_{\Gamma^\varphi}(p) = \{|s| \in U_{\Gamma^\varphi} : s \in m(p)\}$
– $m_{\Gamma^\varphi}(c_i) = |m(c_i)|$
– R_{Γ^φ} is a binary relation on U_{Γ^φ} defined for all $|s|, |t| \in U_{\Gamma^\varphi}$ as:
  $(|s|, |t|) \in R_{\Gamma^\varphi}$ iff for every $\varphi \in \Gamma^\varphi$, if $[R]\varphi \in \Gamma^\varphi$ and $\mathcal{M}, s \models [R]\varphi$, then $\mathcal{M}, t \models [R]\varphi \wedge \varphi$
– $\overline{R}_{\Gamma^\varphi}$ is a binary relation on U_{Γ^φ} defined for all $|s|, |t| \in U_{\Gamma^\varphi}$ as:
  $(|s|, |t|) \in \overline{R}_{\Gamma^\varphi}$ iff for every $\varphi \in \Gamma^\varphi$, if $[R]\varphi \in \Gamma^\varphi$ and $\mathcal{M}, t \models [R]\varphi$, then $\mathcal{M}, s \models [R]\varphi \wedge \varphi$
– N_{Γ^φ} is a binary relation on U_{Γ^φ} defined as in $\mathsf{OMR_N}$-models, that is, $N_{\Gamma^\varphi} \overset{\text{df}}{=} G_1 \cup G_2$:
  $G_1 = \{(x, y) \in U_{\Gamma^\varphi} \times U_{\Gamma^\varphi} : x = m_{\Gamma^\varphi}(c_3)\}$
  $G_2 = \{(x, y) \in U_{\Gamma^\varphi} \times U_{\Gamma^\varphi} : (\lambda \text{ or } \mu) \text{ and } (\gamma \text{ or } \delta) \text{ and } (\zeta \text{ or } \eta)\}$, where:
    $\lambda := ((m_{\Gamma^\varphi}(c_2), x) \in R_{\Gamma^\varphi}$, $\mu := (x = m_{\Gamma^\varphi}(c_2))$,
    $\gamma := (((x, m_{\Gamma^\varphi}(c_4)) \in R_{\Gamma^\varphi})$, $\delta := (x = m_{\Gamma^\varphi}(c_4))$,
    $\zeta := ((y, m_{\Gamma^\varphi}(c_1)) \in R_{\Gamma^\varphi})$ and $\eta := ((m_{\Gamma^\varphi}(c_5), y) \in R_{\Gamma^\varphi})$
– $\overline{N}_{\Gamma^\varphi}$ is the converse of N_{Γ^φ}.

Proposition 4. *Let φ be an $\mathsf{OMR_N}$-formula satisfiable in an $\mathsf{OMR_N^*}$-model $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$. Then, the filtration model $\mathcal{M}_{\Gamma^\varphi}$ determined by the relation $\sim_{\Gamma^\varphi}$ and the model $\mathcal{M}$ is an $\mathsf{OMR_N^*}$-model.*

Proof. Let $\mathcal{M}_{\Gamma^\varphi} = (U_{\Gamma^\varphi}, R_{\Gamma^\varphi}, \overline{R}_{\Gamma^\varphi}, N_{\Gamma^\varphi}, \overline{N}_{\Gamma^\varphi}, m_{\Gamma^\varphi})$ be a filtration model determined by the relation $\sim_{\Gamma^\varphi}$ and the model $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$. First, observe that U_{Γ^φ} is non-empty, as U is so and φ is satisfiable in $\mathcal{M}$. Clearly, $\overline{R}_{\Gamma^\varphi}$ is the converse of R_{Γ^φ}. Now, observe that for all $s, t \in U$:

$$(*)\ \text{ If } (s,t) \in R, \text{ then } (|s|, |t|) \in R_{\Gamma^\varphi}.$$

Indeed, assume $(s,t) \in R$. Let $[R]\varphi \in \Gamma^\varphi$ and $\mathcal{M}, s \models [R]\varphi$. Then, by the assumption, $\mathcal{M}, t \models \varphi$. Now, let $w \in U$ be such that $(t, w) \in R$. Since R is transitive, $(s, w) \in R$. Thus, by the assumption, $\mathcal{M}, w \models \varphi$. So $\mathcal{M}, t \models [R]\varphi \wedge \varphi$, and hence $(|s|, |t|) \in R_{\Gamma^\varphi}$. Now, we prove that R_{Γ^φ} is transitive. Assume $(|s|, |t|) \in R_{\Gamma^\varphi}$ and $(|t|, |w|) \in R_{\Gamma^\varphi}$. Let $[R]\varphi \in \Gamma^\varphi$ be such that $\mathcal{M}, s \models [R]\varphi$. Suppose $\mathcal{M}, w \not\models [R]\varphi \wedge \varphi$. Then, since $(|t|, |w|) \in R_{\Gamma^\varphi}$, we obtain $\mathcal{M}, t \not\models [R]\varphi$. Therefore, $\mathcal{M}, t \not\models [R]\varphi \wedge \varphi$. Thus, since $(|s|, |t|) \in R_{\Gamma^\varphi}$, $\mathcal{M}, s \not\models [R]\varphi$, a contradiction. Hence, R_{Γ^φ} is transitive. By (*) and since R is connected, the proof of connectedness of R_{Γ^φ} is obvious.

Moreover, observe that $[R]\neg c_i \in \Gamma^\varphi$ and $\mathcal{M}, m(c_i) \models [R]\neg c_i$, but we have also $\mathcal{M}, m(c_i) \not\models \neg c_i$. Hence, $(|m(c_i)|, |m(c_i)|) \notin R_{\Gamma^\varphi}$. Since $(m(c_j), m(c_{j+1})) \in R$, by (*) we get $(|m(c_j)|, |m(c_{j+1})|) \in R_{\Gamma^\varphi}$. Furthermore, N_{Γ^φ} and $\overline{N}_{\Gamma^\varphi}$ are defined as in $\mathsf{OMR_N}$-models. Hence, $\mathcal{M}_{\Gamma^\varphi}$ is an $\mathsf{OMR_N^*}$-model. □

From the construction of the model $\mathcal{M}_{\Gamma^\varphi}$, we obtain the following:

Fact 4. *Let φ be a formula satisfiable in an $\mathsf{OMR_N^*}$-model $\mathcal{M}$. Then, the size of the universe of the filtration $\mathsf{OMR_N^*}$-model $\mathcal{M}_{\Gamma^\varphi}$, determined by the relation $\sim_{\Gamma^\varphi}$ and the model $\mathcal{M}$, is bounded by $2^{|\varphi^*|}$, where $|\varphi^*|$ is the size of the formula φ^*.*

Proposition 5. *Let $\mathcal{M}_{\Gamma^\varphi} = (U_{\Gamma^\varphi}, R_{\Gamma^\varphi}, \overline{R}_{\Gamma^\varphi}, N_{\Gamma^\varphi}, \overline{N}_{\Gamma^\varphi}, m_{\Gamma^\varphi})$ be a filtration model determined by the relation $\sim_{\Gamma^\varphi}$ and the model $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$. Then, for all $s, t \in U$ the following hold:*

1. *$(s,t) \in N$ iff $(|s|, |t|) \in N_{\Gamma^\varphi}$.*
2. *$(s,t) \in \overline{N}$ iff $(|s|, |t|) \in \overline{N}_{\Gamma^\varphi}$.*

Proof. First, we show that for every $s \in U$ and for every $i \in \{1, \ldots, 5\}$, the following hold:

(a) $s = m(c_i)$ iff $|s| = |m(c_i)|$.
(b) $(s, m(c_i)) \in R$ iff $(|s|, |m(c_i)|) \in R_{\Gamma^\varphi}$.
(c) $(m(c_i), s) \in R$ iff $(|m(c_i)|, |s|) \in R_{\Gamma^\varphi}$.

(a) Note that $c_i \in \Gamma^\varphi$. Clearly, for every $s \in U$, $\mathcal{M}, s \models c_i$ iff $s = m(c_i)$, thus by the definition of the relation $\sim_{\Gamma^\varphi}$, $s = m(c_i)$ if and only if $|s| = |m(c_i)|$.

(b) Assume $(|s|, |m(c_i)|) \in R_{\Gamma^\varphi}$, that is if $[R]\varphi \in \Gamma^\varphi$ and $\mathcal{M}, s \models [R]\varphi$, then $\mathcal{M}, t \models [R]\varphi \wedge \varphi$. Suppose $(s, m(c_i)) \notin R$. Since R is connected, either $s = m(c_i)$ or $(m(c_i), s) \in R$. If $s = m(c_i)$, then by (a) and the assumption, $(|m(c_i)|, |m(c_i)|) \in R_{\Gamma^\varphi}$. However, $\mathcal{M}_{\Gamma^\varphi}$ is an $\mathsf{OMR}^*_\mathsf{N}$-model, so it satisfies $(|m(c_i)|, |m(c_i)|) \notin R_{\Gamma^\varphi}$, a contradiction. Next, if $(m(c_i), s) \in R$, then by (*) (see the proof of Proposition 4), $(|m(c_i)|, |s|) \in R_{\Gamma^\varphi}$. Thus, by the assumption and transitivity of R_{Γ^φ}, $(|m(c_i)|, |m(c_i)|) \in R_{\Gamma^\varphi}$, a contradiction. The other direction follows from (*). The proof of (c) is similar.

By (a), (b), (c) above and the definition of N_{Γ^φ}, we get the proposition. □

Proposition 6. *Let $\mathcal{M}_{\Gamma^\varphi} = (U_{\Gamma^\varphi}, R_{\Gamma^\varphi}, \overline{R}_{\Gamma^\varphi}, N_{\Gamma^\varphi}, \overline{N}_{\Gamma^\varphi}, m_{\Gamma^\varphi})$ be a filtration model determined by the relation $\sim_{\Gamma^\varphi}$ and the model $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$. Then, for every $\psi \in \Gamma^\varphi$ and for every $s \in U$, the following conditions are equivalent:*

1. $\mathcal{M}, s \models \psi$.
2. $\mathcal{M}_{\Gamma^\varphi}, |s| \models \psi$.

Proof. The proof of the proposition is by the induction on the complexity of formulas in Γ^φ. For propositional variables it holds by the definition of the meaning function m_{Γ^φ}. For propositional constants the proposition follows from the property (a) (see the proof of Proposition 5). Assume the claim holds for $\vartheta, \chi \in \Gamma^\varphi$.

Let $\psi := \neg\theta$. Then, $\mathcal{M}, s \models \neg\theta$ iff $\mathcal{M}, s \not\models \theta$ iff, by the induction hypothesis, $\mathcal{M}_{\Gamma^\varphi}, |s| \not\models \theta$ iff $\mathcal{M}_{\Gamma^\varphi}, |s| \models \neg\theta$.

Let $\psi := \theta \wedge \chi$. Then, $\mathcal{M}, s \models \theta \wedge \chi$ iff $\mathcal{M}, s \models \theta$ and $\mathcal{M}, s \models \chi$iff, by the induction hypothesis, $\mathcal{M}_{\Gamma^\varphi}, |s| \models \theta$ and $\mathcal{M}_{\Gamma^\varphi}, |s| \models \chi$ iff $\mathcal{M}_{\Gamma^\varphi}, |s| \models \theta \wedge \chi$.

For formulas built with the other classical propositional operations the claim can be proved in a similar way.

Let $\psi := [R]\theta$ be such that $[R]\theta \in \Gamma^\varphi$. Assume $\mathcal{M}, s \models [R]\theta$. Let $t \in U$ be such that $(|s|, |t|) \in R_{\Gamma^\varphi}$. Then, by the assumption, $\mathcal{M}, t \models \theta$. Thus, by the induction hypothesis, $\mathcal{M}_{\Gamma^\varphi}, |t| \models \theta$. Hence, $\mathcal{M}_{\Gamma^\varphi}, |s| \models [R]\theta$. Now, assume that $\mathcal{M}_{\Gamma^\varphi}, |s| \models [R]\theta$ and let $t \in U$ be such that $(s, t) \in R$. Then, $(|s|, |t|) \in R_{\Gamma^\varphi}$ and by the assumption $\mathcal{M}_{\Gamma^\varphi}, |t| \models \theta$. Therefore, by the induction hypothesis, $\mathcal{M}, t \models \theta$. Hence, $\mathcal{M}, s \models [R]\theta$. For $\psi := [\overline{R}]\theta$ the proof is similar.

Let $\psi := [N]\theta$ be such that $[N]\theta \in \Gamma^\varphi$. Assume $\mathcal{M}, s \models [N]\theta$. Let $t \in U$ be such that $(|s|, |t|) \in N_{\Gamma^\varphi}$. Then, by Proposition 5, $(s, t) \in N$, so by the assumption $\mathcal{M}, t \models \theta$. Thus, by the induction hypothesis, $\mathcal{M}, |t| \models \theta$. Hence, $\mathcal{M}_{\Gamma^\varphi}, |s| \models [N]\theta$. Now, assume that $\mathcal{M}_{\Gamma^\varphi}, |s| \models [N]\theta$ and let $t \in U$ be such that $(s, t) \in N$. Then, by Proposition 5 and the assumption, $\mathcal{M}_{\Gamma^\varphi}, |t| \models \theta$. Therefore, by the induction hypothesis, $\mathcal{M}, t \models \theta$. Hence, $\mathcal{M}, s \models [N]\theta$. For $\psi := [\overline{N}]\theta$ the proof is similar. □

Proposition 7. *Let φ be an OMR-formula that is satisfied in an $\mathsf{OMR}^*_\mathsf{N}$-model $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$ by a state $a \in U$. Then, the formula φ is satisfied in the $\mathsf{OMR}^*_\mathsf{N}$-model $\mathcal{M}_{\Gamma^\varphi} = (U_{\Gamma^\varphi}, R_{\Gamma^\varphi}, \overline{R}_{\Gamma^\varphi}, N_{\Gamma^\varphi}, \overline{N}_{\Gamma^\varphi}, m_{\Gamma^\varphi})$ by $|a| \in U_{\Gamma^\varphi}$.*

Proof. The proof is straightforward. Let φ be a formula satisfied in an $\mathsf{OMR}^*_\mathsf{N}$-model $\mathcal{M} = (U, R, \overline{R}, N, \overline{N}, m)$ by a state $a \in U$. Take the filtration $\mathsf{OMR}^*_\mathsf{N}$-model $\mathcal{M}_{\Gamma^\varphi} = (U_{\Gamma^\varphi}, R_{\Gamma^\varphi}, \overline{R}_{\Gamma^\varphi}, N_{\Gamma^\varphi}, \overline{N}_{\Gamma^\varphi}, m_{\Gamma^\varphi})$. Clearly, $\varphi^* \in \Gamma^\varphi$ and $|a| \in U_{\Gamma^\varphi}$. By the assumption, we have $\mathcal{M}, a \models \varphi^*$. Thus, by Proposition 6, we obtain $\mathcal{M}_{\Gamma^\varphi}, |a| \models \varphi^*$. Hence, $\mathcal{M}_{\Gamma^\varphi}, |a| \models \varphi$. □

Recall that a logic L is said to have the *finite model property* whenever every L-satisfiable formula is satisfiable in a finite L-model. Moreover, a logic L is said to have the *strong finite model property* whenever there exists a computable function f such that every L-satisfiable formula of size n is satisfiable in a finite L-model of size at most $f(n)$.

From Proposition 7 and Fact 4 we get:

Theorem 5 (Strong Finite Model Property). *The logic $\mathsf{OMR_N}$ has the strong finite model property.*

Now, since the logic $\mathsf{OMR_N}$ has the strong finite model property and the finite $\mathsf{OMR}^*_\mathsf{N}$-models form a recursively enumerable set, searching through all the finite models is an effective procedure for generating all non-provable formulas of $\mathsf{OMR_N}$. Thus, the set of all $\mathsf{OMR_N}$-provable formulas is recursive, which implies decidability of $\mathsf{OMR_N}$.

Theorem 6 (Decidability). *The logic $\mathsf{OMR_N}$ is decidable.*

4 Conclusions

We have showed that the logic $\mathsf{OMR_N}$ for order-of-magnitude qualitative reasoning based on Absolute Order of Magnitude model $\mathsf{AOM}(5)$ with bidirectional negligibility relation has the strong finite model property, and thus it is decidable. So now the natural task is to construct its decision procedure based on (dual) tableau system with good space complexity. Furthermore, there are other logics than $\mathsf{OMR_N}$ that enable us to deal with the order-of-magnitude reasoning. In particular, the logics based on order-of-magnitude model with various relations, among which are relations of comparability, negligibility, (non-)closeness, comparability, and distance, have been introduced (see [2], [4]). However, the problem whether these logics are decidable is still open. So as a future work it is planned to tackle the problem of decidability for these logics, and in the case of positive answer to construct their decision procedures based on (dual) tableau systems, further implemented in PROLOG.

Acknowledgments. This work was supported by the Polish National Science Centre research project DEC-2011/02/A/HS1/00395.

References

1. Burrieza, A., Ojeda-Aciego, M.: A Multimodal Logic Approach to Order of Magnitude Qualitative Reasoning. In: Conejo, R., Urretavizcaya, M., Pérez-de-la-Cruz, J.-L. (eds.) CAEPIA/TTIA 2003. LNCS (LNAI), vol. 3040, pp. 431–440. Springer, Heidelberg (2004)
2. Burrieza, A., Ojeda-Aciego, M.: A multimodal logic approach to order of magnitude qualitative reasoning with comparability and negligibility relations. Fundamenta Informaticae 68(1-2), 21–46 (2005)
3. Burrieza, A., Muñoz, E., Ojeda-Aciego, M.: Order of Magnitude Qualitative Reasoning with Bidirectional Negligibility. In: Marín, R., Onaindía, E., Bugarín, A., Santos, J. (eds.) CAEPIA 2005. LNCS (LNAI), vol. 4177, pp. 370–378. Springer, Heidelberg (2006)
4. Burrieza, A., Muñoz-Velasco, E., Ojeda-Aciego, M.: A Logic for Order of Magnitude Reasoning with Negligibility, Non-closeness and Distance. In: Borrajo, D., Castillo, L., Corchado, J.M. (eds.) CAEPIA 2007. LNCS (LNAI), vol. 4788, pp. 210–219. Springer, Heidelberg (2007)
5. Golińska-Pilarek, J., Mora, A., Muñoz-Velasco, E.: An ATP of a Relational Proof System for Order of Magnitude Reasoning with Negligibility, Non-closeness and Distance. In: Ho, T.-B., Zhou, Z.-H. (eds.) PRICAI 2008. LNCS (LNAI), vol. 5351, pp. 128–139. Springer, Heidelberg (2008)
6. Golińska-Pilarek, J., Muñoz-Velasco, E.: Relational approach for a logic for order-of-magnitude qualitative reasoning with negligibility, non-closeness and distance. Logic Journal of IGPL 17(4), 375–394 (2009)
7. Golińska-Pilarek, J., Muñoz-Velasco, E.: Dual tableau for a multimodal logic for order-of-magnitude qualitative reasoning with bidirectional negligibility. International Journal of Computer Mathematics 86(10-11), 1707–1718 (2009)
8. Orłowska, E., Golińska-Pilarek, J.: Dual Tableaux: Foundations, Methodology, Case Studies. Trends in Logic 36. Springer (2011)
9. Raiman, O.: Order of magnitude reasoning. Artificial Intelligence 51(1-3), 11–38 (1991)

Fault Tolerance in Belief Formation Networks

Sarah Holbrook and Pavel Naumov

Department of Mathematics and Computer Science
McDaniel College, Westminster, Maryland 21157, USA
{seh002,pnaumov}@mcdaniel.edu

Abstract. The paper investigates the formation of beliefs in multi-agent systems with a fixed topology of the communication channels. Specifically, it considers the relation "beliefs formed by agents in set A are not influenced by faulty or malicious behavior of agents in set B". This relation has a non-trivial Shield Wall property that has no equivalent in other settings in which information flow over a fixed network of communication channels has been previously studied.

A new logical system based on the Shield Wall property is proposed and is proven to be sound and complete with respect to the fault tolerance semantics.

1 Introduction

Belief Networks. In this paper we study dependencies between beliefs in multi-agent systems. An example of a set of such dependencies in a four-agent system is given in Figure 1. In this figure, the accused (*Eva*), two witnesses (*Alice* and *Bob*), *Jury*, and *Public* are agents and *EvaMurdered* and *EvaIsGuilty* are two beliefs. The two Horn clauses given in the figure are *belief formation rules.* These rules express how different agents form new beliefs based on existing beliefs of the other agents. For example, according to the first cause, if both witnesses believe that Eva is a murderer, then the jury will form a belief that Eva is guilty. According to the second rule, once the jury decides that Eva is guilty, this belief could propagate to the public. We will refer to settings such as the one in Figure 1 as *belief formation networks* or just belief networks. The process of belief propagation on such networks will be called *belief formation*, to emphasize the fact that the propagated belief ("Eva is guilty") could be different from the originating belief ("Eva is a murderer") and to differentiate this non-probabilistic approach from belief propagation on Bayesian networks [1,2].

$$\mathsf{Alice{:}EvaMurdered} \wedge \mathsf{Bob{:}EvaMurdered} \rightarrow \mathsf{Jury{:}EvaIsGuilty}$$
$$\mathsf{Jury{:}EvaIsGuilty} \rightarrow \mathsf{Public{:}EvaIsGuilty}$$

Fig. 1. Belief Network $\mathcal{N}_1$

In the example above, the public forms its beliefs based only on the jury's beliefs, not directly on witnesses testimonies. The jury, on the other hand, bases its beliefs on the witnesses' testimonies, not on public opinion. These properties of the belief network are captured by belief dependency graph G_1 depicted in Figure 2.

Fault Tolerance. There are many different properties of belief networks and belief dependency graphs that one can consider. The focus of this paper is on *fault tolerance* – the ability of an agent in a belief network to resist forming false beliefs when some other agents exhibit faulty or malicious behavior. For example, assume, for the above discussed network $\mathcal{N}_1$, that neither Alice nor Bob have witnessed the murder. Then no beliefs will be formed in this network. If Bob misbehaves and lies to the jury, then, assuming Alice does not lie, neither the jury nor the public will form a false belief. We denote this by $Jury \parallel Bob$ and $Public \parallel Bob$. At the same time, if both Alice and Bob, misbehave, then this might result in the jury and the public forming a false belief: $\neg(Jury \parallel Alice, Bob)$ and $\neg(Public \parallel Alice, Bob)$.

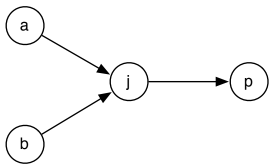


Fig. 2. Graph G_1 shows belief dependencies between agents *Alice* (a), *Bob* (b), *Jury* (j), and *Public* (p)

Alice:EvaMurdered $\rightarrow$ Jury:EvaIsGuilty
Bob:EvaMurdered $\rightarrow$ Jury:EvaIsGuilty
Jury:EvaIsGuilty $\rightarrow$ Public:EvaIsGuilty

Fig. 3. Belief Network $\mathcal{N}_2$

Different belief networks can have the same dependency graph. A fault tolerance claim which is true in one of these networks might be false in the other. For example, statements $Jury \parallel Bob$ and $Public \parallel Bob$ are not true in the belief network $\mathcal{N}_2$ depicted in Figure 3, although it has the same dependency graph G_1 as network $\mathcal{N}_1$.

1.1 Gateway

In this paper we study properties of fault tolerance that are common for all belief networks sharing the same dependency graph. An example of such a property for graph G_1 is $Public \parallel Jury \rightarrow Public \parallel Alice, Bob$. This statement is a special case of a more general *Gateway* principle. Let A, B, and W be three sets of vertices of a graph G. We say that set W is a gateway (see Figure 4) to set A from set B if every path connecting a vertex from set B with a vertex from A contains at least one vertex from set W. (For directed graphs, every *directed* path from set B to set A must contain a vertex from set W.)

Gateway Principle. *If a set W is a gateway to a set A from a set B of a graph G, then the property $A \parallel W \rightarrow A \parallel B$ is true for any belief network with belief dependencies specified by the graph G.*

In the example above, $A = \{Public\}$, $W = \{Jury\}$, and $B = \{Alice, Bob\}$.

The gateway principle is an intuitively expected property of fault tolerance. Similar principles in other information flow settings were proposed earlier for functional dependency relation on hypergraphs [3] as well as independence relation on graphs [4], directed acyclic graphs [5], and hypergraphs [6]. (Independence is also known in literature as nondeducibility [7].) In all these settings an appropriate version of the gateway principle was not only sound, but, together with several other relevant properties, gave a complete axiomatization of each of these relations over a fixed graph.

It turns out, however, that, unlike the above settings, the gateway principle is far from giving a complete description of all fault tolerance properties of an arbitrary graph. For example, the graph discussed above, G_1, has the following property: $Jury \parallel Alice, Bob \to Public \parallel Alice, Bob$. This property is a special case of a more general principle:

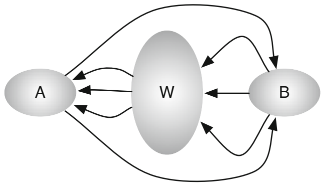


Fig. 4. Gateway

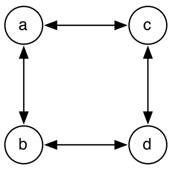


Fig. 5. Graph G_2

Second Gateway Principle. *If a set W is a gateway to a set A from a set B of a graph G, then property $W \parallel B \to A \parallel B$ is true for any belief network with belief dependencies specified by the graph G.*

We will formally prove soundness of both gateway principles later. These two principles, though, still do not form a complete logical system for reasoning about fault tolerance on directed graphs. Indeed, consider dependency graph G_2 depicted in Figure 5.

Proposition 1. *$a \parallel c \to (b \parallel d \to a, b \parallel c, d)$, for any belief network with belief dependencies specified by graph G_2.*

Proof Idea. By "false" belief, we mean a belief that cannot be formed if all agents are behaving properly. Assume that a certain malfunctioning of agents c and d results in at least one of agents a and b forming a false belief. Let t_0 be the moment when such false belief is formed for the first time. Without loss of generality, assume that the belief is formed by the agent a. Note that it means that at any moment $t < t_0$ agent b did not hold any false beliefs. Thus, the only reason for agent a to form a false belief at moment t_0 is a malfunctioning of the agent c. Therefore, $\neg(a \parallel c)$, which is a contradiction with the assumption. This argument will be formalized later in the proof of Theorem 3.

It is easy to see that statement $a \parallel c \to (b \parallel d \to a, b \parallel c, d)$ cannot be shown using either of the two gateway principles above. It is an example of a more general principle that we call the Shield Wall principle.

1.2 Shield Wall

Ancient warriors (and modern police forces) used multiple shields to protect themselves. Each shield does not provide complete protection for its owner, but if a group of warriors arranges its members in a circle with all shields facing outward, then the whole group is completely protected. This idea is formally captured in our Shield Wall principle:

Shield Wall Principle. *Let G be a graph. If B is a set of vertices, $\{A_i\}_{i \le n}$ are pair-wise disjoint sets of vertices, and $\{S_i\}_{i \le n}$ are sets of vertices such that*

1. sets A_i and S_i are disjoint for each $i \leq n$,
2. $S_i \cup (\bigcup_{j \neq i} A_j)$ is a gateway to A_i from B for each i,

then the property $\bigwedge_{i \leq n}(A_i \parallel S_i) \to \left(\bigcup_{i \leq n} A_i\right) \parallel B$ is true for any belief network with belief dependencies specified by the graph G.

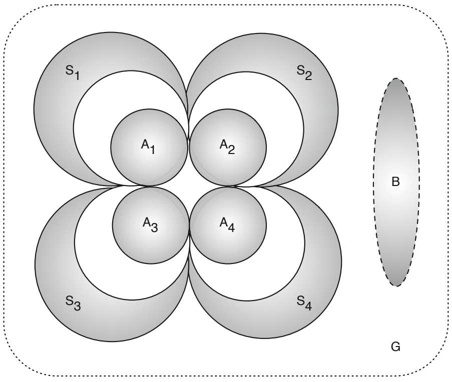


Fig. 6. Shield Wall

Informally, see Figure 6, sets $A_1, \dots, A_n$ are "warriors" that are "protected" by "shields" $S_1, \dots, S_n$. The fact that shield S_i protects warrior A_i is captured by the assumption $A_i \parallel S_i$. The fact that each warrior is blocked from the "enemy" B is expressed by the assumption that $S_i \cup (\bigcup_{j \neq i} A_j)$ is a gateway to set A_i from set B. The conclusion of the Shield Wall principle can be interpreted as a statement that the whole group $\bigcup_i A_i$ is protected against the enemy B.

For the graph on Figure 5, $n = 2$, $A_1 = \{a\}$, $A_2 = \{b\}$, $S_1 = \{c\}$, $S_2 = \{d\}$, and $B = \{c, d\}$. Thus, by the Shield Wall principle, $a \parallel c \to (b \parallel d \to a, b \parallel c, d)$. The Shield Wall principle also manifests itself in international visa and custom treaties, such as the European Schengen Agreement, under which each member ("warrior") enforces passport and custom control along external borders with non-members ("shields"). This results in all member-states being protected from undesired visitors and goods.

1.3 Logics of Fault Tolerance

Logical aspects of fault tolerance have been analyzed before. This work lead to the origination of a LICS workshop series on Logical Aspects of Fault Tolerance [8,9]. Logical frameworks that have been used for such analysis are Temporal Logic (see, for example, [10]) and Deontic Logic [11]. Unlike the approach proposed in this paper, these frameworks do not integrate topological structure of the multi-agent system.

2 Graph Terminology

In this paper, all graphs are assumed to be directed. A directed path from vertex v_1 to vertex v_n in a graph $G = (V, E)$ is a sequence of vertices $v_1, \dots, v_n$ such that $(v_i, v_{i+1}) \in E$ for each $i < n$. For example, a, j, p is a directed path in graph G_1, depicted in Figure 2.

Definition 1. *A set $W \subseteq V$ is a gateway to set $A \subseteq V$ from set $B \subseteq V$ of a graph $G = (V, E)$ if each path to a vertex in set A from a vertex in set B contains a vertex from set W.*

3 Semantics

In this section we give a formal definition of fault tolerance in belief networks that was discussed informally in the introduction.

Definition 2. *For any directed graph $G = (V, E)$, a belief formation network over G is a triple $\mathcal{N} = (\Omega, s, R)$, where*

1. *Ω is an arbitrary finite set (of "beliefs"),*
2. *the "initial state" s is an arbitrary mapping of vertices into sets of beliefs: $V \mapsto 2^{\Omega}$,*
3. *R is an arbitrary finite set of belief propagation rules*

$$(v_1 : \alpha_1) \wedge (v_2 : \alpha_2) \wedge \cdots \wedge (v_n : \alpha_n) \to w : \beta,$$

such that $v_1, \ldots, v_n, w \in V$, $\alpha_1, \ldots, \alpha_n, \beta \in \Omega$, and $(v_i, w) \in E$ for each $i \leq n$.

Definition 3. *For any belief network $\mathcal{N} = (\Omega, s, R)$ over $G = (V, E)$, state t of the network $\mathcal{N}$ is an arbitrary function from V to subsets of Ω.*

If a belief network $\mathcal{N}$ can go from state x to state y through a single application of a belief formation rule, then we write $x \rightharpoonup_{\mathcal{N}} y$. The formal definition of this relation is given below.

Definition 4. *Let x and y be two states of a belief network $\mathcal{N} = (\Omega, s, R)$ over $G = (V, E)$. We write $x \rightharpoonup_{\mathcal{N}} y$ if there is $v_0 \in V$ such that*

1. *$x(v) = y(v)$ for each vertex $v \neq v_0$,*
2. *there is a rule*

$$(v_1 : \alpha_1) \wedge (v_2 : \alpha_2) \wedge \cdots \wedge (v_n : \alpha_n) \to v_0 : \beta, \tag{1}$$

in set R such that (a) $\alpha_i \in x(v_i)$ for each $i \leq n$, (b) $y(v_0) = x(v_0) \cup \{\beta\}$.

Vertex v_0 will be called the "active" vertex of the rule (1). Let relation $x \rightharpoonup^*_{\mathcal{N}} y$ be the transitive and reflexive closure of the relation $x \rightharpoonup_{\mathcal{N}} y$.

Lemma 1. *Let $\mathcal{N} = (\Omega, s, R)$ be an arbitrary belief network over $G = (V, E)$. For any states t_1, t_2 such that $s \rightharpoonup^*_{\mathcal{N}} t_1$ and $s \rightharpoonup^*_{\mathcal{N}} t_2$, there is a state t such that $s \rightharpoonup^*_{\mathcal{N}} t$ and $t(v) = t_1(v) \cup t_2(v)$ for each $v \in V$.*

Proof. Since $s \rightharpoonup^*_{\mathcal{N}} t_1$, there is a sequence of belief formation rules, that, when applied, transform belief network $\mathcal{N}$ from state s into state t_1. By assumption $s \rightharpoonup^*_{\mathcal{N}} t_2$, there is a similar sequence for state t_2. Combine these sequences in, say, consecutive order. Let t be the resulting state. □

By $\mathit{Mindset}_{\mathcal{N}}(v)$ we mean all beliefs that can be potentially formed at vertex v.

Definition 5. *For any belief network $\mathcal{N} = (\Omega, s, R)$ over $G = (V, E)$ and any $v \in V$, $Mindset_{\mathcal{N}}(v) = \bigcup_{s \twoheadrightarrow^*_{\mathcal{N}} t} t(v)$.*

Note that $Mindest_{\mathcal{N}}$, being a function from vertices into subsets of Ω, can also be viewed as a state of the belief network.

Lemma 2. *If $\mathcal{N} = (\Omega, s, R)$ is an arbitrary belief network over $G = (V, E)$, then $s \twoheadrightarrow^*_{\mathcal{N}} Mindset_{\mathcal{N}}$.*

Proof. The statement follows from finiteness of the set Ω and Lemma 1. $\square$

For any belief network $\mathcal{N}$ and any set of vertices B, by $\mathcal{N}_B$ we mean a modified network in which vertices in set B can misbehave or malfunction. We capture this formally in the definition below by adding new belief formation rules to $\mathcal{N}_B$ that allow vertices in set B to form any belief from set Ω without preconditions.

Definition 6. *For any belief network $\mathcal{N} = (\Omega, s, R)$ over a graph $G = (V, E)$ and any $B \subseteq V$, we define belief network $\mathcal{N}_B$ to be (Ω, s, R_B), where*

$$R_B = R \cup \{b : \alpha \mid b \in B, \alpha \in \Omega\}.$$

Definition 7. *For any graph $G = (V, E)$, by $\Phi(G)$ we mean the minimal set of formulas such that*

1. *$\perp \in \Phi(G)$,*
2. *$A \parallel B \in \Phi(G)$ for each $A, B \subseteq V$,*
3. *$\phi \to \psi \in \Phi(G)$ for each $\phi, \psi \in \Phi(G)$.*

The next definition is the key definition in this paper. It formally defines the fault tolerance relation $A \parallel B$ in belief networks as the inability of vertices in set A to form any additional beliefs if vertices in set B are malfunctioning.

Definition 8. *For any belief network $\mathcal{N} = (\Omega, s, R)$ over graph $G = (V, E)$ and any formula $\phi \in \Phi(G)$, we define truth relation $\mathcal{N} \vDash \phi$ by recursion on structural complexity of the formula ϕ:*

1. *$\mathcal{N} \nVdash \perp$*
2. *$\mathcal{N} \Vdash A \parallel B$ if and only if, for each $v \in A$, $Mindset_{\mathcal{N}_B}(v) \subseteq Mindset_{\mathcal{N}}(v)$*
3. *$\mathcal{N} \Vdash \phi_1 \to \phi_2$ if $\mathcal{N} \nVdash \phi_1$ or $\mathcal{N} \Vdash \phi_2$.*

4 Axioms

Our formal logical system for a graph G, in addition to Modus Ponens inference rules and propositional tautologies in the language $\Phi(G)$, contains the following three axioms:

1. **Empty Set:** $A \parallel \varnothing$,
2. **Monotonicity:** $A, B \parallel C \to A \parallel C$,

3. **Shield Wall:** $\bigwedge_{0<i\le n}(A_i \parallel S_i) \to (\bigcup_{0<i\le n} A_i) \parallel B$, where $\{A_i\}_i$ are pairwise disjoint sets and $S_i \cup (\bigcup_{j\ne i} A_j)$ is a gateway to A_i from B for each $0 < i \le n$.

We write $\vdash_G \phi$ if $\phi \in \Phi(G)$ is provable from the axioms above and propositional tautologies in the language $\Phi(G)$ using the Modus Ponens inference rule. We write $X \vdash_G \phi$ if ϕ is provable using the additional set of axioms X. We often omit the parameter G when its value is clear from the context.

5 Examples

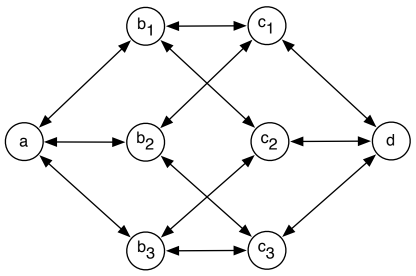
In this section, we give several examples of proofs in our formal system. We start by proving the two gateway principles mentioned in the introduction.

Proposition 2 (first gateway). *If W is a gateway (see Figure 4) to set of vertices A from set of vertices B of a graph G, then $\vdash_G A \parallel W \to A \parallel B$.*

Proof. Let $n = 1$, $A_1 = A$, $S_1 = W$, and $B = B$ in the Shield Wall axiom. □

Proposition 3 (second gateway). *If W is a gateway (see Figure 4) to set of vertices A from set of vertices B of a graph G, then $\vdash_G W \parallel B \to A \parallel B$.*

Proof. Let $n = 2$, $A_1 = A \setminus W$, $A_2 = W$, $S_1 = \varnothing$, $S_2 = B$ and $B = B$. By the Shield Wall axiom, $\vdash_G (A \setminus W) \parallel \varnothing \wedge W \parallel B \to A, W \parallel B$. By the Monotonicity axiom, $\vdash_G (A \setminus W) \parallel \varnothing \wedge W \parallel B \to A \parallel B$. Again by the Monotonicity axiom, $\vdash_G A \parallel \varnothing \wedge W \parallel B \to A \parallel B$. By the Empty Set axiom, $\vdash_G W \parallel B \to A \parallel B$. □



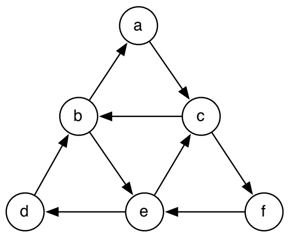


Fig. 7. Graph G_6 (left) and G_7 (right)

Proposition 4. *For G_6 depicted in Figure 7,*

$$\vdash_{G_6} (b_1 \parallel c_1, c_2) \wedge (b_2 \parallel c_1, c_3) \wedge (b_3 \parallel c_2, c_3) \to a \parallel d$$

Proof. Let $n = 4$, $A_1 = \{b_1\}$, $A_2 = \{b_2\}$, $A_3 = \{b_3\}$, $A_4 = \{a\}$, $S_1 = \{c_1, c_2\}$, $S_2 = \{c_1, c_3\}$, $S_3 = \{c_2, c_3\}$, $S_4 = \varnothing$, and $B = \{d\}$ in the Shield Wall axiom. Then, $\vdash_{G_6} (b_1 \parallel c_1, c_2) \wedge (b_2 \parallel c_1, c_3) \wedge (b_3 \parallel c_2, c_3) \wedge (a \parallel \varnothing) \to a, b_1, b_2, b_3 \parallel d$. By the Empty Set axiom,

$$\vdash_{G_6} (b_1 \parallel c_1, c_2) \wedge (b_2 \parallel c_1, c_3) \wedge (b_3 \parallel c_2, c_3) \to a, b_1, b_2, b_3 \parallel d.$$

By the Monotonicity axiom, $\vdash_{G_6} (b_1 \parallel c_1, c_2) \wedge (b_2 \parallel c_1, c_3) \wedge (b_3 \parallel c_2, c_3) \to a \parallel d$. □

Proposition 5. *For G_7 depicted in Figure 7, $\vdash_{G_7} (b \parallel c) \wedge (e \parallel f) \to d \parallel a$.*

Proof. Let $A_1 = \{d\}$, $A_2 = \{b\}$, $A_3 = \{e\}$, $S_1 = \varnothing$, $S_2 = \{c\}$, $S_3 = \{f\}$, and $B = \{a\}$. Thus, by the Shield Wall axiom, $\vdash_{G_7} (d \parallel \varnothing) \wedge (b \parallel c) \wedge (e \parallel f) \to b, d, e \parallel a$. By the Empty Set axiom, $\vdash_{G_7} (b \parallel c) \wedge (e \parallel f) \to b, d, e \parallel a$. By the Monotonicity axiom, $\vdash_{G_7} (b \parallel c) \wedge (e \parallel f) \to d \parallel a$. □

Proposition 6. *If G_8 is the graph depicted in Figure 8, then*

$$\vdash_{G_8} (x \parallel a, d) \wedge (y \parallel c, f) \wedge (z \parallel e, b) \to x, y, z \parallel a, c, e.$$

Proof. Let $n = 3$, $A_1 = \{x\}$, $A_2 = \{y\}$, $A_3 = \{z\}$, $S_1 = \{a, d\}$, $S_2 = \{c, f\}$, $S_3 = \{e, b\}$, and $B = \{a, c, e\}$. in the Shield Wall axiom. □

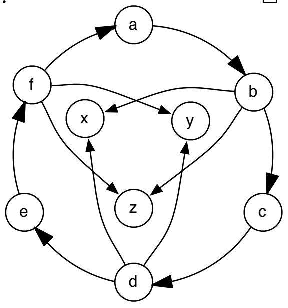


Fig. 8. Graph G_8

The next three examples are statements that are true for any graph G. Nevertheless, their proofs use the Shield Wall axiom.

Proposition 7. *$\vdash \varnothing \parallel A$, for any set of vertices A of any graph G.*

Proof. Consider the Shield Wall axiom for $n = 0$. □

Proposition 8. *$\vdash A \parallel B, C \to A \parallel B$, for any sets of vertices A, B, and C of any graph G.*

Proof. Any path to a vertex in set A from a vertex in set B trivially contains a vertex from $B \cup C$. Hence, $B \cup C$ is a shield that separates set A from set B. The result, thus, follows from the Shield Wall axiom. □

Proposition 9. *$\vdash (A \parallel B) \wedge (C \parallel B) \to A, C \parallel B$, for any sets of vertices A, B, and C of any graph G.*

Proof. Let $A_1 = A$, $A_2 = C$, $S_1 = B$, and $S_2 = B$ in the Shield Wall axiom. □

6 Reverse Shield Wall

In the Shield Wall principle (see Figure 6), we assume that each of A_i can tolerate faulty behavior of the appropriate S_i and conclude that all A_i together can tolerate faulty behavior of B. What if the situation were reversed: each of S_i can tolerate faulty behavior of the appropriate A_i. Can we conclude in this case that B will tolerate the combined faulty behavior of all A_i? Intuitively, this seems to be true since "false" beliefs generated by the union of all A_i will be "locked" inside the wall formed by the shields S_i.

In other words, sets $A_1, \ldots, A_n$ can be thought of as groups of members of a secret society. Each group A_i is prohibited from revealing the society secrets to their non-member acquaintances that form the group S_i, but are free to discuss them with the other society members. As a result, the society secrets are not divulged to the outsiders in the group B.

Conjecture 1. $\bigwedge_{0<i\le n}(S_i \parallel A_i) \to B \parallel \bigcup_{0<i\le n} A_i$, where $\{A_i\}_i$ are pair-wise disjoint sets and $S_i \cup (\bigcup_{j\ne i} A_j)$ is a gateway **from** set A_i **to** set B for each $0 < i \le n$.

It turns out, however, that this conjecture, as stated, is not true. Indeed, consider belief network $\mathcal{N}_1$ over graph G_1 depicted, respectively, in Figure 1 and Figure 2. Let $A_1 = \{Alice\}$, $A_2 = \{Bob\}$, $S_1 = S_2 = \{Jury\}$, $B = \{Public\}$. If Conjecture 1 is true, then the following implication is true for the belief network $\mathcal{N}_1$: $(Jury \parallel Alice) \wedge (Jury \parallel Bob) \to Public \parallel Alice, Bob$, which, as we have discussed in the introduction, is not true.

In spite of the example above, our "secret society" intuition is correct in the sense that assumptions on the topology of the graph in Conjecture 1 could be adjusted to make the statement true and provable in our axiomatic system:

Proposition 10 (reverse shield wall). $\vdash_G \bigwedge_{0<i\le n}(S_i \parallel A_i) \to B \parallel \bigcup_{0<i\le n} A_i$, *where* B,$S_1,\dots,S_n$ *are pair-wise disjoint sets and*

1. $\bigcup_i S_i$ *is a gateway to* B *from* $\bigcup_i A_i$,
2. $B \cup A_i \cup \bigcup_{j\ne i} S_j$ *is a gateway to* S_i *from* $\bigcup_j A_j$, *for each* $0 < i \le n$.

Proof. From the Shield Wall axiom when the settings are such that $B, S_1, \dots, S_n$ are the "warriors" (formerly As), $\varnothing, A_1, \dots, A_n$ are the matching "shields" (formerly Ss) and $\bigcup_i A_i$ is the "enemy" (formerly B):

$$\vdash_G (B \parallel \varnothing) \wedge \bigwedge_{0<i\le n}(S_i \parallel A_i) \to B, \bigcup_{0<i\le n} S_i \parallel \bigcup_{0<i\le n} A_i.$$

By the Monotonicity axiom, $\vdash_G (B \parallel \varnothing) \wedge \bigwedge_{0<i\le n}(S_i \parallel A_i) \to B \parallel \bigcup_{0<i\le n} A_i$. By the Empty Set axiom, $\vdash_G \bigwedge_{0<i\le n}(S_i \parallel A_i) \to B \parallel \bigcup_{0<i\le n} A_i$. □

7 Soundness

We prove soundness of our logical system by justifying separately each of its axioms.

Theorem 1 (Empty Set). $\mathcal{N} \vDash A \parallel \varnothing$, *for any belief network* $\mathcal{N} = (\Omega, s, R)$ *over a graph* $G = (E, V)$ *and any subset* $B \subseteq E$.

Proof. Note that, by Definition 6, $\mathcal{N}_\varnothing = \mathcal{N}$. Thus, $Mindset_{\mathcal{N}_\varnothing}(v) \subseteq Mindset_{\mathcal{N}}(v)$ for each $v \in A$. □

Theorem 2 (Monotonicity). *If* $\mathcal{N} \vDash A, B \parallel C$, *then* $\mathcal{N} \vDash A \parallel C$, *for any belief network* $\mathcal{N} = (\Omega, s, R)$ *over a graph* $G = (E, V)$ *and any subsets* $A, B, C \subseteq E$.

Proof. If $Mindset_{\mathcal{N}_B}(v) \subseteq Mindset_{\mathcal{N}}(v)$ for each $v \in A \cup C$, then it follows that $Mindset_{\mathcal{N}_B}(v) \subseteq Mindset_{\mathcal{N}}(v)$ for each $v \in A$. □

Before proving soundness of the Shield Wall axiom, we establish the following technical lemma.

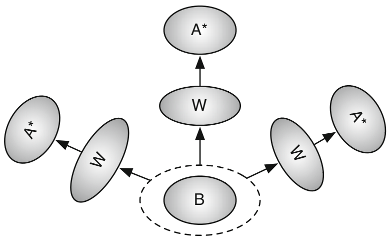


Fig. 9. Graph G_3

Lemma 3 (Gateway Lemma). *Let set W be a gateway to set A from set B in graph G and let $\mathcal{N} = (\Omega, s, R)$ be an arbitrary belief network over G. For any state t such that $s \rightharpoonup^*_{\mathcal{N}_B} t$ and any state s' such that $t(v) \subseteq s'(v)$ for any $v \in W$, there is a state t' such that $s' \rightharpoonup^*_{\mathcal{N}} t'$ and $t(v) \subseteq t'(v)$ for each $v \in A$.*

Proof. Let A^* be the set of all vertices $v \in V$ such that W is a gateway to set $\{v\}$ from set B (see Figure 9). By the assumption of the lemma, $A \subseteq A^*$.

Additionally, by the assumption $s \rightharpoonup^*_{\mathcal{N}_B} t$, there are states $s_1, \dots, s_n$ such that $s = s_1$, $s_n = t$, and

$$s_1 \rightharpoonup_{\mathcal{N}_B} s_2 \rightharpoonup_{\mathcal{N}_B} \cdots \rightharpoonup_{\mathcal{N}_B} s_{n-1} \rightharpoonup_{\mathcal{N}_B} s_n \tag{2}$$

Starting from state s', reproduce all belief formation steps in path (2) in which the active vertex belongs to A^*. The result will be a path

$$s'_1 \rightharpoonup_{\mathcal{N}} s'_2 \rightharpoonup_{\mathcal{N}} \cdots \rightharpoonup_{\mathcal{N}} s'_{k-1} \rightharpoonup_{\mathcal{N}} s'_k,$$

where $s'(v) = s'_1(v)$ and $s'_k(v) = t(v)$ for each $v \in A^*$. Take s'_k to be t'. □

Theorem 3 (Shield Wall). *Let $\mathcal{N} = (\Omega, s, R)$ be a belief network over a graph $G = (E, V)$, and $\{A_i\}_{i \le n}$, $\{S_i\}_{i \le n}$, and B be a sets of vertices such that*

1. *$\{A_i\}_{i \le n}$ are pair-wise disjoint,*
2. *A_i is disjoint with S_i for each $i \le n$,*
3. *B is disjoint with $\bigcup_i A_i$,*
4. *$S_i \cup (\bigcup_{j \ne i} A_j)$ is a gateway to A_i from B for each $i \le n$,*
5. *$\mathcal{N} \vDash A_i \parallel S_i$ for each $i \le n$.*

Then $\mathcal{N} \vDash \bigcup_i A_i \parallel B$.

Proof. We need to prove that $Mindset_{\mathcal{N}_B}(v) \subseteq Mindset_{\mathcal{N}}(v)$ for each $v \in \bigcup_i A_i$. Assume the opposite. Since beliefs are formed one at a time, there must be state t and $v_0 \in A_{i_0}$ such that

1. $s \rightharpoonup_{\mathcal{N}_B} t$,
2. $t(v_0) \nsubseteq Mindset_{\mathcal{N}}(v_0)$,
3. $t(v) \subseteq Mindset_{\mathcal{N}}(v)$ for each $v \in \bigcup_i A_i \setminus \{v_0\}$.

By Lemma 2, $s \twoheadrightarrow^*_{\mathcal{N}} Mindset_{\mathcal{N}}$. Thus, $s \twoheadrightarrow_{\mathcal{N}_{S_{i_0}}} s'$, where

$$s'(v) = \begin{cases} t(v) \cup Mindset_{\mathcal{N}}(v) & \text{if } v \in S_{i_0} \setminus \bigcup_i A_i, \\ Mindset_{\mathcal{N}}(v) & \text{otherwise.} \end{cases}$$

Note that $t(v) \subseteq Mindset_{\mathcal{N}}(v) = s'(v)$ for each $v \in \bigcup_{i \neq i_0} A_i$. Hence, $t(v) \subseteq s'(v)$ for each $v \in S_{i_0} \cup (\bigcup_{i \neq i_0} A_i)$, which, by an assumption of the theorem, is a gateway to A_{i_0} from B.

By Lemma 3, there is a state t' such that $s' \twoheadrightarrow^*_{\mathcal{N}} t'$ and $t(v) \subseteq t'(v)$ for each $v \in A_{i_0}$. Then, $s \twoheadrightarrow^*_{\mathcal{N}_{S_{i_0}}} t'$. At the same time, $t'(v_0) \nsubseteq Mindset_{\mathcal{N}}(v_0)$, because $t(v) \subseteq t'(v)$ and $t(v_0) \nsubseteq Mindset_{\mathcal{N}}(v_0)$. Since v_0 has been chosen from the set A_{i_0}, the above is a contradiction with the assumption $\mathcal{N} \vDash A_{i_0} \parallel S_{i_0}$. □

8 Completeness

Let $G = (V, E)$ be a directed graph, X be a maximal consistent subset of $\Phi(G)$, and $\phi \in \Phi(G)$ such that $X \nvdash \phi$. In this section we construct a belief network $\mathcal{N} = (\Omega, s, R)$ over graph G such that $\mathcal{N} \nvDash \phi$.

In the physical world, each agent often leaves her "mark" on the beliefs that pass through her. In our construction, a belief is nothing but the collection of such "marks". Thus, we formally define a belief as a subset of vertices:

Definition 9. *Set of beliefs Ω is the powerset of V.*

Definition 10. *Initial state s is the function that maps each vertex into the empty set of beliefs.*

Definition 11. *Set $S \subseteq V$ is called a "shield" of vertex $v \in V$ if $X \vdash v \parallel S$.*

Definition 12. *The set of belief formation rules R of the network $\mathcal{N}$ is the minimal set such that for any $w \in V$, if $S_1, \dots S_n$ are all shields of vertex w, then R contains of all rules of the form $u : \beta \wedge \bigwedge_{i \leq n} v_i : \alpha_i \to w : \beta \cup \{u\}$, where*

1. *$u, v_1, \dots, v_n, w \in V$,*
2. *$\beta, \alpha_1, \dots, \alpha_n \subseteq V$,*
3. *$(v_i, w) \in E$ for each $i \leq n$,*
4. *$(u, w) \in E$,*
5. *$S_i \cap (\alpha_i \cup \{v_i\}) = \varnothing$ for each $i \leq n$.*

The following lemma immediately follows from the above definition of the belief network $\mathcal{N}$:

Lemma 4. *For any $v \in V \setminus B$. If $x \in Mindset_{\mathcal{N}_B}(v)$, then set $x \cap B$ is not empty.* □

Lemma 5. *If $X \vdash A \parallel B$, then $\mathcal{N} \vDash A \parallel B$.*

Proof. We need to prove that $Mindset_{\mathcal{N}_B}(w) \subseteq Mindset_{\mathcal{N}}(w)$ for each $w \in A$. Indeed, the assumption $X \vdash A \parallel B$, by the Monotonicity axiom, implies that $X \vdash w \parallel B$. Thus, set B is a shield of vertex w. Thus, by Lemma 4, all beliefs in network $\mathcal{N}$ outside of set B are "marked" by at least one element of B. Hence, by Definition 12, vertex w will never form any beliefs. Therefore, set $Mindset_{\mathcal{N}_B}(w)$ is empty. □

Theorem 4. *If $\mathcal{N} \vDash A \parallel B$, then $X \vdash A \parallel B$.*

Proof. We divide all vertices of the graph G into two disjoint groups: red vertices and blue vertices. A vertex v is called blue if set $Mindset_{\mathcal{N}_B}(v)$ is empty. Otherwise, vertex v is called *red*.

Lemma 6. *$\Omega \subseteq Mindset_{\mathcal{N}_B}(v)$ for any $v \in B$.*

Proof. See Definition 6. □

Lemma 7 (Red Path). *If path $u_1, u_2, \dots, u_k$ is a directed path of red vertices in graph G and $\gamma \in Mindset_{\mathcal{N}_B}(u_1)$, then $\gamma \cup \{u_1, \dots, u_{k-1}\} \in Mindset_{\mathcal{N}_B}(u_k)$.*

Proof. Induction on k. If $k = 1$, then $\gamma \in Mindset_{\mathcal{N}_B}(u_1)$ by the assumption.

Let $k > 1$. If $u_k \in B$, then $\gamma \cup \{u_1, \dots, u_{k-1}\} \in Mindset_{\mathcal{N}_B}(u_k)$ by Lemma 6. We will now assume that $u_k \notin B$. Thus, since vertex u_k is red, by Definition 12, there must exist vertices $v_1, \dots, v_n$ and beliefs $\alpha_1, \dots, \alpha_n$ such that

1. $(v_i, u_k) \in E$ for each $i \leq n$,
2. $\alpha_i \in Mindset_{\mathcal{N}_B}(v_i)$ for each $i \leq n$, and
3. $S_i \cap (\alpha_i \cup \{v_i\}) = \varnothing$ for each $i \leq n$.

Consider now the rule of our belief network

$$u_{k-1} : \gamma \cup \{u_1, \dots, u_{k-2}\} \wedge \bigwedge_{i \leq n} v_i : \alpha_i \to u_k : \gamma \cup \{u_1, \dots, u_{k-1}\},$$

By the Induction Hypothesis, $\gamma \cup \{v_1, \dots, v_{k-2}\} \in Mindset_{\mathcal{N}_B}(u_{k-1})$. Therefore, $\gamma \cup \{v_1, \dots, v_{k-1}\} \in Mindset_{\mathcal{N}_B}(u_k)$. □

Lemma 8. *If for every shield of a vertex u there is a red directed path (possibly except for vertex u) to vertex u from a vertex in B such that the path does not go through the shield, then vertex u itself is red.*

Proof. Let $S_1, \dots, S_n$ be all the shields of the vertex u. By the Empty Set axiom, $X \vdash v \parallel \varnothing$. Thus, $n > 0$. Let $v_1, \dots, v_n$ be vertices such that $(v_i, u) \in E$ and there is a directed red path $v_i^{(n_i)}, \dots, v_i'', v_i', v_i$, from a vertex $v_i^{(n_i)} \in B$ to vertex v_i that does not go through S_i for each $i \leq n$. In other words, intersection $S_i \cap \left\{v_i^{(n_i)}, \dots, v_i', v_i\right\}$ is empty.

By Lemma 7,

$$\left\{v_i^{(n_i)}, \dots, v_i'\right\} \in Mindset_{\mathcal{N}_B}(v_i). \quad (3)$$

Recall that $n > 0$, thus, by Definition 12, network $\mathcal{N}_B$ contains the rule

$$v_1 : \{v_1^{(n_1)}, \ldots, v_1'\} \wedge \bigwedge_{i \le n} v_i : \{v_i^{(n_i)}, \ldots, v_i'\} \to u : \{v_1^{(n_1)}, \ldots, v_1', v_1\}.$$

Hence, due to (3), $\{v_1^{(n_1)}, \ldots, v_1', v_1\} \in \mathit{Mindset}_{\mathcal{N}_B}(u)$. Thus, set $\mathit{Mindset}_{\mathcal{N}_B}(u)$ is not empty, or, in other words, vertex u is red. $\square$

Due to Lemma 8, the set of all blue vertices, *Blue*, satisfies the conditions of the Shield Wall axiom. Thus, $X \vdash \mathit{Blue} \parallel B$. By the assumption, $A \subseteq \mathit{Blue}$. Therefore, by the Monotonicity axiom, $X \vdash A \parallel B$. $\square$

Theorem 5. *$X \vdash \phi$ if and only if $\mathcal{N} \vDash \phi$, for any formula $\phi \in \Phi(A)$.*

Proof. Induction on the structural complexity of formula ϕ. The base case follows from Lemma 5 and Theorem 4. The induction step follows in the standard way from maximality and consistency of the set X. $\square$

Theorem 6 (Completeness). *For any graph G and for any $\phi \in \Phi(G)$. if $\nvdash \phi$, then there is a belief network $\mathcal{N}$ over graph G such that $\mathcal{N} \nvDash \phi$.*

Proof. Suppose that $\nvdash \phi$. Consider any maximal consistent set X containing $\neg\phi$. Let $\mathcal{N}$ be the canonical belief network defined in this section. By Theorem 5, $\mathcal{N} \nvDash \phi$. $\square$

9 Conclusion

In this paper, we have studied fault tolerance properties of belief formation networks under an assumption that once a belief is formed by an agent, it is never forgotten by the agent. One can introduce "forgetting" belief networks by adding a second type of transition to Definition 4. During such a new transition $x \multimap_{\mathcal{N}} y$, the active vertex v_0 "forgets" some belief α. Thus, $y(v_0) = x(v_0) \setminus \{\alpha\}$. It is easy to show, however, that the soundness and completeness results in this paper can be generalized to the "forgetting" belief networks.

References

1. Shafer, G., Shenoy, P.P.: Probability propagation. Ann. Math. Artif. Intell. 2, 327–351 (1990)
2. Cano, J.E., Delgado, M., Moral, S.: An axiomatic framework for propagating uncertainty in directed acyclic networks. Int. J. Approx. Reasoning 8(4), 253–280 (1993)
3. More, S.M., Naumov, P.: The Functional Dependence Relation on Hypergraphs of Secrets. In: Leite, J., Torroni, P., Ågotnes, T., Boella, G., van der Torre, L. (eds.) CLIMA XII 2011. LNCS, vol. 6814, pp. 29–40. Springer, Heidelberg (2011)
4. More, S.M., Naumov, P.: Logic of secrets in collaboration networks. Ann. Pure Appl. Logic 162(12), 959–969 (2011)

5. Donders, M., Miner More, S., Naumov, P.: Information Flow on Directed Acyclic Graphs. In: Beklemishev, L.D., de Queiroz, R. (eds.) WoLLIC 2011. LNCS, vol. 6642, pp. 95–109. Springer, Heidelberg (2011)
6. Miner More, S., Naumov, P.: Hypergraphs of Multiparty Secrets. In: Dix, J., Leite, J., Governatori, G., Jamroga, W. (eds.) CLIMA XI. LNCS (LNAI), vol. 6245, pp. 15–32. Springer, Heidelberg (2010)
7. Sutherland, D.: A model of information. In: Proceedings of Ninth National Computer Security Conference, pp. 175–183 (1986)
8. Marcus, L.: Preface. Electr. Notes Theor. Comput. Sci. 258(2), 1–2 (2009)
9. Bonakdarpour, B., Mailbaum, T. (eds.): Proceedings of the 2nd International Workshop on Logical Aspects of Fault-Tolerance, LAFT (2011)
10. Ezekiel, J., Lomuscio, A.: Combining fault injection and model checking to verify fault tolerance in multi-agent systems. In: Sierra, C., Castelfranchi, C., Decker, K.S., Sichman, J.S. (eds.) AAMAS (1), IFAAMAS, pp. 113–120 (2009)
11. Castro, P.F., Maibaum, T.S.E.: Reasoning about System-Degradation and Fault-Recovery with Deontic Logic. In: Butler, M., Jones, C., Romanovsky, A., Troubitsyna, E. (eds.) Fault Tolerance. LNCS, vol. 5454, pp. 25–43. Springer, Heidelberg (2009)

Large-Scale Cost-Based Abduction in Full-Fledged First-Order Predicate Logic with Cutting Plane Inference

Naoya Inoue and Kentaro Inui

Tohoku University, 6-3-09 Aramaki Aza Aoba, Aobaku, Sendai 980-8579, Japan
{naoya-i,inui}@ecei.tohoku.ac.jp

Abstract. Abduction is inference to the best explanation. Abduction has long been studied intensively in a wide range of contexts, from artificial intelligence research to cognitive science. While recent advances in large-scale knowledge acquisition warrant applying abduction with large knowledge bases to real-life problems, as of yet no existing approach to abduction has achieved both the efficiency and formal expressiveness necessary to be a practical solution for large-scale reasoning on real-life problems. The contributions of our work are the following: (i) we reformulate abduction as an Integer Linear Programming (ILP) optimization problem, providing full support for first-order predicate logic (FOPL); (ii) we employ Cutting Plane Inference, which is an iterative optimization strategy developed in Operations Research for making abductive reasoning in full-fledged FOPL tractable, showing its efficiency on a real-life dataset; (iii) the abductive inference engine presented in this paper is made publicly available.

Keywords: abduction, cost-based abduction, cutting plane inference, integer linear programming.

1 Introduction

Abduction is inference to the best explanation. Abduction has long been studied in a wide range of contexts. For example, abduction has been viewed as a promising framework for describing the mechanism of human perception [1–4, etc.]. The idea is that the declarative nature of abduction enables us to infer the most plausible, implicitly stated information combining several types of inference, and pieces of explicitly observed information, as humans do. Hobbs et al. [2] showed the process of natural language interpretation can reasonably be described as abductive inference; finding the lowest-cost abductive proof provides the solutions to a broad range of natural language pragmatics problems, such as word sense disambiguation, anaphora, and metonymy resolution. It will be a significant contribution for such research areas if we could provide an efficient abductive reasoning engine which scales to large problems.

In this paper, we explore first-order predicate logic-based abduction *with large knowledge bases (KBs) for solving "real-life" problems*. While the lack of world knowledge resources hampered applying abduction to real-life problems in the 1980s and 1990s, a number of techniques that acquire world knowledge resources have been developed in the last decade [5–9, etc.]. Consequently, several researchers start applying abduction to real-life problems, exploiting large KBs. For instance, inspired by Hobbs et al. [2], Ovchinnikova et al. [10] propose an abduction-based natural language processing framework using forty thousands of axioms extracted from the popular ontological resources, WordNet [5] and FrameNet [6]. They evaluate their approach on the real-life natural language processing task of Recognizing Textual Entailment (RTE) [11].

However, in order to apply large-scale abductive inference to real-life problems, we still need to address the following issue: *how to search for the best explanation efficiently*. Abduction is known to be an NP-hard problem in general [12]; this hampers the application of abduction with large knowledge resources to real-life problems. In fact, Ovchinnikova et al. [10] report that the Mini-TACITUS abductive reasoning system [13] could not search the entire search space of explanations within 30 minutes in most of the RTE problems in their experiments. In the literature, many researchers have tried to overcome abduction's inefficiency by a range of methods from approximation [14–16, etc.] to exact inference [17, 18, etc.]. However, most of the proposed methods are optimized for propositional logic in principle. Inoue and Inui [18] provides an efficient approach to first-order predicate logic (FOPL)-based abduction, showing superior efficiency to Mini-TACITUS system [13]; however, it does not provide full support of FOPL (e.g. negation is not supported). In addition, as the reader will see in Sec. 3.2 and Sec. 4, it does not scale to larger problems due to the intractability emerging from the use of FOPL inference.

In this paper, we address this issue with the following contributions:

(i) we extend Inoue and Inui [18]'s Integer Linear Programming (ILP)-based inference method, *providing full support for FOPL including negation*;
(ii) we describe how Cutting Plane Inference, an iterative optimization strategy developed in Operations Research, can be exploited for *making abductive reasoning in full-fledged FOPL tractable*, showing its efficiency by *providing evaluation on a large, real-life dataset*;
(iii) the abductive inference engine presented in this paper is made publicly available.

The structure of our paper is as follows. We start with a brief introduction of *cost-based* abduction, where the quality of explanation is evaluated by some cost function (Sec. 2.1). We then briefly describe Inoue and Inui [18]'s ILP-based formulation of cost-based abduction (Sec. 2.2). In the next section, we first describe how their formalization can be extended for handling negation (Sec. 3.1). We then show how Cutting Plane Inference (CPI) enables us to apply abductive reasoning in full-fledged FOPL with large KBs (Sec. 3.2). Finally, we evaluate the efficiency of our CPI-based framework on a large, real-life problem of natural language processing (Sec. 4), and give a comparison of our work with the prior implementations of cost-based abduction (Sec. 5).

2 Background

2.1 Cost-Based Abduction

Abduction is inference to the best explanation. Formally, logical abduction is defined as follows:

- **Given:** Background knowledge B, and observations O, where both B and O are sets of first-order logical formulas
- **Find:** A *hypothesis* (or *explanation*) H such that $H \cup B \models O, H \cup B \not\models \bot$, where H is a set of first-order logical formulas. We say that p is *hypothesized* if $H \cup B \models p$, and that p is *explained* if $(\exists q)\ q \rightarrow p \in B$ and $H \cup B \models q$.

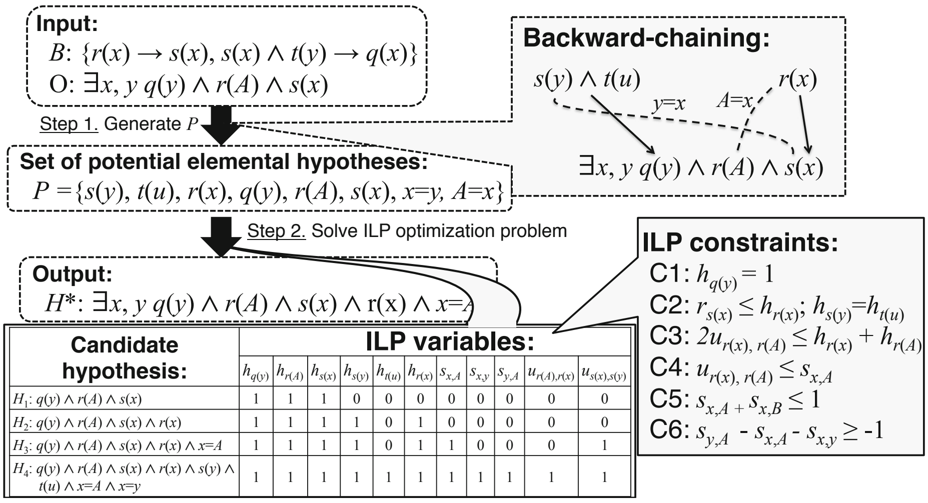
Typically, there exist several hypotheses H explaining O. We call each of them a *candidate hypothesis*, and each literal in a hypothesis an *elemental hypothesis*. *Cost-based abduction* (CBA) is defined as abduction which identifies the minimum-cost explanation H^* among a set $\mathcal{H}$ of candidate explanations. Formally, we find $H^* = \arg\min_{H \in \mathcal{H}} cost(H)$, where *cost* is a function $\mathcal{H} \rightarrow \mathbb{R}$, which is called the *cost function*. Several kinds of cost functions have been proposed in prior work on cost-based abduction [1, 2, 19, 20, etc.]. For instance, Hobbs et al. [2] use a cost function that favors a fewest elemental hypotheses and a shorter proof path. The function is represented by the sum of costs of elemental hypotheses, where the cost of elemental hypothesis is in proportion to the distance from the observations on a proof graph.

2.2 ILP-Based Formulation of CBA

In this section, we briefly review Inoue and Inui [18]'s ILP formulation of cost-based abduction. The main idea is that explanation finding in CBA can be regarded as the weighted combinatorial optimization problem of literals. They thus formulate CBA as an ILP optimization problem, where the search space of CBA is represented as ILP variables and constraints, and the cost function is used as the ILP objective, in order to exploit the state-of-the-art combinatorial optimization technology in Operations Research.

Here we give an intuitive description of their approach, using the diagram illustrated in Figure 1. Given an abduction problem (i.e., background knowledge B and observations O), they first create set P of *potential elemental hypotheses*, a set of instantiated literals that are potentially included as constituents of explanations of O (i.e. Step 1 in Figure 1). This procedure is called the *search-space generation*. For enumerating potential elemental hypotheses, they apply *backward-chaining* with axioms in B, and instantiate the body of axioms. For instance, in Figure 1, we add two instantiated literals $s(y)$, $t(u)$ to P, which might be the explanations of $q(y) \in O$, performing backward-chaining on $q(y)$ with axiom $s(x) \wedge t(y) \rightarrow q(x)$. Using the set P, they represent the search space of explanations as an ILP optimization problem as follows.

Hypothesis Inclusion: For each $p \in P$, ILP variables $h_p \in \{0, 1\}$ are introduced to represent whether p is hypothesized ($h_p = 1$) or not ($h_p = 0$). For example, H_2 in Figure 1 holds $h_{r(x)} = 1$, where $r(x)$ is included in H_2.


Input:
B: {r(x) → s(x), s(x) ∧ t(y) → q(x)}
O: ∃x, y q(y) ∧ r(A) ∧ s(x)
Step 1. Generate P
Backward-chaining:
s(y) ∧ t(u)
r(x)
y=x
A=x
∃x, y q(y) ∧ r(A) ∧ s(x)
Set of potential elemental hypotheses:
P = {s(y), t(u), r(x), q(y), r(A), s(x), x=y, A=x}
Step 2. Solve ILP optimization problem
Output:
H*: ∃x, y q(y) ∧ r(A) ∧ s(x) ∧ r(x) ∧ x=A
ILP constraints:
C1: h_q(y) = 1
C2: r_s(x) ≤ h_r(x); h_s(y)=h_t(u)
C3: 2u_r(x), r(A) ≤ h_r(x) + h_r(A)
C4: u_r(x), r(A) ≤ s_x,A
C5: s_x,A + s_x,B ≤ 1
C6: s_y,A - s_x,A - s_x,y ≥ -1

Candidate hypothesis:	**ILP variables:**										
	$h_{q(y)}$	$h_{r(A)}$	$h_{s(x)}$	$h_{s(y)}$	$h_{t(u)}$	$h_{r(x)}$	$s_{x,A}$	$s_{x,y}$	$s_{y,A}$	$u_{r(A),r(x)}$	$u_{s(x),s(y)}$
H_1: $q(y) \land r(A) \land s(x)$	1	1	1	0	0	0	0	0	0	0	0
H_2: $q(y) \land r(A) \land s(x) \land r(x)$	1	1	1	1	0	1	0	0	0	0	0
H_3: $q(y) \land r(A) \land s(x) \land r(x) \land x{=}A$	1	1	1	1	0	1	1	0	0	0	1
H_4: $q(y) \land r(A) \land s(x) \land r(x) \land s(y) \land t(u) \land x{=}A \land x{=}y$	1	1	1	1	1	1	1	1	1	1	1

Fig. 1. Summary of Inoue and Inui [18]'s ILP-based approach

Cost of Hypothesis: To calculate $cost(H)$ of each candidate hypothesis H, they use the cost function defined in Hobbs et al. [2]'s weighted abduction. In weighted abduction, $cost(H)$ is represented by the sum of the costs for $p \in P$ such that p is hypothesized (i.e., $h_p = 1$) but not explained. They thus introduce another ILP variable $r \in \{0, 1\}$ for representing whether p is explained ($r_p = 1$) or not ($r_p = 0$). The final objective function of the ILP problem is given by: **min.** $cost(H) = \sum_{p \in \{p | p \in P, h_p = 1, r_p = 0\}} cost(p)$, where $cost(p)$ is the cost of a literal p. The cost of a literal is determined by the total unreliability of backward-inferences that are used for abducing the literal. This optimization amounts to Step 2 in Figure 1. They optimize this objective with the six types of ILP constraints as shown in Figure 1. In the rest of this section, we describe only two of them, which are necessary for the readers to follow the discussion below due to spatial limitations.

To handle first-order predicate logic, variable substitution must be taken into account to control the unification of elemental hypotheses. For representing the status of unification, they introduce another class of variables $u_{p,q} \in \{0, 1\}$ for each $p, q \in P$, which takes 1 if p is unified with q. Concerning variable substitution, another type of ILP variables s are introduced, where $s_{x,y} = 1$ if x is substituted for y, 0 otherwise. s is symmetric (i.e., $s_{x,y} = s_{y,x}$). In Figure 1, $u_{r(x),r(A)}$ and $s_{x,A}$ are introduced. In H_3, the variables $u_{r(x),r(A)}$, $s_{x,A}$ are set to 1 because $r(x)$ is unified with $r(A)$, and $x = A$ is assumed. Note that unification of $r(x)$ with $r(A)$ is allowed only if x are substituted with A. In addition, the substitution relation must be transitive (e.g. $y = A$ must hold if $x = y$ and $x = A$ hold). For keeping those consistency, they impose two ILP constraints:

Constraint 4[1]**:** Two literals $q(x_1, x_2, ..., x_n) \equiv q(\mathbf{x})$ and $q(y_1, y_2, ..., y_n) \equiv q(\mathbf{y})$ are allowed to be unified (i.e., $u_{q(\mathbf{x}),q(\mathbf{y})} = 1$) only if all variable substitutions x/y involved in the unification are activated (i.e., $s_{x_i,y_i} = 1$ for all $i \in \{1, 2, ..., n\}$). This can be expressed as:

$$n \cdot u_{q(\mathbf{x}),q(\mathbf{y})} \leq \sum_{i=1}^{n} s_{x_i,y_i} \tag{1}$$

In Figure 1, the constraint $u_{r(x),r(A)} \leq s_{x,A}$ is generated since x needs to be substituted for A when $r(x)$ and $r(A)$ are unified.

Constraint 6: s is transitive; namely $s_{x,z}$ must be 1 if $s_{x,y} = 1$ and $s_{y,z} = 1$. This can be expressed as the following constraints[2]:

$$s_{x,z} - s_{x,y} - s_{y,z} \geq -1 \tag{2}$$
$$-s_{x,z} + s_{x,y} - s_{y,z} \geq -1 \tag{3}$$
$$-s_{x,z} - s_{x,y} + s_{y,z} \geq -1 \tag{4}$$

They generate $O(n^3)$ transitivity constraints, where n is the number of logical terms. As the reader will see in Sec. 4, this makes inference intractable in large-scale inference. We propose how this drawback can be overcome by exploiting Cutting Plane Inference in Sec. 3.2.

3 Full-Fledged First-Order Predicate Logic Abduction with Cutting Plane Inference

In this section, we first present an extended formulation of Inoue and Inui [18] over full fledged first-order predicate logic. We then describe how to apply Cutting Plane Inference to best-explanation finding, for avoiding the intractability that arises from the extension and generation of transitivity constraints.

3.1 Handling Negation for Supporting Full-Fledged FOPL

The ILP formulation described in Sec. 2.2 does not provide full support for FOPL; it cannot represent negation. As a background knowledge, they accept only Horn clauses as axioms, and positive literals as observations. However, the capability of handling negations is crucial for a wide range of abductive reasoning. For example, in abduction-based natural language interpretation, one can imagine that it needs to handle negated expressions, such as *"I don't know."*

In the rest of this section, we give two formulations for expressing negative literals (e.g. $\neg p$) and inequality of variables (e.g. $x \neq y$), for making Inoue and

[1] The numbers of constraints correspond to the numbers presented in [18].

[2] Inoue and Inui [18] introduce the form of inequality $s_{x,y}+s_{y,z} \leq 2\cdot s_{x,z}$ as transitivity constraints. However, this constraint does not appropriately represent transitivity; thus we replace them with inequalities (2)–(4).

Inui [18]'s framework support full FOPL. By handling negative literals, we are able to assume the background knowledge to be a set of full-fledged first-order logical formulae represented in the clausal normal form, i.e. a set of implicitly skolemized disjunctive literals (e.g. $\{\neg p(x), q(x)\}$, $\{\neg p(x), q(f(x), x), r(f(x))\}$). To eliminate disjunctions from a clause, we convert each clause $\{L_1, \ldots, L_n\}$ to a set of single literal-headed clauses of the form of $\neg L_1 \wedge \ldots \wedge \neg L_{i-1} \wedge \neg L_{i+1} \wedge \ldots \wedge \neg L_n \rightarrow L_i$ for each L_i. Henceforth, we call a clause of this form an *axiom*, the right hand side the *head*, and the left hand side the *body*.

First, consider the case where two literals $p(x)$ and $\neg p(y)$ are in set P of potential elemental hypotheses such that $p(x)$ and $p(y)$ are unifiable. We want to prohibit the two literals from being hypothesized simultaneously if x is substituted with y. One can imagine a simple inequality such as $h_{p(x)} + h_{\neg p(y)} \leq 1$, however, it is not enough because $p(x)$ and $p(y)$ can be both hypothesized (i.e. $h_{p(x)}$ and $h_{\neg p(y)}$ can be 1 simultaneously) if x is not substituted for y. This constraint can be correctly represented by incorporating the ILP variable $s_{x,y}$ which represents the variable substitution of x for y:

Proposed Constraint 1: Two literals $q(x_1, x_2, ..., x_n) \equiv q(\mathbf{x})$ and $\neg q(y_1, y_2, ..., y_n) \equiv \neg q(\mathbf{y})$ cannot be both hypothesized ($h_{q(\mathbf{x})} = 1$ and $h_{\neg q(\mathbf{y})} = 1$) if variable substitutions x_i/y_i are activated ($s_{x_i,y_i} = 1$) for all $i \in \{1, 2, ..., n\}$. This can be expressed as: $h_{q(\mathbf{x})} + h_{\neg q(\mathbf{y})} + \sum_{i=1}^{n} s_{x_i,y_i} \leq 1 + n$. Note that the case where $\mathbf{x} = \mathbf{y}$ reduces to: $h_{q(\mathbf{x})} + h_{\neg q(\mathbf{x})} \leq 1$. This type of constraint grows in $O(nm)$ for each predicate p, where n is the number of positive instantiation of p in P, and m is the number of negative instantiation of p in P.

The important question here is how to find the pair $q(\mathbf{x})$ and $\neg q(\mathbf{y})$. In order to find potential contradictions, one can perform forward reasoning. However, the problem is how to control the overall search process because chaining might be repeated infinitely. Terminating the search at a certain depth could miss potential contradictions. Given $q(\mathbf{x})$ and $r(\mathbf{x})$ as potential elemental hypotheses, for example, we may fail to find that $\neg q(\mathbf{x})$ could be derived from $r(\mathbf{x})$ in several forward-chaining steps. This is a long-standing problem in logic-based reasoning. One can address this problem by adopting some heuristics such as A* search.

We now describe how the inequality of variables, where two variables are prohibited to unify due to some constraints, can be formulated. This kind of constraint is also important for abductive inference. For example, imagine natural language interpretation systems. Given the sentence "*A girl sent a present to another girl*", it is desirable to express that the two girls must not be identical. Such a constraint can be expressed straightforwardly in the ILP formulation:

Proposed Constraint 2: For each pair of (existentially quantified) variables x and y in set P of potential elemental hypotheses that must not be identical (i.e. $x \neq y$), introduce the following equality:

$$s_{x,y} = 0. \tag{5}$$

In our experiments, we use the six types of constraints described in Sec. 2.2 and two constraints newly introduced above.

Algorithm 1. CPI4CBA(Background Knowledge **B**, Observation **O**)

```
1:  $(\Psi, I) \leftarrow$ createBaseILP$(B, O)$
2:  repeat
3:      $S \leftarrow$ solveILP$(\Psi, I)$; $V \leftarrow \{\}$
4:      for $(x, y) \in$ unifiedTerms$(S)$ do
5:          for $z \in$ termsUnifiableWith$(x) \cup$ termsUnifiableWith$(y)$ do
6:              if $(s_{x,z} = 0$ and $s_{y,z} = 1)$ or $(s_{x,z} = 1$ or $s_{y,z} = 0)$ then
7:                  $V \leftarrow V \cup \{-s_{x,y} - s_{x,z} + s_{y,z} \geq -1, -s_{x,y} + s_{x,z} - s_{y,z} \geq -1\}$
8:              end if
9:          end for
10:     end for
11:     $I \leftarrow I \cup V$
12: until $V \neq \phi$
```

3.2 Cutting Plane Inference for CBA

The major drawback of the ILP formulation is that it needs to generate $O(n^3)$ transitivity constraints, where n is the number of logical terms, because we perform inference over FOPL-based representation. That makes inference intractable (see Sec. 4 for empirical evidence) because it generates an ILP optimization problem that has quite a large number of constraints. Moreover, handling negation quadratically increases Proposed Constraint 1.

How do we overcome this drawback? The idea is that "all the transitivity constraints may not be violated all at once; so we gradually optimize and add transitivity constraints *if violated* in an iterative manner." More formally, we propose to apply *Cutting Plane Inference (CPI)* to the CBA problems. CPI is an exact inference optimization technique that is originally developed for solving large linear programming (LP) problems in Operations Research [21]. CPI has been successfully applied to a wide range of constrained optimization problems where constraints are very large [22–25, etc.], from probabilistic deductive inference problems [23] to machine learning problems [24]. To the best of our knowledge, however, our work is the first successful work to apply CPI to abductive reasoning tasks. In principle, CPI solves optimization problem in an iterative manner as follows: it solves an optimization problem without constraints, and then adds violated constraints to the optimization problem. When the iteration terminates, it guarantees solutions to be optimal. The proposed algorithm, called *CPI4CBA*, is also an exact inference framework.

How do we apply the technique of CPI to cost-based abduction problems? Intuitively, we iterate the following two steps: (i) solving an abduction problem without enforcing transitivity on logical atomic terms, and (ii) generating transitivity constraints dynamically when transitiveness of unification is violated (e.g. $x = y \wedge y = z \wedge z \neq x$). The iteration terminates if there is no violated unification transitivity. The pseudo-code is given in Algorithm 1. In line 1, we first create an ILP optimization problem described in Sec. 2.2 and Sec. 3.1 but without transitivity constraints (i.e. Constraint 6), where Ψ denotes a set of ILP variables, and I denotes a set of ILP constraints. In line 2–12, we repeat: checking consistency of unification transitiveness, adding constraints for violated transitiveness, and re-optimizing. In line 3, we find the solution S for the current ILP optimization problem. Then, for each pair (x, y) of logical atomic terms unified in the solution S (line 4), find the logical term z which is unifiable with x or y (line 5). If the transitive relation x, y with respect to z is violated (i.e. $s_{x,z} = 0 \wedge s_{y,z} = 1$ or $s_{x,z} = 1 \wedge s_{y,z} = 0$), then we generate constraints for preventing this violation, and keep it in set V of constraints (line 6–8). Finally, we again perform an ILP optimization with newly generated constraints (line 11 and 3). The iteration ends when there is no violated transitiveness (line 12).

The key advantages of CPI4CBA is that it can reduce the time of search-space generation, and it is also expected to reduce the time of ILP optimization. CPI4CBA does not generate all the transitivity constraints before optimization, which saves the time for search-space generation. In addition, optimization problems that we solve would become smaller than the original problem in most cases, because not all the transitivity constraints may not be necessary to be considered. In the worst case, we need to solve the optimization problem that is same as the original one; but in most cases we found out that we do not need to. We will show its empirical evidence through large-scale evaluation in Sec. 4.

4 Runtime Evaluation

How much does CPI improve the *runtime* of ILP-based reasoner? Does CPI scale to larger real-life problems? To answer these questions, we evaluated the CPI4CBA algorithm in two settings: (i) **STORY**, the task of plan recognition, and (ii) **RTE**, the popular, knowledge-intensive, real-life natural language processing task of *Recognizing Textual Entailment* (RTE). While most of the existing abductive reasoning systems are evaluated on rather small, and/or artificial datasets [26–28, etc.], our evaluation takes a real-life, much larger datasets (see Sec. 4.1). In our experiments, we compare our system with: (i) Inoue and Inui's formulation [18], and (ii) the systems [26, 28, 29] based on Markov Logic Networks (MLNs) [30]. For our experiments, we have used a 12-Core Opteron 6174 (2.2GHz) 128 GB RAM machine. We used Gurobi Optimizer[3], which is an efficient ILP solver. It is commercial but an academic license is freely available.

4.1 Settings

STORY: For this setting, we have used Ng and Mooney [31]'s story understanding dataset, which is widely used for evaluation of abductive plan recognition systems [26–28]. In this task, we need to abductively infer the top-level plans of characters from actions which are represented by the logical forms (e.g. $\mathit{getting\_off}(\mathit{Getoff16}) \wedge \mathit{agent\_get\_off}(\mathit{Getoff16}, \mathit{Fred16}) \wedge \mathit{name}(\mathit{Fred16}, \mathit{Fred})$). The dataset consists of 50 plan recognition problems and 107 background Horn clauses (e.g. $\mathit{go\_step}(r, g) \wedge \mathit{going}(g) \rightarrow \mathit{robbing}(r)$). The dataset contains on

[3] http://www.gurobi.com/

average 12.6 literals in observed logical forms. To make the predicates representing top-level plans (e.g. shopping, robbing) disjoint, we generated 73 disjointness axiom by using the formulation[4] described in Sec. 3.1. Note that in Inoue and Inui [18]'s evaluation, disjointness constraints are not used. Regarding a cost function, we followed Hobbs et al. [2]'s weighted abduction theory. For each axiom, we have set the sum of the axiom weights equal to 1.2 (e.g. $\mathit{inst\_shopping}(s)^{0.6} \wedge \mathit{store}(t,s)^{0.6} \rightarrow \mathit{shopping\_place}(t)$).

RTE: For observations (input), we employed the second challenge of RTE dataset[5]. In the task of RTE, we need to correctly determine whether one text (called *text*, or T) entails another (called *hypothesis*, or H) or not. The dataset consists of development set and test set, each of which includes 800 natural language text-hypothesis pairs. We have used all of the 800 texts from test set. We have converted texts into logical forms presented in [32] using the Boxer semantic parser [33]. The number of literals in observations is 29.6 literals on average. For background knowledge, we have extracted 289,655 axioms[6] from WordNet 3.0 [5], and 7,558 axioms from FrameNet 1.5 [6] following [10]. In principle, the WordNet knowledge base contains several kinds of lexical relations between words, such as IS-A, ontological relations (e.g. $\mathit{dog}(x) \rightarrow \mathit{animal}(x)$). FrameNet knowledge bases contain lexeme-to-frame mappings, frame-frame relations, etc. For example, the mapping from surface realization "give to" to a frame "Giving" is given by: $\mathit{Giving}(e_1, x_1, x_2, x_3)^{1.3} \wedge \mathit{donor}(e_1, x_1)^{0.1} \wedge \mathit{recipient}(e_1, x_2)^{0.2} \wedge \mathit{theme}(e_1, x_3)^{0.1} \rightarrow \mathit{give}(e_1, x_1, x_3) \wedge \mathit{to}(e_2, e_1, x_2)$. We again followed Hobbs et al. [2]'s weighted abduction theory for calculating the cost of hypothesis. We calculated the costs by following Ovchinnikova et al. [10] in this setting.

4.2 Results and Discussion

The reasoner was given a 2-minute time limit for each inference step (i.e. search-space generation and ILP optimization). In Table 1, we show the results of each setting for two inference method in Table 1: (i) *IAICBA*: the inference method without CPI (i.e. Inoue and Inui [18]'s formulation with proposed constraints 1 and 2), and (ii) *CPI4CBA*: inference method with CPI (i.e. our proposal). In order to investigate the relation between the size of search space and the runtime, we show the results for each depth, which we used for limiting the length of backward-chaining. In the "Generation" column, we show the runtime that is taken for search-space generation in seconds averaged over all problems whose search-space generation is finished within 2 minutes. In the parenthesis, we show the percentage of those problems. In the column "ILP inf", we show the runtime of ILP optimization averaged on only problems such that both search-space generation and ILP optimization are finished within 2 minutes, as well as the percentage of those problems (e.g. 80 % means "for 80 % of all the

[4] For example, we generate $h_{robbing(x)} + h_{shopping(y)} + s_{x,y} \leq 2$.
[5] `http://pascallin.ecs.soton.ac.uk/Challenges/RTE2/`
[6] Extracted relations are: word-to-synset mapping, hypernym-hyponym, cause-effect, entailment, derivational, instance-of relations.

Table 1. The results of averaged inference time in **STORY** and **RTE**

Setting	Method	Depth	Generation [sec.]	ILP inf [sec.]	# of ILP cnstr
STORY	IAICBA	1	0.02 (100.0 %)	0.60 (100.0 %)	3,708
		2	0.12 (100.0 %)	5.34 (100.0 %)	23,543
		3	0.33 (100.0 %)	8.11 (100.0 %)	50,667
		∞	0.35 (100.0 %)	9.00 (100.0 %)	61,122
	CPI4CBA	1	0.01 (100.0 %)	0.34 (100.0 %)	784 (Δ 451)
		2	0.07 (100.0 %)	4.15 (100.0 %)	7,393 (Δ 922)
		3	0.16 (100.0 %)	3.36 (100.0 %)	16,959 (Δ 495)
		∞	**0.22 (100.0 %)**	**5.95 (100.0 %)**	24,759 (Δ 522)
RTE	IAICBA	1	0.01 (100.0 %)	0.25 (99.7 %)	1,104
		2	0.08 (100.0 %)	2.15 (98.1 %)	5,185
		3	0.56 (99.9 %)	5.66 (93.0 %)	16,992
		∞	4.78 (90.7 %)	15.40 (60.7 %)	36,773
	CPI4CBA	1	0.01 (100.0 %)	0.05 (100.0 %)	269 (Δ 62)
		2	0.04 (100.0 %)	0.35 (99.6 %)	1,228 (Δ 151)
		3	0.09 (100.0 %)	1.66 (99.0 %)	2,705 (Δ 216)
		∞	**0.84 (98.4 %)**	**11.73 (76.9 %)**	10,060 (Δ 137)

problems, search-space generation was finished within 2 minutes, and so was ILP inference.”). In the “# of ILP cnstr” column, we show the averaged number of generated ILP constraints. Concerning CPI4CBA, the number denotes the total number of constraints considered in the end, including the constraints added by CPI. The number marked by Δ indicates the number of constraints that are added during CPI (i.e. how many times line 7 in Algorithm 1 executed).

Overall, the runtimes in both search-space generation and ILP inference are dramatically improved from IAICBA to CPI4CBA in both settings, as shown in Table 1. In addition, CPI4CBA can find optimal solutions in ILP inference for more than 76 % of the problems, even for depth ∞. This indicates that CPI4CBA scales to larger problems. From the results of IAICBA in **RTE** settings, we can see the significant bottleneck of Inoue and Inui [18]’s formulation in large-scale reasoning: the time of search-space generation. The search-space generation could be done within 2 minutes for only 90.7 % of the problems. CPI4CBA successfully overcomes this bottleneck. CPI4CBA is clearly advantageous in the search-space generation because it is not necessary to generate transitivity constraints, an operation that grows cubically before optimization.

In addition, CPI4CBA reduces the time of ILP inference significantly. In ILP inference, CPI did not guarantee the reduction of inference time in theory; *however*, as shown in Table 1, we found that the number of ILP constraints actually used is much less than the original problem. CPI4CBA successfully reduces the complexity of the ILP optimization problems in practice. This is also supported by the fact that CPI4CBA keeps 76.9% in “ILP inf” for Depth = ∞ because it solves very large ILP optimization problems that fail to be generated in IAICBA.

Finally, we compare our results with other existing systems. Overall, the presented reasoner is dramatically faster than the other systems. First, we immediately see that the proposed method is more efficient than Inoue and Inui

[18]'s formulation (i.e. IAICBA). Regarding the MLN-based systems [26, 28, 29], our results are comparable, or more efficient than the existing systems. For the **STORY** setting, Singla and Mooney [28] report the results of two systems with an exact inference technique using CPI for MLNs [23]: (i) Kate and Mooney [26]'s approach: 2.93 seconds, and (ii) Singla and Mooney [28]'s approach: 0.93 seconds[7]. MLN-based approaches seem to be reasonably efficient for small datasets. However, it does not scale to larger problems; for the **RTE** setting, Blythe et al. [29] report that only 28 from 100 selected RTE-2 problems could be run to completion. The processing time was 7.5 minutes on average [personal communication]. On the other hand, our method solves 76.9% of all the problems, where suboptimal solutions are still available for the rest of 21.5%, and it takes only 0.84 seconds for search-space generation, and 11.73 seconds for ILP inference.

5 Related Work

A number of methods attempting to efficiently find the best explanation have been proposed [15–18, 34, 35, etc.]; however, most of them focus on improving the inefficiency of propositional logic-based abduction. Although propositionalization techniques are available for applying these methods to FOPL abduction, it will lead to an exponential growth of ground instances. Hence they would not scale to larger problems for FOPL abduction with large KBs.

Recently, Markov Logic Networks (MLNs) [30] are used for emulating abduction [26, 28, 29, etc.]. They provide full support of first-order predicate logic; however, MLN-based approaches require special procedures to convert abduction problems into deduction problems because of the deductive nature of MLNs. The pioneering work of MLN-based abduction [26] converts background axioms by (i) reversing implication and (ii) constructing axioms representing mutual exclusiveness of explanation (e.g. the set of background knowledge axioms $\{p_1 \rightarrow q, p_2 \rightarrow q, p_3 \rightarrow q\}$ is converted into the following MLN formulae: $q \rightarrow p_1 \vee p_2 \vee p_3$, $q \rightarrow \neg p_1 \vee \neg p_2$, $q \rightarrow \neg p_1 \vee \neg p_3$ etc.). MLN-based approaches suffer from the inefficiency of inference due to the increase of converted axioms.

One important work for FOPL abduction is Inoue and Inui [18]'s approach formulating first-order predicate logic abduction as an ILP optimization problem. However, as mentioned in Sec. 1 and Sec. 2.2, this formulation has two significant drawbacks for large-scale reasoning on real-life problems: (i) the combinatorial growth of transitivity constraints which arises from support for FOPL (see Sec. 3.2), (ii) negation is not supported.

6 Conclusion

We have proposed an ILP-based formulation for cost-based abduction in full-fledged first-order predicate logic, extending Inoue and Inui [18]'s formulation.

[7] This is the result of MLN-HC in [28]. MLN-HCAM cannot be directly compared with our results, since the search space is different from our experiments because they unify some assumptions in advance to reduce the search space.

Compared to prior work, our method is more expressive and efficient. Although full-fledged FOPL reasoning is computationally expensive, our proposed optimization strategy CPI brings us to a significant boosting of the efficiency of the reasoner. We have evaluated our method on two datasets, including real-life problems (i.e. RTE dataset with axioms generated from WordNet and FrameNet). Our evaluation revealed that our inference method CPI4CBA was highly efficient than other existing systems. In future work, we will develop methods for automatic tuning of costs of elemental hypotheses. Specifically, we plan to represent the cost function as a weighted linear feature function, and then apply a standard linear training algorithm such as perceptrons. Also, we will evaluate the abduction-based framework in terms of the prediction accuracy on real-life tasks. We intend to apply abduction to co-reference resolution, the task of identifying referential relations in natural language texts, as a first step. The abductive inference engine presented in this paper is made publicly available.

Acknowledgments. This work was partially supported by Grant-in-Aid for JSPS Fellows (22-9719) and Grant-in-Aid for Scientific Research (23240018). The authors would like to thank the reviewers for their insightful comments.

References

1. Charniak, E., Goldman, R.P.: A Probabilistic Model of Plan Recognition. In: AAAI, pp. 160–165 (1991)
2. Hobbs, J.R., Stickel, M., Martin, P., Edwards, D.: Interpretation as abduction. Artificial Intelligence 63, 69–142 (1993)
3. Shanahan, M.: Perception as Abduction: Turning Sensor Data Into Meaningful Representation. Cognitive Science 29(1), 103–134 (2005)
4. Peraldi, S.E., Kaya, A., Melzer, S., Möller, R., Wessel, M.: Multimedia Interpretation as Abduction. In: Int'l Workshop on Description Logics (2007)
5. Fellbaum, C. (ed.): WordNet: an electronic lexical database. MIT Press (1998)
6. Ruppenhofer, J., Ellsworth, M., Petruck, M., Johnson, C., Scheffczyk, J.: FrameNet II: Extended Theory and Practice. Technical report, Berkeley, USA (2010)
7. Chambers, N., Jurafsky, D.: Unsupervised Learning of Narrative Schemas and their Participants. In: ACL, pp. 602–610 (2009)
8. Schoenmackers, S., Davis, J., Etzioni, O., Weld, D.: Learning First-order Horn Clauses from Web Text. In: EMNLP, pp. 1088–1098 (2010)
9. Hovy, D., Zhang, C., Hovy, E., Penas, A.: Unsupervised discovery of domain-specific knowledge from text. In: ACL, pp. 1466–1475 (2011)
10. Ovchinnikova, E., Montazeri, N., Alexandrov, T., Hobbs, J.R., McCord, M., Mulkar-Mehta, R.: Abductive Reasoning with a Large Knowledge Base for Discourse Processing. In: IWCS, Oxford, UK, pp. 225–234 (2011)
11. Dagan, I., Dolan, B., Magnini, B., Roth, D.: Recognizing textual entailment: Rational, evaluation and approaches - Erratum. NLE 16(1), 105 (2010)
12. Bylander, T., Allemang, D., Tanner, M.C., Josephson, J.R.: The computational complexity of abduction. Artificial Intelligence 49(1-3), 25–60 (1991)
13. Mulkar, R., Hobbs, J., Hovy, E.: Learning from Reading Syntactically Complex Biology Texts. In: Commonsense, Palo Alto (2007)

14. Poole, D.: Logic Programming, Abduction and Probability: a top-down anytime algorithm for estimating prior and posterior probabilities. New Generation Computing 11(3-4), 377–400 (1993)
15. Ishizuka, M., Matsuo, Y.: SL Method for Computing a Near-optimal Solution using Linear and Non-linear Programming in Cost-based Hypothetical Reasoning. In: PRCAI, pp. 611–625 (1998)
16. Chivers, S.T., Tagliarini, G.A., Abdelbar, A.M.: An Evolutionary Optimization Approach to Cost-Based Abduction, with Comparison to PSO. In: IJCNN, pp. 2926–2930 (2007)
17. Santos, E.: A linear constraint satisfaction approach to cost-based abduction. Artificial Intelligence 65 (1), 1–27 (1994)
18. Inoue, N., Inui, K.: ILP-Based Reasoning for Weighted Abduction. In: AAAI Workshop on Plan, Activity and Intent Recognition (2011)
19. Raina, R., Ng, A.Y., Manning, C.D.: Robust textual inference via learning and abductive reasoning. In: AAAI (2005)
20. Stern, A., Dagan, I.: A confidence model for syntactically-motivated entailment proofs. In: ACL, pp. 455–462 (2011)
21. Dantzig, G.B., Fulkerson, R., Johnson, S.M.: Solution of a large-scale traveling salesman problem. Operations Research 2(4), 393–410 (1954)
22. Riedel, S., Clarke, J.: Incremental Integer Linear Programming for Non-projective Dependency Parsing. In: EMNLP, pp. 129–137 (2006)
23. Riedel, S.: Improving the Accuracy and Efficiency of MAP Inference for Markov Logic. In: UAI, pp. 468–475 (2008)
24. Joachims, T., Finley, T., Yu, C.J.: Cutting-plane training of structural svms. In: Machine Learning, pp. 27–59 (2009)
25. Berant, J., Aviv, T., Goldberger, J.: Global Learning of Typed Entailment Rules. In: ACL, pp. 610–619 (2008)
26. Kate, R.J., Mooney, R.J.: Probabilistic Abduction using Markov Logic Networks. In: PAIRS (2009)
27. Raghavan, S., Mooney, R.J.: Bayesian Abductive Logic Programs. In: STARAI, pp. 82–87 (2010)
28. Singla, P., Domingos, P.: Abductive Markov Logic for Plan Recognition. In: AAAI, pp. 1069–1075 (2011)
29. Blythe, J., Hobbs, J.R., Domingos, P., Kate, R.J., Mooney, R.J.: Implementing Weighted Abduction in Markov Logic. In: IWCS, Oxford, UK, pp. 55–64 (2011)
30. Richardson, M., Domingos, P.: Markov logic networks. In: ML, pp. 107–136 (2006)
31. Ng, H.T., Mooney, R.J.: Abductive Plan Recognition and Diagnosis: A Comprehensive Empirical Evaluation. In: KR, pp. 499–508 (1992)
32. Hobbs, J.R.: Ontological promiscuity. In: ACL, Chicago, Illinois, pp. 61–69 (1985)
33. Bos, J.: Wide-Coverage Semantic Analysis with Boxer. In: STEP, pp. 277–286 (2008)
34. Prendinger, H., Ishizuka, M.: First-Order Diagnosis by Propositional Reasoning: A Representation-Based Approach. In: DX, pp. 220–225 (1999)
35. Abdelbar, A.M., Hefny, M.: An efficient lp-based admissible heuristic for cost-based abduction. JETAI 17(3), 297–303 (2005)

Belief Base Change Operations for Answer Set Programming

Patrick Krümpelmann and Gabriele Kern-Isberner

Technische Universität Dortmund, Germany

Abstract. We present a principled approach to the problem of belief revision in (non-monotonic) logic programming under the answer set semantics. Unlike previous approaches we use a belief base approach. Belief bases are sets of sentences that are, in contrast to belief sets, not deductively closed. We show that many of the classic base revision postulates are applicable to the logic programming case. We discuss further postulates for logic program revision and show that many of them follow from classical base revision postulates. For those postulates that do not follow from base revision postulates we propose new postulates that may also be justified from the base revision perspective. Moreover we develop a new construction for prioritized multiple base revision based on a consolidation operation via remainder sets. This construction is applicable in both the classical propositional and the logic programming cases. We connect postulates and construction by proving a representation theorem showing that the construction is exactly characterized by the proposed set of postulates.

1 Introduction

Belief dynamics within classical logics have been studied for over 25 years now [1]. The theory of classic belief revision has well formulated notions for theory change. The notions of the objects of change are well defined. In general these can be belief sets or belief bases, i. e. sets of classical propositional sentences which are deductively closed or not, respectively. For both, well defined sets of rationality postulates for different change operations have been defined. For logic programs most approaches to dynamics are very pragmatic and lack a formal representation of requirements of the change process. Few works examined the formal properties of approaches to change logic programs. Most of these use the classic AGM postulates for the revision of belief sets for logic programs. The adequate definition of a belief set, a consequence relation and a notion of equality is crucial but not at all trivial for the applicability of the classic postulates. First attempts found that "the majority of the adapted AGM and update postulates are violated ..." for a variety of approaches based on the causal rejection principle [2]. Hereby logic programs were interpreted as epistemic states, and a belief set was defined to be the set of rules satisfied by all answer sets of the program. A successful approach of a relation to the AGM theory was achieved by the use of monotonic SE-Models first presented in [3]. They made use of the semantic characterization of programs via SE-models and applied an adapted version of distance based revision operators from classic belief revision. This approach was shown to satisfy the majority of the adopted AGM postulates. AGM style revision operations for answer set programming (ASP) have disadvantages and show undesired results from the ASP point of view as first noted in [4]. These undesired results for change operations in ASP are due to the application of a semantic approach to ASP.

In this work we approach change operations of ASP from the belief base perspective. We focus on the revision operation and base it on a consolidation operation which can be used to specify other types of change operations as well. The well developed classical base revision approach has not been considered in the light of ASP before. Indeed, we argue that the belief base approach is the natural one for ASP. AGM change operations on belief sets can be seen as operations on the knowledge level, abstractly describing how an ideal reasoner would change its beliefs. This underlies the assumption of an omnipotent reasoner while ASP's main features are efficient computation of finite programs with finite answer sets. Deductive closure is defined neither for programs nor for answer sets. Belief bases are also more expressive; since on the knowledge level one cannot distinguish between inferred beliefs and fundamental, or self-supporting, ones. While this abstraction from the fundamental beliefs and their syntactic representation has advantages for the global picture of belief change we argue that ASP is primarily a syntax based approach. A key feature of ASP is that beliefs are formulated in form of easily understandable rules that allow for explicit exceptions and the explanation of inferences. From the base revision perspective the result of a change operation for ASP should be founded, understandable and close to the original syntax. In this work we present a general exploration of the application of classic base revision theory to change operations on ASP. We discuss ASP specific postulates from the literature in the light of a base revision approach, proofs of the relationships among both and formulation of adapted postulates. Finally, we develop a base revision construction which is applicable to ASP and prove a characterization theorem for it.

The remainder of this paper is structured as follows. In the next two sections we give some preliminaries on belief base revision and answer set programming. Following on this we develop base revision postulates for ASP, discuss them and relate them to other postulates for ASP belief change. After that we present our construction of multiple base revision and show its applicability to ASP and the correspondence to the postulates. Finally we discuss our approach and conclude.

2 Belief Base Revision

The classic theory of belief revision is formulated for a classical propositional language $\mathcal{L}_{prop}$. A *belief base* $\mathcal{B} \subseteq \mathcal{L}_{prop}$ is a set of classical sentences and a *belief set* $\mathcal{BS}$ is a deductively closed set of sentences. The theory of belief base change operations has been intensively studied in [5]. Postulates have been established as well as constructions and representation theorems connecting both. Let $*$ be a base revision operator which given a belief base $\mathcal{B}$ and a sentence $\alpha \in \mathcal{L}_{prop}$ returns the revised belief base $\mathcal{B}*\alpha$. Moreover, an expansion operator $+$ is defined for belief bases as the *non-closing expansion operator* defined as $\mathcal{B}+\alpha = \mathcal{B}\cup\{\alpha\}$. The basic set of postulates demanded for a belief base revision operator is the following:

Success: $\alpha \in \mathcal{B} * \alpha$
Inclusion: $\mathcal{B} * \alpha \subseteq \mathcal{B} + \alpha$
Vacuity: If $\mathcal{B} \cup \alpha$ is consistent, then $\mathcal{B} + \alpha \subseteq \mathcal{B} * \alpha$
Consistency: If α is consistent, then $\mathcal{B} * \alpha$ is consistent
Relevance: If $\beta \in (\mathcal{B} \cup \alpha) \setminus (\mathcal{B} * \alpha)$, then there is a set H such that $\mathcal{B} * \alpha \subseteq H \subseteq \mathcal{B} \cup \alpha$ and H is consistent but $H \cup \{\beta\}$ is inconsistent
Uniformity: If for all $\mathcal{B}' \subseteq \mathcal{B}, \mathcal{B}' \cup \alpha$ is inconsistent iff $\mathcal{B}' \cup \beta$ is inconsistent, then $\mathcal{B} \cap (\mathcal{B} * \alpha) = \mathcal{B} \cap (\mathcal{B} * \beta)$

The success postulate states that the new information should be part of the revision result. Inclusion demands that revision by some information should not introduce more information than expansion. Vacuity demands that if the new information is consistent with the belief base then no information should be discarded. Consistency postulates that if the new information is consistent in itself then the result of the revision is consistent as well. Relevance states that any discarded piece of information would have lead to inconsistency in a super set of the revision result. Finally, uniformity says that the effect of revision on $\mathcal{B}$ is basically determined by subsets which are inconsistent with the new information. Besides expansion and revision, contraction – as an operation removing information from the belief base is the third change operation considered in classical scenarios. It is linked to belief base revision via the Levi-Identity: $\mathcal{B} * \alpha = (\mathcal{B} - \neg\alpha) + \alpha$. A standard construction is to define a contraction $\mathcal{B} - \alpha$ as the intersection of a selection of maximal sets not containing α. Formally for a given belief base $\mathcal{B}$ and a sentence α the set of *remainder sets* denoted by $\mathcal{B} \perp \alpha$ is such that for each $X \in \mathcal{B} \perp \alpha$: $X \subseteq \mathcal{B}$, $\alpha \notin Cn(X)$ and there is no X' such that $X \subset X' \subseteq \mathcal{B}$ and $\alpha \notin Cn(X')$. Let $\mathcal{B}$ be a set of sentences. A function γ is a *selection function* for $\mathcal{B}$ iff for all sentences α if $\mathcal{B} \perp \alpha \neq \emptyset$ then $\emptyset \neq \gamma(\mathcal{B} \perp \alpha) \subseteq \mathcal{B} \perp \alpha$ and if $\mathcal{B} \perp \alpha = \emptyset$ then $\gamma(\mathcal{B} \perp \alpha) = \{\mathcal{B}\}$. The *partial meet contraction* operator is then defined as $\mathcal{B} -_\gamma \alpha = \bigcap \gamma(\mathcal{B} \perp \alpha)$. It has been shown that a revision operator satisfies all of the above postulates iff it is constructible via the Levi-identity and a partial meet contraction operator.

3 Answer Set Programming

In this work we focus on extended logic programs under the answer set semantics based on [6]. Most results generalize to generalized disjunctive logic programs and variations of semantics which will be considered in future work. Extended logic programs consist of rules over a set of propositional atoms $\mathcal{A}$ using strong negation $\neg$ and default negation *not.* A literal L can be an atom

A or a negated atom $\neg A$. The complement of a literal L is denoted by $\neg L$ and is A iff $L = \neg A$ and $\neg A$ iff $L = A$. Let $\mathcal{A}$ be the set of all atoms and Lit the set of all literals $Lit = \mathcal{A} \cup \{\neg A \mid A \in \mathcal{A}\}$. $\mathcal{D} = \{not\, L \mid L \in Lit\}$ denotes the set of all default negated literals which we call *assumptions* in the following. $\mathfrak{L} = Lit \cup \mathcal{D}$ represents the set of all literals and assumptions. A rule r is written as $L \leftarrow L_0, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n$. where the head of the rule $H(r) = L$ is either empty or consists of a single literal and the body $\mathcal{B}(r) = \{L_0, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n\}$ is a subset of $\mathfrak{L}$. The body consists of a set of literals $\mathcal{B}(r)^+ = \{L_0, \ldots, L_m\}$ and a set of assumptions denoted by $\mathcal{B}(r)^- = \{L_{m+1}, \ldots, L_n\}$. If $\mathcal{B}(r) = \emptyset$, r is a fact and if $H(r) = \emptyset$, r is a constraint. A program without default negation is a *strict* program. Let the set of constructible programs over $\mathcal{A}$ be denoted by $\mathcal{P}_{\mathcal{A}}$.

A set of literals which is consistent, i. e. it does not contain complementary literals L and $\neg L$, is called a state I. A literal L is true in I iff $L \in I$ and false otherwise. The body $\mathcal{B}(r)$ of a given rule r is true in I iff each $L \in \mathcal{B}(r)^+$ is true in I and each $L \in \mathcal{B}(r)^-$ is false in I. A rule r is true in I iff $H(r)$ is true in I whenever $\mathcal{B}(r)$ is true in I. A state is a model of a program P iff r is true in I for all $r \in P$. The reduct P^S of a program P relative to a set S of literals is defined as: $P^S = \{H(r) \leftarrow \mathcal{B}^+(r) \mid r \in P, \mathcal{B}^-(r) \cap S = \emptyset\}$. An answer set of a program P is a state I which is a minimal model of its reduct P^I. The set of answer sets of a program P is denoted by $AS(P)$. A program P is *consistent* iff $AS(P) \neq \emptyset$. For some fixed set of atoms $\mathcal{A}$ two programs $P \in \mathcal{P}_{\mathcal{A}}$ and $P' \in \mathcal{P}_{\mathcal{A}}$ are *uniformly equivalent* iff for all sets of facts $F \in \mathcal{P}_{\mathcal{A}}$, $AS(P \cup F) = AS(P' \cup F)$, *strongly equivalent* iff F can be any program and *ordinarily equivalent* iff $F = \emptyset$.

4 Postulates for ASP Base Revision

We consider a *multiple base revision operator* $* : \mathcal{P}_{\mathcal{A}} \times \mathcal{P}_{\mathcal{A}} \to \mathcal{P}_{\mathcal{A}}$ for ASP. Revision aims at solving conflicts between prior and posterior beliefs that can be discovered by inconsistencies. In the context of logic programs a notable property of inconsistency is that the state of inconsistency of a program can change non-monotonically. While in the classical case any subset of a consistent set of sentences is consistent, for a consistent logic program Q there can be a subset $P \subset Q$ such that P is inconsistent.

Example 1. The program $P = \{a., \neg a \leftarrow not\ b.\}$ is obviously inconsistent. $P' = P \cup \{b \leftarrow not\ c.\}$ is consistent with $AS(P') = \{\{a, b\}\}$ while $P'' = P' \cup \{c.\}$ is inconsistent again.

The base revision postulates given above can be directly translated to the logic programming case with $+$ being the non-closing expansion $P + Q = P \cup Q$. For $*$ being a multiple base revision operator for logic programs we define the following postulates:

Success: $Q \subseteq P * Q$
Inclusion: $P * Q \subseteq P + Q$
Vacuity: If $P + Q$ is consistent, then $P + Q \subseteq P * Q$

Consistency: If Q is consistent, then $P * Q$ is consistent.

Relevance: If $r \in (P \cup Q) \setminus (P * Q)$, then there is a program H such that $P * Q \subseteq H \subseteq P \cup Q$ and H is consistent but $H \cup \{r\}$ is inconsistent.

Fullness: If $r \in (P \cup Q) \setminus (P * Q)$, then $P * Q$ is consistent and $(P * Q) \cup \{r\}$ is inconsistent.

Uniformity: If for all $P' \subseteq P$, $P' \cup \{Q\}$ is inconsistent iff $P' \cup \{R\}$ is inconsistent, then $P \cap (P * Q) = P \cap (P * R)$

The postulate of *Fullness* is a stronger version of *Relevance* and resembles a stronger requirement to the minimality of change. For the classical case *Fullness* is arguably too strong [5], however for weaker logics it has also been proven to be useful, cf. [7]. For logic programing it also does not have the same undesirable implications as we shall see later on.

Lemma 1. *If* $*$ *satisfies* Fullness*, then it satisfies* Relevance*. If* $*$ *satisfies* Relevance*, then it satisfies* Vacuity*.*[1]

Due to the non-monotonicity of inconsistency we can strengthen the *Consistency* postulate for the logic programming case such that they capture the same idea as for the classical case. The *Consistency* postulate expresses in the classical case that the outcome of the revision shall be consistent whenever possible. For classical propositional logic this is possible iff the input is consistent. Due to the *Success* postulate and the monotonicity of the state of inconsistency any revision with an inconsistent input is inconsistent. If the input is consistent, then rejecting all previous information in any case leads to a consistent belief base. In logic programming, however, the input can be inconsistent, the *Success* postulate satisfied and yet the revision outcome can be consistent. This is shown by the following example.

Example 2. Let $P = \{b., \neg a.\}$ and $Q = \{a., \neg a \leftarrow \mathit{not}\ b.\}$. The program Q is inconsistent but the revision $P * Q = \{b., a., \neg a \leftarrow \mathit{not}\ b.\}$ is consistent and satisfies all other postulates.

Hence we strengthen the *Consistency* postulate to adequately capture non-monotonic inconsistency by use of an appropriate premise for the possibility of consistency.

NM-Consistency: If there exists some consistent X, $Q \subseteq X \subseteq P \cup Q$, then $P * Q$ is consistent.

Proposition 1. *If a revision operator* $*$ *satisfies* NM-Consistency*, then it satisfies* Consistency*.*

The base revision postulates have been accepted as characterizations of desirable revision operations for classical belief bases. The postulates can be applied to belief bases represented as logic programs as just shown. However, it has to be shown that the defined postulates for belief base revision lead to desirable

[1] Due to the page limit we do not include proofs in this paper.

inference behavior given the non-monotonicity of the answer set semantics. Further postulates for the connection of the revision on the program level and the resulting answer sets might have to be formulated. Such ASP specific postulates for belief dynamics have been proposed in [2] and adopted by several authors for the evaluation of their approaches afterwards [3,8,9]. Those postulates base on the notion of ordinary equivalence based on the identity of the sets of answer sets (AS) in [2][2], on strong equivalence (SE) in [3] and partially on uniform equivalence (UE) in [8]. Here, we generalize the postulates by considering different notions of equivalence. Therefore we parameterize them with a notion of equivalence based on $\circ$. We obtain the original postulates with $\circ = AS$ but consider a family of equivalences $\circ \in \{AS, UE, SE, P\}$; with $\equiv_P$ being the syntactic identity of the programs, i. e. $P \equiv_P P'$ iff $P = P'$. The equivalences are increasingly stronger with the order given above. More precisely, it holds that for any two programs P and P' that if $P \equiv_P P'$ then $P \equiv_{SE} P'$, if $P \equiv_{SE} P'$ then $P \equiv_{UE} P'$ and if $P \equiv_{UE} P'$ then $P \equiv_{AS} P'$. Thus apart from the original postulates we also consider stronger versions of these. It should be noted that this family of notions of equivalence could be extended by all intermediate notions as formalized in [10]. For one of the postulates, namely the *Tautology* postulate, we need to define the notion of tautological programs.

Definition 1. *Let P be a program over the set of literals Lit. P is* tautological*, denoted by $P_\top$, iff for each rule $r \in P$, r is true in all states $I \subseteq Lit$.*

The generalized postulates for ASP revision from [2] are:

Initialisation$_\circ$: $\emptyset * P \equiv_\circ P$
Idempotence$_\circ$: $P * P \equiv_\circ P$
Absorption$_\circ$: $(P * Q) * Q \equiv_\circ P * Q$
Tautology$_\circ$: $P * P_\top \equiv_\circ P$
Disjointness$_\circ$: If $P = P_1 \cup P_2$ and P_1 and P_2 have disjoint sets of literals, then $P * Q \equiv_\circ (P_1 * Q) \cup (P_2 * Q)$.
Parallelism$_\circ$: If Q_1 and Q_2 have disjoint sets of literals, then $P * (Q_1 \cup Q_2) \equiv_\circ (P * Q_1) \cup (P * Q_2)$.

The implications on the equivalences given above lead to implications of the resulting postulates. Most importantly we get that if an operator $*$ satisfies a postulate for some notion of equivalence, then it also satisfies the variants of the postulates for all weaker notions of equivalence, e. g., *Initialisation$_P$* implies *Initialisation$_{SE}$*, *Initialisation$_{UE}$* and *Initialisation$_{AS}$*. On the other hand, if $*$ is shown to violate *Initialisation$_{AS}$*, then it also violates *Initialisation$_{UE}$*, *Initialisation$_{SE}$* and *Initialisation$_P$*. The same holds for all other postulates. In the following we only show results for the strongest version of a postulate and omit the implications of them. We obtain the following results for the connection of base revision postulates and ASP change postulates:

Proposition 2. *Let $*$ be a revision operator on logic programs.*

[2] For finite alphabets.

1. *If $*$ satisfies Success and Inclusion, then it satisfies Initialisation$_P$.*
2. *If $*$ satisfies Success and Inclusion, then it satisfies Idempotence$_P$.*
3. *If $*$ satisfies Success, Consistency, Inclusion and Vacuity, then it satisfies Absorption$_P$.*

Hence, we have just shown that the first three ASP postulates follow for all considered notions of equivalence from very basic base revision postulates. This verifies the adequateness of the base revision approach for ASP. The remaining four ASP postulates do not follow from the base revision postulates and are in conflict with some of them. This might be due to the fact that they were formulated for approaches for handling update sequences of logic programs which were shown to "neither have update nor revision flavor" in the classical sense [2]. However, the underlying ideas of these postulates are useful and can be adapted to the base revision setting, as we show in the following.

The *Tautology$_{AS}$* postulate is violated if the belief base is inconsistent before and consistent after revising by a tautology.

Example 3. Let $P = \{a., \neg a.\}$ and $Q = \{b \leftarrow b.\}$ with $AS(P) = \emptyset$ and $AS(Q) = \{\emptyset\}$. For any revision operator $*$ satisfying Tautology$_{AS}$ we get $AS(P * Q) = \emptyset$, e.g. $P * Q = \{a., \neg a., b \leftarrow b.\}$. Consistency on the other hand allows for changes to make the belief base consistent. For any revision operator $*'$ satisfying Consistency we get $AS(P *' Q) \neq \emptyset$. In this case $P *' Q = \{a., b \leftarrow b.\}$ or $P *' Q = \{\neg a., b \leftarrow b.\}$.

Thus *Tautology* cannot be satisfied by any operator satisfying consistency because of the case in which an inconsistent belief base is made consistent by the revision by a tautology.

Proposition 3. *Let $*$ be a revision operator on logic programs.*

1. *If $*$ satisfies Consistency, then it violates Tautology$_{AS}$.*
2. *If $*$ satisfies Vacuity, then it violates Tautology$_P$.*

Tautology in general addresses an important issue in ASP revision, namely that if the belief base is consistent, the revision by a tautology should not lead to any changes of the semantics. This is not satisfied by many approaches to dynamics in ASP. To adapt the *Tautology$_{AS}$* postulate to the base revision setting it is desirable that *Tautology$_{AS}$* is satisfied, except for the case in which the belief base is inconsistent. To this end we introduce the following weakening of *Tautology.*

Consistent Tautology$_\circ$: If P is consistent, then $P * P_\top \equiv_\circ P$.

The problem of the approaches not satisfying the *Tautology* postulate is that they make unnecessary changes not only for tautological revisions. Any revision by a program that has no semantic influence should not add any answer sets. This idea has been discussed in [11] and formalized for dynamic logic programming as the *refined extension principle.* Here we formalize this idea in the following postulate for all $\circ$ equivalences:

Consistent Irrelevance$_\circ$: If P is consistent and $P \equiv_\circ P \cup Q$, then $P * Q \equiv_\circ P$.

Clearly *Consistent Tautology*$_\circ$ follows from *Consistent Irrelevance*$_\circ$ iff for all $P \in \mathcal{P}_\mathcal{A}$, $P \cup P_\top \equiv_\circ P$. This holds for $\equiv_{AS}, \equiv_{UE}, \equiv_{SE}$ but not for $\equiv_P$.

Proposition 4. *If $*$ satisfies Consistent Irrelevance$_\circ$ and $\circ \in \{AS, UE, SE\}$, then $*$ satisfies Consistent Tautology$_\circ$.*

Consistent Irrelevance$_\circ$ is a postulate formulating some form of minimal change on the semantical level. Two basic postulates for minimal change of the base revision postulates, *Inclusion* and *Vacuity*, are sufficient to guarantee *Consistent Irrelevance*$_\circ$.

Proposition 5. *If $*$ satisfies Inclusion and Vacuity, then it satisfies Consistent Irrelevance$_\circ$ for $\circ \in \{AS, UE, SE, P\}$.*

Corollary 1. *If $*$ satisfies Inclusion and Vacuity, then it satisfies Consistent Tautology$_\circ$ for $\circ \in \{AS, UE, SE\}$.*

These results are consistent with Proposition 3 which includes a negative result for the $\circ = P$ case of Corollary 1. We consider *Parallelism*$_\circ$ and *Disjointness*$_\circ$ together since the underlying idea is similar. The problem with *Parallelism*$_\circ$ and *Disjointness*$_\circ$ is that the respective third program Q resp. P can contain rules connecting the disjoint sets of literals such that inconsistencies arise in combination of both sets of literals but not based on a single one.

Proposition 6. *Let $*$ be a revision operator on logic programs.*

1. *If $*$ satisfies Success, Consistency and Vacuity then it violates Parallelism$_{AS}$.*
2. *If $*$ satisfies Vacuity and Consistency then it violates Disjointness$_{AS}$.*

Consider the following example which demonstrates that *Disjointness*$_{AS}$ is in conflict with the principle of minimal change in base revision.

Example 4. Let $P = \{a., b.\}$ such that $P = P_1 \cup P_2$ with $P_1 = \{a.\}$ and $P_2 = \{b.\}$, and $Q = \{\neg a \leftarrow b.\}$. For the revision of $P_1 * Q$ we can note that there is no conflict and $AS(P_1 \cup Q) = \{\{a.\}\}$ such that there is no need to change anything, such that $P_1 * Q = P_1 \cup Q$ seems to be a reasonable revision. The same holds for $P_2 * Q$ with $AS(P_2 \cup Q) = \{\{b.\}\}$. The Disjointness$_{AS}$ postulate demands that $AS(P * Q) = AS((P_1 * Q) \cup (P_2 * Q))$. In this case $AS((P_1 * Q) \cup (P_2 * Q)) = \emptyset$ which is clearly not a desirable outcome for $AS(P * Q)$.

Disjointness$_{AS}$ is in conflict with the minimal change of the revisions of $P_1 * Q$ and $P_2 * Q$. In order to satisfy *Disjointness*$_{AS}$ the revisions of $P_1 * Q$ and $P_2 * Q$ have to be cautious enough to anticipate possible inconsistencies with additional input. Here we consider that inconsistencies with some input should be handled by a revision operator applied to this input and not by a previous revision.

Hence these postulates are too strong to be satisfiable by a base revision approach. Therefore we present weakened versions of the postulates that allow for minimal change operations.

Weak Disjointness$_\circ$: If $P = P_1 \cup P_2$ and P_1 and P_2 have disjoint sets of literals $\mathcal{A}_1$ and $\mathcal{A}_2$ and for each set of literals $\mathcal{A}_r$ of a rule $r \in Q$ it holds $\mathcal{A}_r \cap \mathcal{A}_1 = \emptyset$ or $\mathcal{A}_r \cap \mathcal{A}_2 = \emptyset$, then $P * Q \equiv_\circ (P_1 * Q) \cup (P_2 * Q)$.

Weak Parallelism$_\circ$: If Q_1 and Q_2 have disjoint sets of literals $\mathcal{A}_1$ and $\mathcal{A}_2$, and for each set of literals $\mathcal{A}_r$ of a rule $r \in P$ it holds $\mathcal{A}_r \cap \mathcal{A}_1 = \emptyset$ or $\mathcal{A}_r \cap \mathcal{A}_2 = \emptyset$, then $P * (Q_1 \cup Q_2) \equiv_\circ (P * Q_1) \cup (P * Q_2)$.

These weakened versions of *Disjointness*$_\circ$ and *Parallelism*$_\circ$ are not in conflict with the proposed set of base revision postulates, but are still strong enough such that they do not follow from the base revision postulate set. Thus we add them to the set of desirable postulates.

To sum up, we have shown that all postulates for ASP revision that are not in conflict with the base revision setting follow from the base revision postulates. For those ASP revision postulates that are in conflict with the base revision postulates we gave adequately weakened versions. Therefore we are looking for an operator which satisfies *Success, Inclusion, NM-Consistency, Fullness, Uniformity, Weak Disjointness*$_P$ and *Weak Parallelism*$_P$. As shown above, such an operator also satisfies *Vacuity, Consistency, Relevance, Initialisation*$_\circ$*, Idempotence*$_\circ$*, Absorption*$_\circ$*, Consistent Irrelevance*$_\circ$ for $\circ \in \{AS, UE, SE, P\}$, and *Consistent Tautology*$_\circ$ *for* $\circ \in \{AS, UE, SE\}$.

5 Construction of ASP Base Revision

The direct transfer of the construction of belief base operators via the Levi identity to logic programs does not work, because neither the negation nor the inference of a rule is defined and inconsistency cannot be reduced to complementary literals. Even in the very restricted case of a revision of some program P by new information $Q = \{L.\}$ consisting of a single fact, it would not be sufficient to contract such that $\neg L \notin \cap AS(P)$.

So we have to look for different constructions which implement the idea of the Levi-Identity while being adequate for the ASP case. The idea of the Levi-Identity is, that after contracting by $\neg\alpha$ the belief base is consistent with α. That is, the belief base is made consistent with the information to be added before adding it. This does not work in general for the logic programming case because the dependencies within a program are complex and cannot be anticipated without including the input program. The inconsistency of a program P with a new program Q can only be determined by considering $P \cup Q$, as the interaction of rules of both programs generates the inconsistency. Base revision constructions of this type are called *external revision* since a sub-operation takes place outside of the original set. Hence the base revision construction for logic programs has to consider $P \cup Q$ to determine inconsistency. From $P \cup Q$ rules are removed such that the resulting program is consistent with Q. This idea amounts to the consolidation of $P \cup Q$ under certain constraints. In the base revision literature the unary operator ! is called a consolidation operator and results in a consistent subset of the input. A consolidation operator has been used in [12] to define *semi-revision*, which is defined as $\mathcal{B} *_? \alpha = (\mathcal{B} \cup \alpha)!$. The problem with semi-revision for our means is that it is non-prioritized, i. e. the success postulate is not satisfied in general.

We extend the idea of the semi-revision construction to be able to define a prioritized revision operator for logic programs. To this end we define a screened consolidation operator $!_R$ with R being a set of core sentences that are immune to change like in screened revision [13]. We propose the following set of postulates for a screened consolidation:

C-Screen: $R \subseteq P!_R$.
C-Screen-Consistency: If there exists some consistent X, $R \subseteq X \subseteq P$ then $P\,!_R$ is consistent.
C-Inclusion: $P\,!_R \subseteq P$
C-Relevance: If $r \in P$ and $r \notin P!_R$, then there is a set P' such that $P\,!_R \subseteq P' \subseteq P$ and that P' is consistent but $P' \cup \{r\}$ is inconsistent.
C-Fullness: If $r \in P$ and $r \notin P!_R$, then $P!_R$ is consistent and $P!_R \cup \{r\}$ is inconsistent.
C-Screen-Uniformity: Let R, R' and P be sets of rules and R and R' be consistent. If for all $X \subseteq P$, $R \cup X$ is consistent iff $R' \cup X$ is consistent, then $P \cap (P \cup R)!_R = P \cap (P \cup R')!_{R'}$.

Note that as in the classic case satisfaction of *C-Fullness* implies satisfaction of *C-Relevance.*

Proposition 7. *Let $!_R$ be a screened consolidation operator. If $!_R$ satisfies $C-Fullness$, then it satisfies $C-Relevance$.*

Definition 2 (Screened Consolidation). *An operator $!_R$ is an operator of screened consolidation iff it satisfies C-Screen, C-Screen-Consistency, C-Inclusion, C-Relevance, C-Fullness and C-Screen-Uniformity.*

As stated before, the new postulates of *Weak-Disjointness*$_\circ$ and *Weak-Parallelism*$_\circ$ do not follow from the basic set of belief base postulates. This is also true for the postulate set for screened consolidation presented here. For their satisfaction we introduce the notion of topic independence as defined in [12], which bases on topicalizations.

Definition 3. *[12] Let P be a set and $\mathfrak{P}(P)$ the powerset of P. A set $\mathfrak{B} \subseteq \mathfrak{P}(P)$ is a* topicalization *of P iff: $P = \bigcup \mathfrak{B}$ and if $R \subseteq P$, then R is consistent if $R \cap B$ is consistent for all $B \in \mathfrak{B}$*

A topicalization $\mathcal{B}$ of P is hence a cover of P for which it holds that the consistency of each subset of P is only dependent on the consistency of its projection for each topic $B \in \mathcal{B}$.

C-Topic-Independence: [12] If $\mathfrak{B}$ is a topicalization of P, then $P \setminus P!_R = \bigcup_{B \in \mathcal{B}} (B \setminus B!_R)$

The postulate of *C-Topic-Independence* demands that given a topicalization of P each rule that has been removed from P by the consolidation operation $!_R$ is also removed from each topic $B \in \mathcal{B}$ with $r \in B$ by the respective consolidation of the topic and each rule removed by all consolidations of topics of P containing it is removed from P.

Definition 4. *Given a screened consolidation operator* $!_R$ *we define a* multiple ASP base revision operator *by setting:* $P * Q = (P \cup Q)!_Q$

Proposition 8. *Let* $*$ *be a multiple base revision operator defined as* $P * Q = (P \cup Q)!_Q$. *If* $!_R$ *is a screened consolidation operator that satisfies C-Topic-Independence, then* $*$ *satisfies Success, Inclusion, Vacuity, Consistency, NM-Consistency, Relevance, Fullness, Uniformity, Weak-Disjointness*$_P$ *and Weak-Parallelism*$_P$.

We based the revision operation on a consolidation operation and characterized the latter by a set of postulates. We need a construction of a screened consolidation operator that is suitable for the application to logic programs. Two constructions of consolidation operations are available from belief base theory: partial meet and kernel consolidation [12]. We use a partial meet construction here. We start by defining a screened version of remainder sets for logic programs in which all remainder sets contain the screened set of rules.

Definition 5 (Screened Remainder Sets). *For sets of sentences* P *and* R *with* $R \subseteq P$ *the set of* screened consistent remainder sets *of* P, *denoted by* $P \perp_! R$ *is such that for each* $X \in P \perp_! R$: *1)* $R \subseteq X \subseteq P$, *2)* X *is consistent, and 3) there is no* X' *such that* $X \subset X' \subseteq P$ *and* X' *is consistent*

The definition of screened remainder sets differs from the original formulation only in the first condition which is $X \subseteq P$ originally. In contrast to the classical case, the intersection of remainders of logic programs is not necessarily consistent due to the non-monotonicity of logic programs.

Example 5. Let $P = \{a \leftarrow b., \neg a., b., \leftarrow \mathit{not}\ \neg a, \mathit{not}\ b.\}$. The set of remainder sets with the empty screen are $P \perp_! \emptyset = \{\{a \leftarrow b., b., \leftarrow \mathit{not}\ \neg a, \mathit{not}\ b.\}, \{\neg a., b., \leftarrow \mathit{not}\ \neg a, \mathit{not}\ b.\}, \{a \leftarrow b., \neg a., \leftarrow \mathit{not}\ \neg a, \mathit{not}\ b.\}\}$. The intersection of the first two remainders, $\{b., \leftarrow \mathit{not}\ \neg a, \mathit{not}\ b.\}$, is consistent. The intersection of the first and the last remainder, $\{a \leftarrow b., \leftarrow \mathit{not}\ \neg a, \mathit{not}\ b.\}$, is inconsistent.

This leaves us with the option of selecting exactly one of the remainders which also has the advantage of performing less change.

A selection function is specific to a certain belief base since it shall evaluate to the belief base if no remainder sets exists. We obtain a global selection function by making use of a two place selection function [5] with the belief base a parameter.

Definition 6 (Global maxichoice selection function). *Let* P *be a set of sentences.* γ_P *is a* selection function *for* P *iff for all sets of sentences* R

1. *If* $P \perp_! R \neq \emptyset$, *then* $\gamma_P(P \perp_! R) = X$ *for some* $X \in P \perp_! R$.
2. *If* $P \perp_! R = \emptyset$, *then* $\gamma_P(P \perp_! R) = P$.

A global maxichoice selection function *is a function* γ *such that for each* $P \subseteq \mathcal{L}$, $\gamma(P, \cdot) = \gamma_P(\cdot)$ *is a selection function for* P. *We drop the index* P *if it is obvious.*

A global maxichoice selection function is globally defined for all belief bases. Hence, in contrast to one place selection functions, global maxichoice selection functions are not specific to one particular belief base. Moreover, properties of consolidation operators which connect the consolidation results of several dependent belief bases can only be expressed for global maxichoice selection functions. We define a screened maxichoice consolidation operation based on a global maxichoice selection function.

Definition 7. *Let P and R be sets of sentences and γ a maxichoice selection function for P. The operation $P\,!_R$ such that $P\,!_R = \gamma(P \perp_! R)$ is a* screened maxichoice consolidation *based on γ.*

The following representation theorem shows that any screened maxichoice consolidation satisfies the set of postulates for screened consolidation and, moreover, that any operation satisfying these postulates can be constructed as a screened maxichoice consolidation.

Proposition 9. *An operation $!_R$ is an operation of screened maxichoice consolidation iff it satisfies C-Inclusion, C-Screen-Consistency, C-Screen, C-Fullness and C-Uniformity.*

Finally, we link maxichoice consolidation with topic-independence by a notion of monotony for selection functions.

Definition 8. *A global selection function γ is* monotone *iff for all $P, P' \subseteq \mathcal{L}$, if for each $X \in P \perp_! \emptyset$ there exists some $X' \in P' \perp_! \emptyset$ such that $P \setminus X = P \setminus X'$, then $P \setminus \gamma(P \perp_! \emptyset) = P \setminus \gamma(P' \perp_! \emptyset)$.*

An operator of screened consolidation is monotone *iff it is based on a monotone selection function.*

Proposition 10. *If a screened maxichoice consolidation operator is based on a monotone maxichoice selection function, then it satisfies topic-independence.*

6 Discussion and Conclusion

In this work we presented a base revision description and construction for belief bases represented by extended logic programs under the answer set semantics. The majority of work on the dynamics in logic programming has focused on the implementation of inconsistency handling in sequences of logic programs via logic programs, e. g. [11,2,14]. Most of the principle based approaches used the classic AGM postulates for belief set revision as reference and developed alternative postulates as discussed previously, e. g. [2,11]. The closest work to base revision is the state transition system approach presented in [15] which was meant to satisfy base revision postulates for the revision by a fact, but no formal results are shown. In terms of the adaption of classic construction for belief revision only the distance based construction via SE-Models of [3] has been used. Approaches to belief revision in other non-monotonic formalisms are often not based on principles and very specific to the underlying logic [16,17]. A principle based approach to contraction operations in logic programs has been considered in [18]. The results presented in this paper show that the base revision approach to belief revision is applicable to revision of logic programs and formalizes adequate properties such an operation should satisfy. We defined new postulates and a construction satisfying all proposed postulates. This resembles an important foundation for base revision approaches to answer set programming.

Acknowledgements. This work has been supported by the DFG, Collaborative Research Center SFB876, Project A5. (`http://sfb876.tu-dortmund.de`)

References

1. Fermé, E., Hansson, S.: AGM 25 years. Journal of Philosophical Logic 40, 295–331 (2011), doi:10.1007/s10992-011-9171-9
2. Eiter, T., Fink, M., Sabbatini, G., Tompits, H.: On properties of update sequences based on causal rejection. Theory and Practice of Logic Programming 2(6) (2002)
3. Delgrande, J., Schaub, T., Tompits, H., Woltran, S.: Belief revision of logic programs under answer set semantics. In: Proc. of the 11th Int'l Conference on Principles of Knowledge Representation and Reasoning (KR 2008). AAAI Press (2008)
4. Slota, M., Leite, J.: On semantic update operators for answer-set programs. In: Proc. of the 19th European Conference on Artificial Intelligence, ECAI (2010)
5. Hansson, S.O.: A Textbook of Belief Dynamics. Kluwer Academic Publishers, Norwell (2001)
6. Gelfond, M., Leone, N.: Logic programming and knowledge representation — the A-Prolog perspective. Artificial Intelligence 138(1-2), 3–38 (2002)
7. Delgrande, J.P.: Horn clause belief change: Contraction functions. In: Proc. of the 11th int'l Conference on Principles of Knowledge Representation and Reasoning (KR), pp. 156–165. AAAI Press (2008)
8. Delgrande, J.P.: An approach to revising logic programs under the answer set semantics. In: 13th Int'l Workshop on Non-Monotonic Reasoning, NMR 2010 (2010)
9. Osorio, M., Cuevas, V.: Updates in answer set programming: An approach based on basic structural properties. Theory Pract. Log. Program. 7(4), 451–479 (2007)
10. Woltran, S.: A common view on strong, uniform, and other notions of equivalence in answer-set programming. Theory Pract. Log. Program. 8(2), 217–234 (2008)
11. Alferes, J.J., Banti, F., Brogi, A., Leite, J.A.: The refined extension principle for semantics of dynamic logic programming. Studia Logica 79(1) (2005)
12. Hansson, S.O.: Semi-revision. Journal of Applied Non-Classical Logics 7(2) (1997)
13. Makinson, D.: Screened revision. Theoria. 63(1-2), 14–23 (1997)
14. Krümpelmann, P.: Dependency semantics for sequences of extended logic programs. Logic Journal of the IGPL (2012), doi: 10.1093/jigpal/jzs012
15. Kudo, Y., Murai, T.: A Method of Belief Base Revision for Extended Logic Programs Based on State Transition Diagrams. n: Negoita, M.G., Howlett, R.J., Jain, L.C. (eds.) KES 2004, Part I. LNCS (LNAI), vol. 3213, pp. 1079–1084. Springer, Heidelberg (2004)
16. Witteveen, G., van der Hoek, W.: A General Framework for Revising Nonmonotonic Theories. In: Fuhrbach, U., Dix, J., Nerode, A. (eds.) LPNMR 1997. LNCS, vol. 1265, pp. 258–272. Springer, Heidelberg (1997)
17. Billington, D., Antoniou, G., Governatori, G., Maher, M.J.: Revising Nonmonotonic Theories: The Case of Defeasible Logic. In: Burgard, W., Christaller, T., Cremers, A.B. (eds.) KI 1999. LNCS (LNAI), vol. 1701, pp. 101–112. Springer, Heidelberg (1999)
18. Krümpelmann, P., Kern-Isberner, G.: On belief dynamics of dependency relations for extended logic programs. In: Proceedings of the 13th International Workshop on Non-Monotonic Reasoning, NMR 2010 (2010)

A Framework for Semantic-Based Similarity Measures for $\mathcal{ELH}$-Concepts

Karsten Lehmann[1,2] and Anni-Yasmin Turhan[3,⋆]

[1] Optimisation Research Group, NICTA
Karsten.Lehmann@nicta.com.au
[2] Artificial Intelligence Group, Australian National University
[3] Institute for Theoretical Computer, TU Dresden, Germany
turhan@tcs.inf.tu-dresden.de

Abstract. Similarity measures for concepts written in Description Logics (DLs) are often devised based on the syntax of concepts or simply by adjusting them to a set of instance data. These measures do not take the semantics of the concepts into account and can thus lead to unintuitive results. It even remains unclear how these measures behave if applied to new domains or new sets of instance data.

In this paper we develop a framework for similarity measures for $\mathcal{ELH}$-concept descriptions based on the semantics of the DL $\mathcal{ELH}$. We show that our framework ensures that the measures resulting from instantiations fulfill fundamental properties, such as equivalence invariance, yet the framework provides the flexibility to adjust measures to specifics of the modelled domain.

1 Introduction

Concept similarity measures map a pair of concepts from an ontology to a value between 0 and 1 indicating how similar the concepts are. These measures are an important means to discover similar concepts in ontologies. In bio-medical ontology-based applications, for example the Gene ontology [5], they are employed to discover functional similarities of genes. Furthermore, concept similarity measures are used in ontology alignment algorithms [9].

A common approach to find and evaluate similarity measures is to have test data and to tune a similarity measure until it matches the results of a human expert. The disadvantage of this approach is that the behavior of such a measure is hard to predict when applied to new test data, or when used for ontologies modeling a different domain. As a consequence an ontology developer cannot competently decide whether a measure obtained in this way is suitable for a particular task.

Description Logics (DLs) are a family of knowledge representation formalisms with formal semantics. A good similarity measure for DL concepts should take

⋆ Partially supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 “Highly Adaptive Energy-Efficient Computing”.

the semantics of the underlying formalism into account, instead of assessing similarity in a purely syntactical way. Similarity measures are often tailored for particular applications. Thus, one similarity measure will hardly meet the needs of all applications.

In [8] the intended behavior of a measure was discussed and partially captured in terms of properties. These properties were adapted from metric spaces which are related to similarity measures. We follow this approach to address the problems mentioned above. We extend this set of properties by including DL specific ones and mathematically describe those from [8] in terms of DL. The formalization of the properties allows us to prove whether or not an obtained measure has the desired properties. Additionally, we investigate existing DL similarity measures to determine which of the properties they fulfill. We then propose the framework *simi* for similarity measures for $\mathcal{ELH}$-concepts. If instantiated with the right functions and operators as building blocks, *simi* yields measures for which (most of) the formalized properties can be guaranteed. At the same time the framework retains flexibility as it allows users to choose from the list which properties the resulting measure should have and to build their measure accordingly. Furthermore, the resulting similarity measures can be computed efficiently, provided that functions employed can be computed efficiently as well.

Our choice for the DL $\mathcal{ELH}$ is motivated by the fact that large, well-known bio-medical ontologies such as the Gene Ontology [5] or SNOMED [21] are written in (extensions of) $\mathcal{ELH}$. Furthermore, $\mathcal{ELH}$ is a fragment of the DL that corresponds to the OWL 2 EL profile, which is part of the W3C standard for an ontology language for the Semantic Web [23, 19].

The paper is structured as follows: we start with preliminaries on DLs. In Section 3, we introduce the set of properties desirable for similarity measures and in Section 4 we devise a framework for constructing similarity measures that fulfill (most of) the introduced properties. The paper ends with conclusions and directions for future work.

2 Preliminaries

In this section we introduce the basic notions of DLs. For a thorough introduction see [1]. Starting from a finite set of concept names N_C and a finite set of role names N_R, complex concepts can be defined using *concept constructors*. Let A, $B \in N_C$, then *$\mathcal{EL}$-concepts* are formed according to the following syntax rule:

$$C ::= \top \mid A \mid C \sqcap D \mid \exists r.C$$

where $r \in N_R$ and C, D denote arbitrary $\mathcal{EL}$-concepts. A concept of the form $\exists r.C$ is called an *existential restriction* and one of the from $C \sqcap D$ is called a *conjunction*. We call the DL, that only offers conjunction as a concept constructor, $\mathcal{L}_0$. The semantics of concepts is given in terms of interpretations. An *interpretation* $\mathcal{I} = (\Delta, \cdot)$ consists of the *interpretation domain* $\Delta^{\mathcal{I}}$ a non-empty set and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns role names to binary relations on $\Delta^{\mathcal{I}}$ and concepts to subsets of $\Delta^{\mathcal{I}}$. The top-concept $\top$ is mapped to $\Delta^{\mathcal{I}}$. The extension of the interpretation function to conjunctions is $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$ and to existential restrictions $(\exists r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}}$ and $e \in C^{\mathcal{I}}\}$.

A *concept definition* assigns a concept name to a complex concept. We call $A = C$ a *concept definition* and $A \sqsubseteq C$ a *primitive concept definition.* A finite set of (possibly primitive) concept definitions is a *TBox* $\mathcal{T}$. If the (primitive) definitions in a TBox are acyclic and do not contain multiple definitions we call the TBox *unfoldable.* Concept names occurring on the left-hand side of a definition are called *defined concepts.* All other concept names are called *primitive concepts.* Let $s, r \in N_R$. A *role inclusion axiom* (RIA) is a statement of the form: $r \sqsubseteq s$. The DL that extends $\mathcal{EL}$ by RIAs is called $\mathcal{ELH}$. An interpretation is a model for $s \sqsubseteq r$ iff $s^{\mathcal{I}} \subseteq r^{\mathcal{I}}$. A finite set of RIAs is called *RBox* $\mathcal{R}$. An interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ (RBox $\mathcal{R}$) iff it satisfies all its concept definitions (RIAs). We write $s \sqsubseteq_{\mathcal{R}} r$, if $s^{\mathcal{I}} \subseteq r^{\mathcal{I}}$ holds in all models of $\mathcal{R}$ and $s \equiv_{\mathcal{R}} r$, if $s \sqsubseteq_{\mathcal{R}} r$ and $r \sqsubseteq_{\mathcal{R}} s$ hold.

A DL *knowledge base* (KB) $\mathcal{K}$ consists of the *TBox* and the *RBox* and we say that an interpretation $\mathcal{I}$ is a *model* of $\mathcal{K}$, if it is a model for the corresponding TBox and RBox.

Based on the semantics of concepts, reasoning problems can be defined. The concept C is *subsumed* by the concept D w.r.t. the KB $\mathcal{K}$ ($C \sqsubseteq_{\mathcal{K}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{K}$. C and D are *equivalent* w.r.t. $\mathcal{K}$ ($C \equiv_{\mathcal{K}} D$) iff $C \sqsubseteq_{\mathcal{K}} D$ and $D \sqsubseteq_{\mathcal{K}} C$.

For a given concept C, *expansion* replaces exhaustively all occurrences of defined concepts in C by the right-hand sides of their concept definitions. For unfoldable TBoxes all reasoning problems can be reduced to reasoning for concepts by using expansion of concepts w.r.t. the TBox [1].

We denote the set of concepts for a specific DL $\mathcal{L}$ with $\mathcal{C}(\mathcal{L})$, e.g., $\mathcal{C}(\mathcal{EL})$ is the set of all $\mathcal{EL}$-concepts. We call concepts that are either concept names or existential restrictions *atoms* and denote the set of atoms by N_A.

For $\mathcal{EL}$-concepts a unique normal form (modulo associativity and commutativity), was given in [2], which we extend to $\mathcal{ELH}$-concepts in presence of RBoxes. To treat equivalent roles, we define $[r] = \{s \in N_R \mid r \equiv_{\mathcal{R}} s\}$ and fix a function f that picks one role r_i from each equivalence class and replaces each occurrence of a role from $[r_i]$ with r_i. Given an RBox $\mathcal{R}$ and an $\mathcal{ELH}$-concept C, C is in *$\mathcal{ELH}$-normal form*, if the following 4 rules have been applied exhaustively to the concept C and its subconcepts:

1. $A \sqcap \top \longrightarrow A$, 2. $A \sqcap A \longrightarrow A$, 3. $\exists r.C' \longrightarrow \exists f([r]).C'$,
4. $\exists r.C' \sqcap \exists s.D' \longrightarrow \exists r.C'$ if $r \sqsubseteq_{\mathcal{R}} s$ and $C' \sqsubseteq D'$

The transformation of $\mathcal{ELH}$-concepts into $\mathcal{ELH}$-normal form can be done in polynomial time.

3 Properties for Concept Similarity Measures

Formally, a *concept similarity measure sim* is a function mapping from pairs of $\mathcal{ELH}$-concepts to the interval $[0, 1]$. To identify properties of similarity measures for concepts, [8] used *metric spaces* as a starting point, which was also done in other areas of similarity research (see [22, 16, 17, 20]). A metric can be interpreted as a *dissimilarity measure*. The distance represents the dissimilarity between two objects—the lower their distance, the higher the similarity. Using a metric d, we can obtain a similarity function s by defining $s(a, b) := 1 - d(a, b)$. If we adapt the properties of a metric accordingly, we obtain the following properties for similarity functions.

Definition 1. *Let D be a set. A function $s : D \times D \longrightarrow [0, 1]$ is called a* similarity function *for D iff for all $a, b, c \in D$ holds*

1. $s(a, b) = 1 \iff a = b,$ (identity of indiscernibles)
2. $s(a, b) = s(b, a)$, *and* (symmetry)
3. $1 + s(a, b) \geq s(a, c) + s(c, b)$ (triangle inequality).

Next we present definitions of properties of concept similarity measures and the underlying intuitions for these properties. We start with a formal definition of the properties and discuss each of them afterwards.

Definition 2. *Let $C, D, E \in \mathcal{C}(\mathcal{ELH})$. A similarity measure sim is*

1. symmetric *iff* $sim(C, D) = sim(D, C)$.
2. *fulfilling the* triangle inequality *property iff*
$$1 + sim(D, E) \geq sim(D, C) + sim(C, E).$$
3. equivalence invariant *iff* $C \equiv D \implies sim(C, E) = sim(D, E)$.
4. equivalence closed *iff* $sim(C, D) = 1 \iff C \equiv D$.
5. subsumption preserving *iff* $C \sqsubseteq D \sqsubseteq E \implies sim(C, D) \geq sim(C, E)$.
6. reverse subsumption preserving *iff* $C \sqsubseteq D \sqsubseteq E \implies sim(C, E) \leq sim(D, E)$.
7. structurally dependent *iff for all sequences* $(C_n)_n$ *of atoms with* $\forall i, j \in \mathbb{N}, i \neq j : C_i \not\sqsubseteq C_j$ *the concepts*
$$D_n := \bigsqcap_{i \leq n} C_i \sqcap D \quad \text{and} \quad E_n := \bigsqcap_{i \leq n} C_i \sqcap E$$
fulfill the condition $\lim_{n\to\infty} sim(D_n, E_n) = 1$.

The properties 1. to 4. are adopted from the literature, whereas to the best of our knowledge the properties 5. to 7. are introduced for DLs in this paper.

Symmetry is a rather controversial property for similarity in general—while some consider it essential [18], cognitive sciences seems to favor an asymmetric notion of similarity [22, 4]. Even for DL concepts Janowicz et al. [13, 12] prefer asymmetry (but devise symmetric measures), whereas most [3, 7, 6, 10, 8] consider it a fundamental property of similarity of concepts.

Triangle property is inherited from metrics. Two papers mentioned triangle inequality in the context of DLs: [8] argues in favor, while [12] argue against it, because of Tversky's [22] work.

DLs allow the same thing to be described in different ways. Two concepts can be syntactically different and yet semantically equivalent. A similarity measure for complex concepts should depend on the semantics rather than the syntax of the concepts to measure.

Equivalence Invariance ensures that two equivalent concepts have the same similarity towards a third concept. Equivalence invariance is widely accepted as a necessary property for measures for DL concepts ([13, 12, 6, 8]). Yet we found that the methods used to ensure equivalence invariance were not always sound (see Section 3.1).

Equivalence Closure holds for a similarity measure if and only if two concepts are totally similar if and only if they are equivalent. This corresponds with the idea that indiscernible things are identical. Equivalence closure is considered to be a basic property for concept similarity measures [8, 12] especially since it is inherited from metrics.

One asset of DLs is their reasoning services. An intuitive idea is to characterize similarity of concepts in terms of these services. The subsumption relation yields a total partial order on concepts. Consider the case where $C, D, E \in \mathcal{C}(\mathcal{ELH})$ and $C \sqsubseteq D \sqsubseteq E$. A natural requirement of similarity measures is to reflect this constellation.

Subsumption Preservation expresses that the similarity of C and D is higher than the one of C and E because C is 'closer' to D than to E.

Reverse subsumption Preservation states likewise that the similarity of D and E is higher than the similarity of C and E, since E is 'closer' to D than to C.

In [15] we also employ the reasoning service least common subsumer to capture the characteristics of total dissimilarity of concept similarity.

Tversky [22] presents the *feature model*, where an object is described by a set of features. The similarity of two objects is measured by a relation between the number of common features of both objects and the number of unique features of each object. The basic rule is that if

1. the number of common features increases and
2. the number of uncommon features is constant

then the similarity must increase.

Structural Dependence reflects this basic rule. Concepts are our objects to compare and the atoms of a conjunction represent the features of the object. The intuition is that the more features (atoms) two complex concepts share, the higher their similarity should be.

For a more detailed explanation of the last property and for a presentation of examples illustrating the above properties see [15].

Table 1. Overview of similarity measures and their properties

	symm.	triang.	eq. inv.	eq. cl.	subs.	rev. subs.	struc. dep.	DL
simi	✓	-	✓	✓	✓	-	✓	$\mathcal{ELH}$
Jacc [11]	✓	✓	✓	✓	✓	✓	✓	$\mathcal{L}_0$
[13]	✓	-	-	-	-	-	✓	$\mathcal{SHI}$
[12]	✓	-	-	-	-	-	✓	$\mathcal{ALCHQ}$
[7]	-	-	-	-	-	-	-	$\mathcal{ALC}$
[10]	✓	-	✓	-	✓	✓	-	$\mathcal{ALN}$
[6]	✓	-	✓	-	✓	✓	-	$\mathcal{ALC}$
[8]	✓	-	✓	-	✓	✓	-	$\mathcal{ALE}$

3.1 Inspecting Existing Concept Similarity Measures

We distinguish two kinds of concept similarity measures: structural measures and interpretation based measures. *Structural measures* are defined using the syntax of the concepts to measure. Since conjunction and disjunction are commutative and associative, these measures are invariant to the order of the atoms in a conjunction or disjunction. The measures differ regarding the similarity of primitive concepts: [12] uses the TBox whereas [7] and [10] use the canonical interpretation which takes the set of ABox individuals as the interpretation domain (for an introduction to ABoxes see [1]).

Interpretation based measures are defined using interpretations and cardinality, instead of the syntax of the (possibly) complex concepts to measure. Therefore, they are trivially equivalence invariant. The two interpretation based measures we investigated [6, 8] are using the canonical interpretation $\mathcal{I}_\mathcal{A}$. These measures need a populated and representative ABox as a significant domain.

Table 1 presents an overview of similarity measures for concepts written in different DLs (including our measure *simi* to be defined in Section 4) and whether or not they fulfill the properties from Definition 2. The proofs can be found in [15]. The first four measures are purely structural measures. The next two are structural measures which use the canonical interpretations to measure primitives. The last two are purely interpretation based measures.

We included the Jaccard index [11], which is originally a set measure, here adapted to $\mathcal{L}_0$. Interestingly, this is the only measure of those investigated that fulfills the triangle inequality.

Our thorough investigation of the similarity measures defined in the literature showed that defining a similarity measure that fulfills most of the properties from Definition 2 is by no means a trivial task—in particular if the DL allows the use of roles, as the lightweight DL $\mathcal{ELH}$ already does.

4 Developing Concept Similarity Measures for $\mathcal{ELH}$

We present *simi*, a framework for similarity measures for concepts written in the DL $\mathcal{ELH}$ based on the semantics of the logic. It operates on (complex) concepts and an RBox $\mathcal{R}$, which contains role inclusion axioms. If concepts to be processed contain concepts defined in an unfoldable TBox $\mathcal{T}$, we assume that these concepts are expanded w.r.t. $\mathcal{T}$, i.e., all concept names occurring in them are primitive names.

Another preprocessing step is to transform the concepts into $\mathcal{ELH}$-normal form (defined in Section 2). Concepts in this normal form are unique (modulo associativity and commutativity), which ensures that *simi* (and any other measure processing concepts in this normal form) is equivalence invariant. We assume for the remainder of the paper that the concepts are in $\mathcal{ELH}$-normal form.

The framework *simi* constructs similarity measures from several free parameters, i.e., it allows functions to be combined in such a way that, if these functions fulfill certain properties, then the resulting similarity measure can be shown to fulfill all properties from Definition 2 except reverse subsumption preserving and the triangle inequality. Furthermore, it can be computed efficiently.

Simi is inspired by the Jaccard index and it is a conservative extension of the Jaccard index, in the sense that $\forall C, D \in \mathcal{C}(\mathcal{L}_0) : simi(C,D) = Jacc(C,D)$ (proven in [15]). Another inspiration is the equivalence of concepts, which can be regarded as a trivial similarity measure: the similarity of two concepts is 1 if they are equivalent and 0 otherwise. To determine if $C \equiv D$ is true, one can use the subsumption test to find out whether or not $C \sqsubseteq D$ and $D \sqsubseteq C$ are true. We generalize this approach in *simi* by introducing a generalization of the subsumption operator. Since such an operator is in general an asymmetric function, we call it *directed simi* and denote it with $simi_d$ (to be introduced in Section 4.1). Now, once the values $simi_d(C,D)$ and $simi_d(D,C)$ are computed, we have to combine them with an operator to obtain a value for *simi*. Instead of fixing a specific operator, we identify the properties such an operator needs to provide such that *simi* fulfills as many of the properties as possible. We call such an operator a *fuzzy connector* (denoted with $\otimes$). A fuzzy connector $\otimes$ is an operator on the interval $[0,1]$, $\otimes : [0,1]^2 \longrightarrow [0,1]$ such that for all $x, y \in [0,1]$ the following properties are true.

- Commutativity: $x \otimes y = y \otimes x$,
- Equivalence closed: $x \otimes y = 1 \iff x = y = 1$,
- Weak monotonicity: $x \leq y \implies 1 \otimes x \leq 1 \otimes y$,
- Bounded: $x \otimes y = 0 \implies x = 0$ or $y = 0$ and
- Grounded: $0 \otimes 0 = 0$.

Using a fuzzy connector, *simi* is simply defined as

$$simi(C,D) := simi_d(C,D) \otimes simi_d(D,C)$$

where C and D are arbitrary $\mathcal{ELH}$-concepts.

The commutativity of a fuzzy connector ensures that *simi* is symmetric, the property equivalence closed provides the same property for the resulting similarity measure and weak monotonicity is sufficient to prove that *simi* fulfills subsumption preserving. Examples for fuzzy connectors are the average and triangular norms (t-norms, $\otimes$) [14] which fulfill the property that for all $x, y \in [0,1] : x \otimes y = 0 \implies x = 0$ or $y = 0$ as shown in [15].

4.1 A Directed Similarity Measure: *simi*$_d$

To formulate *simi*$_d$, we need a bit of notation. If convenient, we treat concepts as sets of atoms. Let $C \in \mathcal{C}(\mathcal{ELH})$, then it can be written as $C = \bigsqcap_{i \leq n} C_i$ where $\forall i \leq n: \; C_i \in N_A$. The function $(\widehat{\cdot})$ maps concepts to sets of atoms, so for C, $\widehat{C} := \{C_1, C_2, \ldots, C_n\}$. Now, the starting point for the derivation of *simi*$_d$ is the function

$$d(C, D) := \frac{|\widehat{C} \cap \widehat{D}|}{|\widehat{C}|}$$

which is inspired by the Jaccard Index. This function can be used to measure the similarity of sets of concept names. In order to be able to incorporate existential restrictions, we rewrite the numerator of d to

$$|\widehat{C} \cap \widehat{D}| = \sum_{C' \in \widehat{C}} \max_{D' \in \widehat{D}} f(C', D'), \tag{1}$$

where the function $f : N_C \longrightarrow \{0, 1\}$ is defined as $f(C', D') := 1$ if $C' = D'$ and 0 otherwise.

The simplifying assumption for f is that two different concept names denote always totally dissimilar concepts. However, this assumption may not be correct in all cases. Therefore, we generalize f by introducing a measure for concept names. In order to work for existential restrictions, this measure has to be able to deal with role names, too. In addition, we have to ensure some properties for this measure to guarantee properties for *simi*. We call this measure for (concept or role) names a *primitive measure* and denote it with *pm*. More formally, it is a function of type $pm : N_C^2 \cup N_r^2 \longrightarrow [0, 1]$ with the property that for all $A, B \in N_C$ and $r, s, t \in N_r$ the following holds:

- $pm(A, B) = 1 \iff A = B$,
- $pm(r, s) = 1 \iff s \sqsubseteq r$,
- $s \sqsubseteq_{\mathcal{R}} r \implies pm(s, r) > 0$, and
- $t \sqsubseteq_{\mathcal{R}} s \implies pm(r, s) \leq pm(r, t)$.

The first two properties are sufficient to ensure that *simi* fulfills equivalence closed and the last one is needed to prove that *simi* fulfills subsumption preserving. Note that *pm* does not need to be symmetric.

To incorporate existential restrictions into d we have three cases to consider. Namely, we need to be able to compute the similarity of two concept names, of a concept name and an existential restriction and of two existential restrictions. The first case is handled directly by the primitive measure *pm*. In the second case, we assert that a concept name and an existential restriction are always totally dissimilar and thus their similarity is 0. For the third case, let $\exists r.C^*$ and $\exists s.D^*$ be the two existential restrictions. To compute the similarity of both atoms, we proceed component-wise. The similarity of the role names is computed using the primitive measure *pm* and the similarity of the concepts C^* and D^*

is computed by a recursive call to d. Then, to combine both values we use a number $w \in (0, 1)$ and the formula

$$d(\exists r.C^*, \exists s.D^*) := pm(r, s) \cdot [w + (1 - w) \cdot d(C^*, D^*)].$$

Forcing $w > 0$, enables us for $d(C^*, D^*) = 0$ to distinguish between the cases where the roles are similar and where they are not. In the first case, the similarity is w, whereas in the second one, the similarity is 0.

As a suitable w, we suggest the value n where one would say that the concepts

$$C := \underbrace{\exists r. \cdots \exists r.}_{n} A \text{ and } D := \underbrace{\exists r. \cdots \exists r.}_{n} B$$

with $pm(A, B) = 0$ are regarded (almost) totally similar.

In Equation 1, we search for each atom of C for that atom of D with the highest similarity value. This method does not always yield satisfactory results. Consider the case, where $pm(A, B_1) = 0.5$ and $pm(A, B_2) = 0.5$ and we want to measure A towards $B_1 \sqcap B_2$, then the current version of function d does not take into account that A is 'known to be similar' to each of B_1 and B_2 alone and thus should even be more similar to their combination. The function chooses *one* 'best matching partner' instead of combining the two sources of similarity.

To deal with this effect, we propose to replace the maximum operator with a triangular conorm (t-conorm, $\oplus$) [14] which is *bounded*, meaning that for all $x, y \in [0, 1] : x \oplus y = 1 \implies x = 1$ or $y = 1$. There are several reasons for the use of a t-conorm. First, the operator *max* is an instance of a bounded t-conorm. Second, all t-conorms yield values greater or equal to those of *max* which is consistent with our expectation that the value should be higher or equal to the maximum. Also, 0 acts as neutral element for t-conorms. Therefore, all atoms from D that are totally dissimilar do not influence the value. If we use the probabilistic sum ($x \oplus_{sum} y = x + y - xy$) instead of the maximum for our example above, then we obtain the value 0.75 instead of 0.5, since the measure takes both similarity values (towards B_1 and B_2) into account.

Another parameter of $simi_d$ is the *weighting function* (denoted g). It weights the atoms by assigning each of them a value greater than 0, so $g : N_A \longrightarrow \mathbb{R}_{>0}$. The effect is that some atoms can 'contribute more' to the similarity than others, thus a part of the vocabulary can be picked by g to supply a context under which the concepts from the KB are assessed. Let's assume we are interested in similarity regarding Anatomy and our KB, say SNOMED, contains atoms from two different subject areas like Anatomy and medical procedures. Now, weighting the atoms related to Anatomy higher would result in their similarity having a greater influence on the overall similarity value between concepts.

Note, that the KB does not need to be changed or adapted to achieve this. Several different such weighting functions can easily be employed for the same KB. To incorporate the weighting function we generalize the cardinality of a set of atoms to the sum of the weights of its elements. To obtain a well-defined measure, the weight needs to be added to the numerator of d as well.

By combining the above presented parts, we can already obtain a definition of $simi_d$ except for some corner cases involving $\top$. If we want to be formally correct, then the type of the function $simi_d$ depends on the used parameters as well as on the concepts to be measured. However, for better readability, we omit writing these parameters.

Definition 3 **($simi_d$).** *Let $C, D \in \mathcal{C}(\mathcal{ELH}) \setminus \{\top\}$, $E, F \in \mathcal{C}(\mathcal{ELH})$, $A, B \in N_C$ and $r, s \in N_R$.* Directed simi *is the function $simi_d : \mathcal{C}(\mathcal{ELH})^2 \longrightarrow [0,1]$ defined (w.r.t. a bounded t-conorm $\oplus$, a primitive measure pm, a weighting function g and $w \in (0,1)$) by*

$$\begin{aligned} simi_d(\top, \top) := simi_d(\top, D) \ &:= \ 1, \\ simi_d(C, \top) := 0, & \end{aligned}$$

$$simi_d(C, D) := \frac{\sum\limits_{C' \in \widehat{C}} [g(C') \cdot \bigoplus_{D' \in \widehat{D}} simi_a(C', D')]}{\sum\limits_{C' \in \widehat{C}} g(C')},$$

where $simi_a$ measures the similarity of two atoms and is defined as

$$\begin{aligned} simi_a(A, B) &:= pm(A, B), \\ simi_a(\exists r.E, A) &:= simi_a(A, \exists r.E) \quad := \quad 0, \\ simi_a(\exists r.E, \exists s.F) &:= pm(r, s) \cdot [w + (1 - w) simi_d(E, F)]. \end{aligned}$$

4.2 Properties of $simi_d$ and $simi$

We present the lemma needed to prove various properties of *simi*. The proofs can be found in [15] (p. 67 ff). In the following we assume that the primitive measure is *pm*, the weighting function is g, the t-conorm is $\oplus$ and the fuzzy connector is $\otimes$.

Lemma 1. *Let $C, D, E \in \mathcal{C}(\mathcal{ELH})$. Then*

1. $simi_d(C, D) = 1 \iff D \sqsubseteq C$.
2. $D \sqsubseteq E \implies simi_d(C, E) \leq simi_d(C, D)$.

Proof. We present a proof sketch for the left-to-right implication of the first statement. Let $simi_d(C, D) = 1$. If $C = \top$ then $D \sqsubseteq C = \top$ is true. Let $C \neq \top$. To prove $D \sqsubseteq C$ we have to show that $\forall C' \in \widehat{C}\ \exists D' \in \widehat{D} :\ D' \sqsubseteq C'$. Let C' be an arbitrary atom of C. $simi_d(C, D) = 1$ implies that

$$\sum_{C' \in \widehat{C}} g(C') = \sum_{C' \in \widehat{C}} [g(C') \cdot \bigoplus_{D' \in \widehat{D}} simi_a(C', D')].$$

Because of $g(C') \cdot \bigoplus_{D' \in \widehat{D}} simi_a(C', D') \leq g(C')$ we derive that for all $C' \in \widehat{C} :\ \bigoplus_{D' \in D} simi_a(C', D') = 1$. Since the t-conorm is bounded, $\exists D' \in D$ such that $simi_a(C', D') = 1$. The rest of the proof uses structural induction and case distinction.

If $C' = A$ then $simi_a(C', D') = 1$ leads to $D' = A$ which implies $D' \sqsubseteq C'$. Next, let $C' = \exists r.C^*$. $simi_a(C', D') = 1$ implies that D' is of the form $D' = \exists s.D^*$ and $1 = pm(r, s) \cdot [w + (1 - w)simi_d(C^*, D^*)]$. This leads to $pm(r, s) = 1$ which according to the definition of the primitive measure implies $s \sqsubseteq r$. Since $w < 1$, $simi_d(C^*, D^*) = 1$. Using the induction hypothesis we can derive $D^* \sqsubseteq C^*$, therefore $D' \sqsubseteq C'$.

Recall, $simi(C, D) := simi_d(C, D) \otimes simi_d(D, C)$. The resulting function has the following properties.

Theorem 1. *The function simi fulfills*

1. *symmetry,*
2. *equivalence invariance,*
3. *equivalence closed,*
4. *subsumption preserving.*

Let g' be a weighting function with $inf\{g(C') \mid C' \in \mathcal{C}(\mathcal{ELH})\} > 0$. Furthermore, let $\otimes'$ be a fuzzy connector s.t. for all sequences $(x_n)_n$ and $(y_n)_n$ ($x_i, y_i \in [0, 1]$) with $\lim_{n\to\infty} x_n = \lim_{n\to\infty} y_n = 1$ and $\lim_{n\to\infty} x_n \otimes' y_n = 1$. Then simi together with $\otimes'$ and g' fulfills structural dependence.

The main reason why *simi* neither fulfills the triangle inequality nor reverse subsumption preserving is that the computation of $simi_d(C, D)$ does not use the similarity values between the atoms of C (and between the atoms of D). Consider $C := A \sqcap \bigsqcap_{i\leq n} B_i$, where the B_i are very similar to each other, $D := A \sqcap B_0$ and $E := A$ then the similarity of D and E is approximately 0.5, the similarity of C and D is close to 1 (since each B_i is very similar to B_0) but the similarity of C and E converges to 0 with increasing n. For the proofs of other properties of *simi* and further details see [15].

An important property of *simi* is that it can be computed efficiently, provided that the involved parameter functions can be computed efficiently as well.

Lemma 2. *If the specific fuzzy connector, the bounded t-conorm, the primitive measure and the weighting function can be computed in polynomial time, then simi can be computed in time polynomial in the size of the concepts to measure.*

5 Conclusions

Similarity measures are important procedures for central ontology management tasks such as alignment of ontologies. Often these measures are built in an ad-hoc way by simply tuning them to test data.

In this paper we have proposed a different approach to construct a whole range of such measures for $\mathcal{ELH}$-concepts. Our starting point was a set of formally defined properties for concept similarity measures, which make use of the semantics of DL concepts and of DL reasoning services. We devised a framework that, if instantiated with appropriate functions and operators as discussed in this paper, allows to generate similarity measures that have 5 of the proposed 7 properties (reverse subsumption preservation and triangle inequality are missing). In that sense one could claim that our framework for similarity measures is not only semantics-based, but also provides the measures with semantics. Moreover, our approach does not restrict users to a single similarity measure, but allows them to design their own measures by selecting the functions and operators appropriate to yield the needed individual similarity measure. If the selected functions conform to the framework described in this paper, the resulting similarity measure is equipped with the properties.

Similarity is often perceived as a context-dependent characteristic. Even in this case our framework can offer support, in the sense that the directed measure $simi_d$ allows atoms appearing in the concept to be weighted differently using the weighting function g. Different instantiations of g allow different thematic subdomains of the domain of discourse to be highlighted.

To test our framework empirically is a non-trivial task, since each application may require a different instantiation of *simi* with functions and operators. To aquire such instantiations suitable for each application requires profound knowledge of the application in question. Thus for now it remains future work to compare the outcome of *simi* instantiations with other well-accepted similarity measures.

On the theoretical side it would be interesting to investigate such frameworks for more expressive DLs and for the concepts defined w.r.t. general TBoxes. Since a unique normal form is the main means to achieve an equivalence invariant similarity measure, it is not obvious how to extend *simi* to these more expressive scenarios.

References

[1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)

[2] Baader, F., Küsters, R., Molitor, R.: Computing least common subsumers in description logics with existential restrictions. In: Dean, T. (ed.) Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI 1999), Stockholm, Sweden, pp. 96–101. Morgan Kaufmann, Los Altos (1999)

[3] Borgida, A., Walsh, T.J., Hirsh, H.: Towards measuring similarity in description logics. In: Proceedings of the International Workshop on Description Logics (DL 2005) (2005)

[4] Bowdle, B., Gentner, D.: Informativity and asymmetry in comparisons. Cognitive Psychology 34(3), 244–286 (1997); PMID: 9466832

[5] T.G.O. Consortium. Gene Ontology: Tool for the unification of biology. Nature Genetics 25, 25–29 (2000)

[6] d'Amato, C., Fanizzi, N., Esposito, F.: A semantic similarity measure for expressive description logics. In: Convegno Italiano di Logica Computazionale (CILC 2005) (2005)

[7] d'Amato, C., Fanizzi, N., Esposito, F.: A dissimilarity measure for $\mathcal{ALC}$ concept descriptions. In: Proceedings of the ACM Symposium on Applied Computing, SAC 2006, pp. 1695–1699 (2006)

[8] d'Amato, C., Staab, S., Fanizzi, N.: On the Influence of Description Logics Ontologies on Conceptual Similarity. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 48–63. Springer, Heidelberg (2008)

[9] Euzenat, J., Valtchev, P.: Similarity-based ontology alignment in OWL-lite. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), pp. 333–337. IOS Press (2004)

[10] Fanizzi, N., d'Amato, C.: A similarity measure for the $\mathcal{ALN}$ description logic. In: Convegno Italiano di Logica Computazionale (CILC 2006) (2006)

[11] Jaccard, P.: Étude comparative de la distribution florale dans une portion des alpes et des jura. Bulletin de la Societe Vaudoise des Sciences Naturelles 37, 547–579 (1901)

[12] Janowicz, K.: SIM-DL: Towards a semantic similarity measurement theory for the description logic $\mathcal{ALCNR}$ in geographic information retrieval. In: SeBGIS 2006, OTM Workshops 2006, pp. 1681–1692 (2006)

[13] Janowicz, K., Wilkes, M.: SIM-DLA: a novel semantic similarity measure for description logics reducing Inter-Concept to Inter-Instance similarity. In: Proceedings of the 6th European Semantic Web Conference on The Semantic Web Research and Applications, pp. 353–367 (2009)

[14] Klement, E.P., Mesiar, R., Pap, E.: Triangular Norms. SV (2000)

[15] Lehmann, K.: A framework for semantic invariant similarity measures for $\mathcal{ELH}$ concept descriptions. Master's thesis, TU Dresden (2012), `http://lat.inf.tu-dresden.de/research/mas`

[16] Li, M., Chen, X., Li, X., Ma, B., Vitányi, P.M.B.: The similarity metric. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 863–872 (2003)

[17] Li, M., Sleep, M.R.: Melody classification using a similarity metric based on Kolmogorov complexity. In: Proceedings of the Sound and Music Computing Conference (SMC 2004) (2004)

[18] Lin, D.: An Information-Theoretic definition of similarity. In: Proceedings of the Fifteenth International Conference on Machine Learning, pp. 296–304 (1998)

[19] Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 web ontology language profiles. W3C Recommendation (October 27, 2009), `http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/`

[20] Nikolova, N., Jaworska, J.: Approaches to measure chemical similarity - a review. QSAR & Combinatorial Science 22, 1006–1026 (2003)

[21] Spackman, K.: Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed-rt. Journal of the American Medical Informatics Assoc. (2000), Fall Symposium Special Issue

[22] Tversky, A.: Features of similarity. Psychological Review 84, 327–352 (1977)

[23] W3C OWL Working Group. OWL 2 web ontology language document overview. W3C Recommendation (October 27, 2009), `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`

Sequent Systems for Lewis' Conditional Logics[⋆]

Björn Lellmann and Dirk Pattinson

Department of Computing, Imperial College London, UK

Abstract. We present unlabelled cut-free sequent calculi for Lewis' conditional logic $\mathbb{V}$ and extensions, in both the languages with the entrenchment connective and the strong conditional. The calculi give rise to PSPACE-decision procedures, also in the language with the weak conditional. Furthermore, they are used to prove the Craig interpolation property for all the logics under consideration, and yield a PSPACE-decision procedure for a recently considered hybrid version of $\mathbb{V}$.

1 Introduction

Although the use of conditional logics in artificial intelligence and automated reasoning has a long tradition (e.g. [6]), there has been slow progress in the development of proof systems for these logics. Even today, we still see conditional logics for which no proof systems of optimal complexity have been found. In general, the development of proof systems follows two main approaches: one can derive labelled tableau systems from the semantics [14,8] or convert a Hilbert-style axiomatisation to an unlabelled sequent system which is then saturated to guarantee cut-elimination [15,17,10].

Although proof systems for some of the more prominent logics have been developed quite early on [2,9,4,1], the systematic exploration of systems with optimal complexity has attracted interest only recently. In particular, there are no complexity-optimal proof systems for an important class of logics, those that are interpreted over *sphere models*, originally proposed by Lewis [11]. These logics can be characterised using different connectives: the *entrenchment connective* $\preccurlyeq$, the *strong conditional* $\Box\!\!\Rightarrow$, and the *weak conditional* $\Box\!\!\rightarrow$. While these connectives are interdefinable, the translations in general yield an exponential blow-up, and thus complexity results do not necessarily carry over.

Although the logics in the weak conditional language have long been known to be decidable in polynomial space [3], the best proof systems for this language so far only yield a coNEXPTIME upper bound [8]. For the entrenchment connective, there are systems for the logics $\mathbb{VC}$ and $\mathbb{VCS}$, which implicitly yield a PSPACE upper bound [2,4], but no systematic treatment has been given yet, a gap that this paper now closes.

Our main results are the following: we present complexity-optimal unlabelled sequent calculi for the logics $\mathbb{V}$, $\mathbb{VN}$, $\mathbb{VT}$, $\mathbb{VW}$ and $\mathbb{VC}$ in the entrenchment and strong conditional language. With the exception of the calculus for $\mathbb{VC}$ in the entrenchment language these seem to be new. Cut elimination for our calculi follows from the more general approach of *cut elimination by saturation*, and yields purely syntactical decision procedures of optimal PSPACE complexity. A PSPACE decision procedure for the logics in the weak conditional language is established by means of translation. As an application,

⋆ Supported by EPSRC-Project EP/H016317/1.

we establish the Craig interpolation property for all logics considered (in any connective), which we believe is also a new result. Our second application yields a previously unknown PSPACE result for a hybrid version of $\mathbb{V}_{\Box\!\Rightarrow}$ recently considered in [16].

2 Preliminaries

For $n \in \{0, 1, 2, \ldots\}$ we write $[n]$ for $\{1, \ldots, n\}$. Throughout, $\mathcal{V}$ denotes a denumerable set of propositional variables, written $p, q, \ldots$ and we use bold face $\boldsymbol{p}, \boldsymbol{q}, \ldots$ to denote finite sequences of propositional variables. We fix a set Λ of modal operators with associated arities (later, we will specialise Λ to consist of just one binary conditional operator). The set of *Λ-formulae* is defined by $\mathcal{F}(\Lambda) \ni A, B, A_1, \ldots, A_n ::= \bot \mid p \mid A \wedge B \mid A \vee B \mid A \to B \mid \heartsuit(A_1, \ldots, A_n)$ for $p \in \mathcal{V}$ and $\heartsuit \in \Lambda$ with arity n, with remaining connectives defined as usual. We write $\Lambda(S) = \{\heartsuit(A_1, \ldots, A_n) \mid \heartsuit \in \Lambda\ n\text{-ary}, A_1, \ldots, A_n \in S\}$ for the set of formulae constructed from S using a single connective in Λ and $\mathsf{var}(A)$ for the set of propositional variables occurring in the formula A. Uniform substitution of all propositional variables in a formula A using a substitution $\sigma : \mathcal{V} \to \mathcal{F}(\Lambda)$ is denoted by $A\sigma$. A *Λ-logic*, or just *logic* is a set $\mathcal{L} \subseteq \mathcal{F}(\Lambda)$ that contains all propositional tautologies and is closed under uniform substitution, modus ponens and the congruence rule: from $A_i \leftrightarrow B_i$ for $i = 1, \ldots, n$ infer $\heartsuit(A_1, \ldots, A_n) \leftrightarrow \heartsuit(B_1, \ldots, B_n)$ for every n-ary modality $\heartsuit \in \Lambda$. We think of logics as given semantically as the set of universally valid formulae on some class of models and write $\models_{\mathcal{L}} A$ for $A \in \mathcal{L}$. The set $\mathcal{S}(F)$ of *sequents* over F consists of tuples of multisets Γ, Δ of formulae in F, written $\Gamma \Rightarrow \Delta$. The multiset union of two multisets Γ and Δ is written Γ, Δ and we identify formulae with singleton multisets. Substitution extends to both multisets of formulae and sequents in the obvious way (perserving multiplicity), e.g. $(A_1, A_2 \Rightarrow B)\sigma = A_1\sigma, A_2\sigma \Rightarrow B\sigma$. We use the system $\mathsf{G3}cp$ of [18] with axioms $\Gamma, A \Rightarrow \Delta, A$ (where A ranges over the set of formulae) as basis for all systems that extend classical propositional logic and denote its proof rules by G. We adopt the standard structural rules

$$\frac{\Gamma \Rightarrow \Delta}{\Sigma, \Gamma \Rightarrow \Delta, \Pi}\ \mathsf{W}, \quad \frac{\Gamma, A, A \Rightarrow \Delta}{\Gamma, A \Rightarrow \Delta}\ \mathsf{ConL}, \quad \frac{\Gamma \Rightarrow \Delta, A, A}{\Gamma \Rightarrow \Delta, A}\ \mathsf{ConR}, \quad \frac{\Gamma \Rightarrow \Delta, A \quad \Sigma, A \Rightarrow \Pi}{\Gamma, \Sigma \Rightarrow \Delta, \Pi}\ \mathsf{Cut}$$

and write $\models_{\mathcal{L}} \Gamma \Rightarrow \Delta$ if $\mathcal{L}$ is a logic and $\models_{\mathcal{L}} \bigvee \Gamma \to \bigwedge \Delta$.

3 Conditional Logics: Calculi and Main Results

We consider the conditional logics $\mathbb{V}, \mathbb{VN}, \mathbb{VT}, \mathbb{VW}$ and $\mathbb{VC}$ [11,13] in the languages of (binary) *entrenchment* $\preccurlyeq$ and (binary) *weak* and *strong conditionals* $\Box\!\!\rightarrow$ and $\Box\!\!\Rightarrow$. We read entrenchment $A \preccurlyeq B$ as 'A is at least as plausible as B' and adopt Lewis' *sphere semantics*: a *sphere model* is a triple $\mathcal{I} = (I, (\$_i)_{i \in I}, \pi)$ where I is a set (of worlds), each $\$_i \subseteq \mathcal{P}(I)$ is a *system of spheres*, i.e. a family of nested subsets of I closed under unions and nonempty intersections, and $\pi : \mathcal{V} \to \mathcal{P}(I)$ is a valuation. We think of $\$_i$ as providing a measure of comparative similarity, which provides the truth condition

$$\mathcal{I}, i \models A \preccurlyeq B \iff \text{for all spheres } S \in \$_i\ (S \cap [\![B]\!] \neq \emptyset \text{ only if } S \cap [\![A]\!] \neq \emptyset)$$

(CP)	$\dfrac{\vdash B \to (A_1 \vee \dots \vee A_n)}{\vdash (A_1 \preccurlyeq B) \vee \dots \vee (A_n \preccurlyeq B)}$ $(n \geq 1)$	(N)	$\neg(\bot \preccurlyeq \top)$
		(T)	$(\bot \preccurlyeq \neg A) \to A$
(TR)	$((A \preccurlyeq B) \wedge (B \preccurlyeq C)) \to (A \preccurlyeq C)$	(W)	$((\bot \preccurlyeq \neg A) \vee \neg(\neg A \preccurlyeq \top)) \to A$
(CN)	$(A \preccurlyeq B) \vee (B \preccurlyeq A)$	(C)	$((A \preccurlyeq \top) \wedge (\top \preccurlyeq A)) \to A$
$\mathcal{H}\mathbb{V}_{\preccurlyeq}$: CP, TR, CN	$\mathcal{H}\mathbb{VN}_{\preccurlyeq} : \mathcal{H}\mathbb{V}, N$ $\mathcal{H}\mathbb{VT}_{\preccurlyeq} : \mathcal{H}\mathbb{V}, T$	$\mathcal{H}\mathbb{VW}_{\preccurlyeq} : \mathcal{H}\mathbb{V}, W$	$\mathcal{H}\mathbb{VC}_{\preccurlyeq} : \mathcal{H}\mathbb{V}, C$

Fig. 1. Hilbert axiomatisation of the V-logics as smallest logics closed under rules/axioms

where $[\![A]\!] = \{i \in I \mid i \models A\}$ is the truth set of a formula A, together with the standard clauses for propositional variables and boolean connectives. The *strong conditional operator* $\Box\!\!\Rightarrow$ can be defined in terms of entrenchment by $(A \Box\!\!\Rightarrow B) \leftrightarrow \neg((A \wedge \neg B) \preccurlyeq (A \wedge B))$. Over a sphere model, $A \Box\!\!\Rightarrow B$ asserts that $A \wedge B$ is more possible or plausible than $A \wedge \neg B$. This leads to the interpretation

$$\mathcal{I}, i \models A \Box\!\!\Rightarrow B \iff \text{for some sphere } S \in \$_i \; (S \cap [\![A]\!] \neq \emptyset \text{ but } S \cap [\![A \wedge \neg B]\!] = \emptyset)\,.$$

Similarly, the weak conditional $\Box\!\!\rightarrow$ can be expressed in terms of entrenchment by $(A \Box\!\!\rightarrow B) \leftrightarrow ((\bot \preccurlyeq A) \vee \neg((A \wedge \neg B) \preccurlyeq (A \wedge B)))$ where the only difference is that a weak conditional $A \Box\!\!\rightarrow B$ is also accepted if the conditional antecedent A is considered impossible, i.e. false in every sphere for the current world.

If A is a formula and C is a class of sphere models, then A is *universally valid* on C if $\mathcal{I}, i \models A$ for all $\mathcal{I} = (I, (\$_i)_{i \in I}, \pi) \in C$ and all $i \in I$. We write $\mathbb{V}_*$ for the logic of all sphere models, i.e. the set of all formulae that are universally valid in all sphere models in the language of the binary connective $* \in \{\preccurlyeq, \Box\!\!\Rightarrow, \Box\!\!\rightarrow\}$. We consider the following extensions [11, page 120] of $\mathbb{V}_*$ determined by the following additional conditions on sphere models $\mathcal{I} = (I, (\$_i)_{i \in I}, \pi)$, understood as universally quantified over all $i \in I$:

- The logic $\mathbb{VN}_*$ is determined by all *normal* sphere models, i.e. those with $\bigcup \$_i \neq \emptyset$
- The logic $\mathbb{VT}_*$ is determined by all *totally reflexive* sphere models, i.e. $i \in \bigcup \$_i$
- The logic $\mathbb{VW}_*$ is the logic of all *weakly centered* sphere models, i.e. those for which there is $S \in \$_i$ with $S \neq \emptyset$ and $i \in S'$ whenever $\emptyset \neq S' \in \$_i$
- The logic $\mathbb{VC}_*$ is the logic of all *centered* sphere models, i.e. those with $\{i\} \in \$_i$.

Those logics are known [11, pages 124–130] to enjoy a sound and complete axiomatisation in a Hilbert calculus with rules and axioms summarised in Figure 1. By reducing the decision problem for standard modal logics K, D, T to the decision problems for the corresponding conditional logics [11, p.137] using the translations $\Diamond A \leftrightarrow \neg(\bot \preccurlyeq A)$ and $\Diamond A \leftrightarrow (A \Box\!\!\Rightarrow \top)$ and $\Diamond A \leftrightarrow \neg(A \Box\!\!\rightarrow \bot)$ all the logics are easily seen to be PSPACE-hard. Our main contribution are new, cut-free sequent calculi for the logics above that enable backwards proof search in polynomial space. Our calculi contain the standard rules for the propositional connectives together with the rules summarised in Figure 2. Intuitively, rules $R_{1,2}$ and $R_{2,0}$ guarantee derivability of the axioms (TR) and (CN), while the rules $R_{n,0}$ cover the rules of (CP). The remaining rules of $\mathcal{R}_{\mathbb{V}_{\preccurlyeq}}$ are needed to guarantee saturation (see Section 5), and additional rules for the other logics correspond to additional axioms. The rule sets for $\Box\!\!\Rightarrow$ are constructed by translation.

$$\frac{\{\, B_k \Rightarrow A_1,\dots,A_n,D_1,\dots,D_m \mid k \le n \,\} \cup \{\, C_k \Rightarrow A_1,\dots,A_n,D_1,\dots,D_{k-1} \mid k \le m \,\}}{\Gamma,(C_1 \preccurlyeq D_1),\dots,(C_m \preccurlyeq D_m) \Rightarrow \Delta,(A_1 \preccurlyeq B_1),\dots,(A_n \preccurlyeq B_n)}\; R_{n,m}$$

$$\frac{A \Rightarrow \qquad \Rightarrow B}{\Gamma,(A \preccurlyeq B) \Rightarrow \Delta}\; R_N \qquad \frac{A \Rightarrow \qquad \Gamma \Rightarrow \Delta, B}{\Gamma,(A \preccurlyeq B) \Rightarrow \Delta}\; R_T$$

$$\frac{\{\, C_k \Rightarrow A_1,\dots,A_n,D_1,\dots,D_{k-1} \mid k \le m \,\} \qquad \Gamma \Rightarrow \Delta,A_1,\dots,A_n,D_1,\dots,D_m}{\Gamma,(C_1 \preccurlyeq D_1),\dots,(C_m \preccurlyeq D_m) \Rightarrow \Delta,(A_1 \preccurlyeq B_1),\dots,(A_n \preccurlyeq B_n)}\; W_{n,m}$$

$$\frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta,(A \preccurlyeq B)}\; R_{C1} \qquad \frac{\Gamma, A \Rightarrow \Delta \qquad \Gamma \Rightarrow \Delta, B}{\Gamma,(A \preccurlyeq B) \Rightarrow \Delta}\; R_{C2}$$

$$\frac{\begin{array}{c}\{\, C_k,\{B_i \mid i \in I\} \Rightarrow \{A_i \mid i \notin I\},\{C_j \mid j \in J\},\{D_j \mid k > j \notin J\} \mid k \le m,\ I \subseteq [n],\ J \subseteq [k-1] \,\}\\ \cup \{\, A_k, B_k, \{\, B_i \mid i \in I \,\} \Rightarrow \{\, A_i \mid i \notin I \,\},\{C_j \mid j \in J\},\{D_j \mid j \notin J\} \mid k \le n, I \subseteq [n], J \subseteq [m] \,\}\end{array}}{\Gamma,(A_1 \Box\!\!\Rightarrow B_1),\dots,(A_n \Box\!\!\Rightarrow B_n) \Rightarrow \Delta,(C_1 \Box\!\!\Rightarrow D_1),\dots,(C_m \Box\!\!\Rightarrow D_m)}\; R'_{n,m}$$

$$\frac{\Rightarrow A \qquad \Rightarrow B}{\Gamma \Rightarrow \Delta,(A \Box\!\!\Rightarrow B)}\; R'_N \qquad \frac{\Gamma \Rightarrow \Delta, A \qquad A \Rightarrow B}{\Gamma \Rightarrow \Delta,(A \Box\!\!\Rightarrow B)}\; R'_T$$

$$\frac{\begin{array}{c}\{\, C_k,\{B_i \mid i \in I\} \Rightarrow \{A_i \mid i \notin I\},\{C_j \mid j \in J\},\{D_j \mid k > j \notin J\} \mid k \le m,\ I \subseteq [n],\ J \subseteq [k-1] \,\}\\ \cup \{\, \Gamma,\{B_i \mid i \in I\} \Rightarrow \{A_i \mid i \notin I\},\{C_j \mid j \in J\},\{D_j \mid j \notin J\} \mid I \subseteq [n], J \subseteq [m] \,\}\end{array}}{\Gamma,(A_1 \Box\!\!\Rightarrow B_1),\dots,(A_n \Box\!\!\Rightarrow B_n) \Rightarrow \Delta,(C_1 \Box\!\!\Rightarrow D_1),\dots,(C_m \Box\!\!\Rightarrow D_m)}\; W'_{n,m}$$

$$\frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma, B \Rightarrow \Delta}{\Gamma,(A \Box\!\!\Rightarrow B) \Rightarrow \Delta}\; R'_{C1} \qquad \frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta,(A \Box\!\!\Rightarrow B)}\; R'_{C2}$$

$$\begin{array}{ll}
\mathcal{R}_{\mathbb{V}_\preccurlyeq} = \{R_{n,m} \mid n \ge 1, m \ge 0\} & \mathcal{R}_{\mathbb{V}_{\Box\!\Rightarrow}} = \{R'_{n,m} \mid n \ge 1, m \ge 0\}\\
\mathcal{R}_{\mathbb{VN}_\preccurlyeq} = \mathcal{R}_{\mathbb{V}} \cup \{R_N\} & \mathcal{R}_{\mathbb{VN}_{\Box\!\Rightarrow}} = \mathcal{R}_{\mathbb{V}_{\Box\!\Rightarrow}} \cup \{R'_N\}\\
\mathcal{R}_{\mathbb{VT}_\preccurlyeq} = \mathcal{R}_{\mathbb{V}} \cup \{R_T\} & \mathcal{R}_{\mathbb{VT}_{\Box\!\Rightarrow}} = \mathcal{R}_{\mathbb{V}_{\Box\!\Rightarrow}} \cup \{R'_T\}\\
\mathcal{R}_{\mathbb{VW}_\preccurlyeq} = \mathcal{R}_{\mathbb{VT}} \cup \{W_{n,m} \mid n \ge 1, m \ge 0\} & \mathcal{R}_{\mathbb{VW}_{\Box\!\Rightarrow}} = \mathcal{R}_{\mathbb{VT}_{\Box\!\Rightarrow}} \cup \{W'_{n,m} \mid n \ge 1, m \ge 0\}\\
\mathcal{R}_{\mathbb{VC}_\preccurlyeq} = \mathcal{R}_{\mathbb{V}} \cup \{R_{C1}, R_{C2}\} & \mathcal{R}_{\mathbb{VC}_{\Box\!\Rightarrow}} = \mathcal{R}_{\mathbb{V}_{\Box\!\Rightarrow}} \cup \{R'_{C1}, R'_{C2}\}
\end{array}$$

Fig. 2. The rules and rule sets

As usual, we call a formula *principal* in a rule if it appears in the conclusion of the rule but not in any premiss. A premiss of a rule is *contextual* if it inherits the context (written $\Gamma \Rightarrow \Delta$ in Figure 2) from the conclusion. That is, the right hand premiss of R_T and the premisses of both R_{C_1} and R_{C_2} are contextual premisses of the respective rules. If $\mathcal{R}$ is one of the rule sets of Figure 2, we write $\mathcal{R}^*$ for the rule set that arises by adding the principal formulae of each rule to each of its contextual premisses and refer to $\mathcal{R}^*$ as the *modification* of $\mathcal{R}$. For example, the right (contextual) premiss of the rule R_T then becomes $\Gamma,(A \preccurlyeq B) \Rightarrow \Delta, B$ whereas the left (non-contextual) premiss of R_T remains unchanged. We write $\vdash_{\mathcal{R}} \Gamma \Rightarrow \Delta$ in case $\Gamma \Rightarrow \Delta$ is derivable using rules in $\mathcal{R}$, and $\vdash_{\mathcal{R}^*}$ for derivability using the modification of $\mathcal{R}$. We denote use of additional rules by juxtaposition, e.g. $\mathsf{G}\mathcal{R}\mathsf{ConCut}$ denotes derivability where Cut and Contraction (both on the left and on the right) may be used in addition to rules in G and $\mathcal{R}$. The remainder of the paper establishes our main contributions, the first being soundness and completeness of the corresponding rules in presence of contraction (see Sections 4,5,6).

Theorem 1 (Soundness and Completeness). *If* $* \in \{\preccurlyeq, \Box\!\!\Rightarrow\}$ *and* $\mathcal{L}$ *is one of the logics* $\mathbb{V}_*, \mathbb{VN}_*, \mathbb{VT}_*, \mathbb{VW}_*, \mathbb{VC}_*$ *then* $\vdash_{\mathsf{G}\mathcal{R}_{\mathcal{L}}\mathsf{Con}} \Gamma \Rightarrow \Delta$ *if and only if* $\models_{\mathcal{L}} \Gamma \Rightarrow \Delta$.

The primary purpose of the modifications of the rules in Figure 2 is to achieve admissibility of contraction between principal formulae and those in the context. It is easy to see that this does not change the set of derivable sequents (Sections 5,6).

Proposition 2 (Elimination of Contraction). *If* $* \in \{\preccurlyeq, \Box\!\!\Rightarrow\}$ *and* $\mathcal{L}$ *is one of the logics* $\mathbb{V}_*, \mathbb{VN}_*, \mathbb{VT}_*, \mathbb{VW}_*, \mathbb{VC}_*$ *then* $\vdash_{(\mathsf{G}\mathcal{R}_\mathcal{L})^*} \Gamma \Rightarrow \Delta$ *if and only if* $\vdash_{\mathsf{G}\mathcal{R}_\mathcal{L}\mathsf{Con}} \Gamma \Rightarrow \Delta$.

This already implies that cut elimination holds for all logics formulated in terms of entrenchment and strong conditional. The calculi are complexity optimal (Sections 5,6):

Theorem 3 (Complexity). *If* $* \in \{\preccurlyeq, \Box\!\!\Rightarrow\}$ *and* $\mathcal{L}$ *is one of the logics* $\mathbb{V}_*, \mathbb{VN}_*, \mathbb{VT}_*, \mathbb{VW}_*, \mathbb{VC}_*$, *then derivability in* $(\mathsf{G}\mathcal{R}_\mathcal{L})^*$ *is decidable in* PSPACE *using backwards proof search. If* $* = \Box\!\!\rightarrow$, *then* $\mathcal{L}$ *is decidable in* PSPACE *by translating to* $\Box\!\!\Rightarrow$.

As an immediate application, the calculi above allow us to establish, for the first time, that all logics considered here have the Craig interpolation property (Section 7).

Theorem 4 (Craig Interpolation). *If* $* \in \{\preccurlyeq, \Box\!\!\Rightarrow, \Box\!\!\rightarrow\}$ *and* $\mathcal{L}$ *is one of the logics* $\mathbb{V}_*, \mathbb{VN}_*, \mathbb{VT}_*, \mathbb{VW}_*, \mathbb{VC}_*$, *then* $\mathcal{L}$ *has the Craig interpolation property.*

We prove the above theorems and give precise definitions in the following sections.

4 Soundness and Completeness of the Entrenchment Rules

We first consider the rules in the entrenchment language. The corresponding results for the rules for the strong implication will be established in Section 6.

Theorem 5. *For* $\mathcal{L} \in \{\mathbb{V}_\preccurlyeq, \mathbb{VN}_\preccurlyeq, \mathbb{VT}_\preccurlyeq, \mathbb{VW}_\preccurlyeq, \mathbb{VC}_\preccurlyeq\}$ *the rules in* $\mathcal{R}_\mathcal{L}$ *are sound for* $\mathcal{L}$.

Proof. We proceed by induction on the derivations and refer to [4] for $\mathcal{R}_{\mathbb{VC}_\preccurlyeq}$.
For $\mathcal{R}_{\mathbb{V}_\preccurlyeq}$: Suppose the last applied rule was $R_{n,m}$, with conclusion $(C_1 \preccurlyeq D_1), \dots, (C_m \preccurlyeq D_m) \Rightarrow (A_1 \preccurlyeq B_1), \dots, (A_n \preccurlyeq B_n)$ and premisses as given in Figure 2, and suppose all the premisses are valid. Let $\mathcal{I} = (I, (\$_i)_{i \in I}, \pi)$ be a sphere model and $i \in I$. Suppose $i \in [\![C_k \preccurlyeq D_k]\!]$ for all $k \in [m]$ and that for a $k \in [n]$ we have $i \notin [\![A_\ell \preccurlyeq B_\ell]\!]$ for all $\ell \in [n]$, $\ell \neq k$. Choose $S \in \$_i$ and $j \in S \cap [\![B_k]\!]$. Since $\models_{\mathbb{V}_\preccurlyeq} B_k \to \bigvee_{\ell \in [m]} D_\ell \vee \bigvee_{\ell \in [n]} A_\ell$ we have $j \in \bigcup_{\ell \in [n]} [\![A_\ell]\!] \cup \bigcup_{\ell \in [m]} [\![D_\ell]\!]$. Thus either $j \in \bigcup_{\ell \in [n]} [\![A_\ell]\!]$ or $j \in [\![D_\ell]\!]$ for a $\ell \in [m]$. In the latter case, since $i \in [\![C_\ell \preccurlyeq D_\ell]\!]$ we find a $j_2 \in S \cap [\![C_\ell]\!]$, and since $\models_{\mathbb{V}_\preccurlyeq} C_\ell \to \bigvee_{\ell' < \ell} D_{\ell'} \vee \bigvee_{\ell' \in [n]} A_{\ell'}$ we have $j_2 \in \bigcup_{\ell' < \ell} [\![D_{\ell'}]\!] \cup \bigcup_{\ell' \in [n]} [\![A_{\ell'}]\!]$. Continuing like this we find a $j' \in I$ with $j' \in S \cap \bigcup_{\ell \in [n]} [\![A_\ell]\!]$. Now if $j' \notin [\![A_k]\!]$ there is a $\ell \neq k$ with $j' \in [\![A_\ell]\!]$. But since $i \notin [\![A_\ell \preccurlyeq B_\ell]\!]$ there is an $S' \in \$_i$ with $S' \subsetneq S$ and $S' \cap [\![B_\ell]\!] \neq \emptyset = S' \cap [\![A_\ell]\!]$. As above we get a $j'' \in S' \cap \bigcup_{t \in [n]} [\![A_t]\!] = S' \cap \bigcup_{t \in [n], t \neq \ell} [\![A_t]\!]$. Repeating the argument we finally get an $S'' \in \$_i$ with $\emptyset \neq S'' \cap \bigcup_{\ell \in [n]} [\![A_\ell]\!] = S'' \cap [\![A_k]\!]$, and since by construction $S'' \subseteq S$ we have $i \in [\![A_k \preccurlyeq B_k]\!]$.

For $\mathcal{R}_{\mathbb{VN}_\preccurlyeq}$: Assume $\models_{\mathbb{VN}_\preccurlyeq} \neg A$ and $\models_{\mathbb{VN}_\preccurlyeq} B$, let $\mathcal{I}$ be a normal sphere model, i.e., for all $i \in I$ we have $\bigcup \$_i \neq \emptyset$, and let $i \in I$. Since $\bigcup \$_i \neq \emptyset$ there is a $j \in S \in \$_i$. But then $j \in [\![B]\!]$ and for all $t \in S$ we have $t \notin [\![A]\!]$. Thus $i \notin [\![A \preccurlyeq B]\!]$.

For $\mathcal{R}_{\mathbb{VT}_\preccurlyeq}$: Suppose $\models_{\mathbb{VT}_\preccurlyeq} \neg A$ and $\models_{\mathbb{VT}_\preccurlyeq} \bigwedge \Gamma \to \bigvee \Delta \vee B$, and let $\mathcal{I}$ be totally reflexive, i.e., for all $i \in I$ we have $i \in \bigcup \$_i$. Then for any $i \in I$ we have either $i \in [\![B]\!]$ and are done, or we can choose a $S \in \$_i$ with $i \in S$. But we know that $j \notin [\![A]\!]$ for all $j \in S$, and thus we get $i \notin [\![A \preccurlyeq B]\!]$.

For $\mathcal{R}_{\mathbb{VW}_{\preccurlyeq}}$: Similar to $\mathbb{V}_{\preccurlyeq}$. Let $\models_{\mathbb{VW}_{\preccurlyeq}} \Gamma \Rightarrow D_1, \dots, D_m, A_1, \dots, A_n, \Delta$ and $\models_{\mathbb{VW}_{\preccurlyeq}} C_k \Rightarrow D_1, \dots, D_{k-1}, A_1, \dots, A_n$ for all $k \in [m]$, and suppose that $\mathcal{I}$ is weakly centered, i.e., for all $i \in I$ there is an $S \in \$_i$ with $S \neq \emptyset$ and for all $S \in \$_i$ with $S \neq \emptyset$ we have $i \in S$. Then for $i \in I$ we have either $i \notin \bigcup_{\ell \in [m]} [\![D_\ell]\!] \cup \bigcup_{\ell \in [n]} [\![A_\ell]\!]$ and are done; or we have $i \in [\![A_\ell]\!]$ for a $\ell \in [n]$ and are done; or we have $i \in [\![D_k]\!]$ for a $k \in [m]$. In the latter case we take $S \in \$_i$ with $S \neq \emptyset$. Then $i \in S$. If $i \notin [\![C_k \preccurlyeq D_k]\!]$ we are done; otherwise there is a $i_1 \in S \cap C_k$. Since $\models_{\mathbb{VW}_{\preccurlyeq}} C_k \to \bigvee_{\ell < k} D_\ell \vee \bigvee_{\ell \in [n]} A_\ell$ we have $i_1 \in \bigcup_{\ell < k} [\![D_\ell]\!] \cup \bigcup_{\ell \in [n]} [\![A_\ell]\!]$. Repeating the argument yields a $j \in S \cap \bigcup_{\ell \in [n]} [\![A_\ell]\!]$. Choose k_1 with $j \in [\![A_{k_1}]\!]$. If $i \notin [\![A_{k_1} \preccurlyeq B_{k_1}]\!]$, then there is a $S' \subsetneq S$ with $S' \cap [\![A_{k_1}]\!] = \emptyset$ and $S' \cap [\![B_{k_1}]\!] \neq \emptyset$. As above we get a $j_2 \in S' \cap \bigcup_{\ell \in [n]} [\![A_\ell]\!] = S' \cap \bigcup_{\ell \neq k_1} [\![A_\ell]\!]$. Repeating the argument again we successively eliminate the A_ℓ's and get a $k' \in [n]$ such that for all $S \in \$_i$ with $S \cap [\![B_{k'}]\!] \neq \emptyset$ we have $S \cap [\![A_{k'}]\!] \neq \emptyset$. But this means $i \in [\![A_{k'} \preccurlyeq B_{k'}]\!]$. □

Next we establish completeness of the sequent systems with the cut rule. Cut-free completeness follows from the generic cut elimination result of the next section. Since all our systems include the congruence rule and thus are closed under uniform substitution, it suffices to show that all the rules and axioms of the Hilbert-style characterisation $\mathcal{HL}$ of a given logic $\mathcal{L}$ from Figure 1 are derivable in the corresponding sequent system with cut. Since the Hilbert-systems are complete [11], this establishes the result.

Theorem 6 (Completeness). *For* $\mathcal{L} \in \{\mathbb{V}_{\preccurlyeq}, \mathbb{VN}_{\preccurlyeq}, \mathbb{VT}_{\preccurlyeq}, \mathbb{VW}_{\preccurlyeq}, \mathbb{VC}_{\preccurlyeq}\}$ *the sequent system* $\mathsf{G}\mathcal{R}_{\mathcal{L}}\mathsf{ConWCut}$ *is complete with respect to* $\mathcal{L}$.

Proof. Showing that the rules and axioms of $\mathcal{H}\mathbb{V}_{\preccurlyeq}, \mathcal{H}\mathbb{VN}_{\preccurlyeq}$ and $\mathcal{H}\mathbb{VT}_{\preccurlyeq}$ can be derived in the corresponding sequent system is easy. For $\mathcal{H}\mathbb{VW}_{\preccurlyeq}$ note that adding the axiom (W) is equivalent to adding the axioms (T) and $(\neg A \preccurlyeq \top) \vee A$, where the latter is easily derived using $W_{1,0}$. For $\mathcal{H}\mathbb{VC}_{\preccurlyeq}$, using R_{C2} we get $(A \preccurlyeq \top) \to A$ and thus (C). □

5 Cut Elimination for the Entrenchment Rules

Our approach towards proving cut elimination for the sequent systems of the previous section is based on a general method for the construction of cut-free calculi: *cut elimination by saturation*. We call a set of (sequent) rules *saturated* if it is closed under the operations of cut and contraction, introduced below. Cut elimination by saturation elevates both cut and contraction from the level of *proof rules* to the level of *operations* on proof rules, i.e. constructions that allow us to derive new proof rules while preserving soundness. Cut closure holds if for any two given rules, performing a cut on the conclusions and collecting the premisses of both rules results in a (cut-free) derivable rule (after eliminating variables that no longer occur in the conclusion) and contraction closure stipulates that the result of identifying literals in the conclusion of a rule gives a rule already present in the rule set. Assuming saturation cut elimination holds, every cut can be replaced by a derivable rule, reducing level or rank of the cuts. The key ingredient in sequent systems for non-iterative logics is the concept of a *shallow rule*, introduced in previous work [10]. Intuitively, a shallow rule adds one layer of modalities in the conclusion, while its premisses may or may not propagate the context.

Definition 7. A *shallow rule* is a triple $R = (P_n; P_c; \Sigma \Rightarrow \Pi)$ where $P_n \subseteq \mathcal{S}(\mathcal{V})$ and $P_c \subseteq \mathcal{S}(\mathcal{V})$ are finite sets of sequents (the *non-contextual* and *contextual* premisses, respectively) and $\Sigma \Rightarrow \Pi \in \mathcal{S}(\Lambda(\mathcal{V}))$ are the *principal formulae* subject to the following variable restriction: every variable $p \in \mathcal{V}$ may occur at most once in $\Sigma \Rightarrow \Pi$ and occurs in the premisses iff it occurs in the principal formulae. An *instance* of a shallow rule

$$\frac{\{\Upsilon\sigma \Rightarrow \Omega\sigma \mid \Upsilon \Rightarrow \Omega \in P_n\} \quad \cup \quad \{\Gamma, \Theta\sigma \Rightarrow \Delta, \Xi\sigma \mid \Theta \Rightarrow \Xi \in P_c\}}{\Gamma, \Sigma\sigma \Rightarrow \Delta, \Pi\sigma}$$

is given by a context $\Gamma \Rightarrow \Delta$ and a substitution $\sigma : \mathcal{V} \to \mathcal{F}(\Lambda)$. We often annotate the contextual premisses with the context (usually $\Gamma \Rightarrow \Delta$) if no confusion can arise.

Remark 8. The variable restriction on the principal formulae is for technical convenience and not restrictive, as a duplicate occurrence of a variable p is avoided by replacing it by a fresh variable q and adding non-contextual premisses $p \Rightarrow q$ and $q \Rightarrow p$.

Example 9. The rules of classical propositional logic such as $\frac{\Gamma \Rightarrow \Delta, A \quad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \wedge B}$ ($\wedge$R) and all of our rules for conditional logics in Figure 2 are shallow. All premisses in $\wedge$R and R_{C2} are contextual, while all premisses in $R_{n,m}$ and R_N are non-contextual. Rule R_T has both a contextual and a non-contextual premiss.

A set $\mathcal{R}$ of shallow rules induces a sequent system in the standard way.

Definition and Convention 10. Whenever we speak about a *set of shallow rules* $\mathcal{R}$ we assume that $\mathcal{R}$ is closed under injective renaming of propositional variables. Let $\mathcal{R}$ be a set of shallow rules and $S \subseteq \mathcal{S}(\mathcal{F}(\Lambda))$ a set of sequents. A sequent $\Gamma \Rightarrow \Delta$ is *$\mathcal{R}$-derivable from S*, in symbols $S \vdash_{\mathcal{R}} \Gamma \Rightarrow \Delta$, if it is an element of the least set $S \vdash_{\mathcal{R}}$ containing S and closed under the axiom rules $\overline{\Gamma, A \Rightarrow \Delta, A}$ and the *congruence rules* $\frac{A_1 \Rightarrow B_1 \quad B_1 \Rightarrow A_1 \quad \dots \quad A_n \Rightarrow B_n \quad B_n \Rightarrow A_n}{\Gamma, \heartsuit(A_1,\dots,A_n) \Rightarrow \Delta, \heartsuit(B_1,\dots,B_n)}$ and all instances of rules in $\mathcal{R}$. We write $S \vdash_{\mathcal{R}\mathcal{R}'}$ for $S \vdash_{\mathcal{R}\cup\mathcal{R}'}$ and simply $\vdash_{\mathcal{R}}$ for $\emptyset \vdash_{\mathcal{R}}$. The rule set $\mathcal{R}'$ is *$\mathcal{R}$-admissible* if $\vdash_{\mathcal{R}\mathcal{R}'} \subseteq \vdash_{\mathcal{R}}$. *Derivations* are defined as usual [18] and a (not necessarily shallow) rule $R = P_1 \dots P_n / C$ with premisses $P_1, \dots, P_n$ and conclusion C is *$\mathcal{R}$-derivable* if $\{P_1, \dots, P_n\} \vdash_{\mathcal{R}} C$.

Lemma 11 (Admissibility of Weakening). *$\vdash_{\mathcal{R}} \Gamma \Rightarrow \Delta$ whenever $\vdash_{\mathcal{R}\mathsf{W}} \Gamma \Rightarrow \Delta$.*

The proof is standard. For admissibility of Contraction and Cut, the rule set needs to be closed under the operations of *rule contraction* and *cut between rules* described next.

Definition 12 (Cut as an Operation on Proof Rules). If (O_n, O_c) are sets of sequents (that we think of as non-contextual and contextual premisses, respectively) and p is a variable, then the *p-elimination on O_n and O_c* is the pair $(O_n, O_c) \ominus p := (O'_n, O'_c)$ where

$$\begin{aligned} O'_n &= \{\Gamma, \Sigma \Rightarrow \Delta, \Pi \mid \langle \Gamma, p \Rightarrow \Delta; \Sigma \Rightarrow \Pi, p\rangle \in O_n \times O_n\} \cup \{\Gamma \Rightarrow \Delta \in O_n \mid p \notin \Gamma, \Delta\} \\ O'_c &= \{\Gamma, \Sigma \Rightarrow \Delta, \Pi \mid \langle \Gamma, p \Rightarrow \Delta; \Sigma \Rightarrow \Pi, p\rangle \in (O_n \cup O_c)^2 \setminus (O_n \times O_n)\} \\ &\quad \cup \{\Gamma \Rightarrow \Delta \in O_c \mid p \notin \Gamma \cup \Delta\} \end{aligned}$$

and we write $(O_n, O_c) \ominus p_1, \ldots, p_n$ for the repeated application of variable elimination. If $R = (P_n; P_c; \Sigma \Rightarrow \Pi, \heartsuit \boldsymbol{p})$ and $R' = (P'_n; P'_c; \heartsuit \boldsymbol{p}, \Sigma' \Rightarrow \Pi')$ are shallow rules, the *cut of R and R′ on* $\heartsuit \boldsymbol{p}$ is the shallow rule $\mathsf{cut}(R, R', \heartsuit \boldsymbol{p}) = (Q_n; Q_c; \Sigma, \Sigma' \Rightarrow \Pi, \Pi')$ where $(Q_n, Q_c) = (P_n \cup P'_n, P_c \cup P'_c) \ominus \boldsymbol{p}$. A set $\mathcal{R}$ of shallow rules is *cut closed* if for any $R, R' \in \mathcal{R}$ with principal formulae $\Gamma \Rightarrow \Delta, \heartsuit \boldsymbol{p}$ and $\heartsuit \boldsymbol{p}, \Gamma' \Rightarrow \Delta'$ the rule $\mathsf{cut}(R, R', \heartsuit \boldsymbol{p})$ is $\mathcal{R}\mathsf{WCon}$-derivable.

That is, the cut between R and R' is a (shallow) rule, whose principal formulae arise by applying cut to the principal formulae of R and R' and whose premisses are the premisses of both R and R' with superfluous variables eliminated by variable elimination, i.e. cuts on the variables that no longer appear in the conclusion. Note that a premiss is contextual in the cut between two rules if at least one step in the variable elimination process did involve a contextual premiss. Cut closed rule sets are simply closed under performing cuts between rules. Also note that in presence of the rules for classical propositional logic the constructed rules are derivable using the old rules and Cut, since we can reconstruct the cut formulae for the premisses using the rules from G:

Lemma 13 ([10]). *For shallow rules* R_1, R_2 *with principal formulae* $\Sigma \Rightarrow \Pi, \heartsuit \boldsymbol{p}$ *and* $\heartsuit \boldsymbol{p}, \Sigma' \Rightarrow \Pi'$ *the rule* $\mathsf{cut}(R_1, R_2, \heartsuit \boldsymbol{p})$ *is* $\mathsf{G}R_1R_2\mathsf{Cut}$*-derivable.*

A similar construction applies to contraction:

Definition 14 (Contraction as Operation on Proof Rules). If S is a set of sequents and $\boldsymbol{p} = (p_1, \ldots, p_n)$ and $\boldsymbol{q} = (q_1, \ldots, q_n)$ are n-tuples of variables, then $S[\boldsymbol{q} \leftarrow \boldsymbol{p}]$ is the result of replacing every occurrence of q_i in a sequent in S by p_i for all $i = 1, \ldots, n$ and contracting duplicate instances of $p_1, \ldots, p_n$. Let $R = (P_n; P_c; \Gamma, \heartsuit \boldsymbol{p}, \heartsuit \boldsymbol{q} \Rightarrow \Delta)$ be a shallow rule. The *left contraction of R on* $\heartsuit \boldsymbol{p}$ *and* $\heartsuit \boldsymbol{q}$ is the shallow rule $\mathsf{ConL}(R, \heartsuit \boldsymbol{p}, \heartsuit \boldsymbol{q}) = (P_n[\boldsymbol{q} \leftarrow \boldsymbol{p}]; P_c[\boldsymbol{q} \leftarrow \boldsymbol{p}]); \Gamma, \heartsuit \boldsymbol{p} \Rightarrow \Delta)$. The *right contraction* $\mathsf{ConR}(R, \heartsuit \boldsymbol{p}, \heartsuit \boldsymbol{q})$ is defined dually. A rule set $\mathcal{R}$ is *contraction closed* if for every rule $R \in \mathcal{R}$ the rules $\mathsf{ConL}(R, \heartsuit \boldsymbol{p}, \heartsuit \boldsymbol{q})$ and $\mathsf{ConR}(R, \heartsuit \boldsymbol{p}, \heartsuit \boldsymbol{q})$ can be simulated by applications of Weakening and Contraction, followed by at most one application of a rule $R' \in \mathcal{R}$ and Weakening.

Saturated rule sets combine both properties.

Definition 15. A set of shallow rules is *saturated* if it is both cut and contraction closed.

Theorem 16. *For* $\mathcal{L} \in \{\mathbb{V}_{\preccurlyeq}, \mathbb{VN}_{\preccurlyeq}, \mathbb{VT}_{\preccurlyeq}, \mathbb{VW}_{\preccurlyeq}, \mathbb{VC}_{\preccurlyeq}\}$ *the rule set* $\mathsf{G}\mathcal{R}_{\mathcal{L}}$ *is saturated.*

Proof (Sketch). It is easy to see that the rules of G are saturated. Since cuts between propositional and conditional rules on principal formulae of both rules are impossible we thus only need to consider the rule sets $\mathcal{R}_{\mathcal{L}}$. For cut closure of $\mathcal{R}_{\mathbb{V}_{\preccurlyeq}}$ it can be seen that cuts between two rules $R_{n,m}$ and $R_{k,\ell}$ are subsumed by the rule $R_{n+k-1,m+\ell-1}$. Contraction closure is evident. The sets $\mathcal{R}_{\mathbb{VN}_{\preccurlyeq}}$ and $\mathcal{R}_{\mathbb{VT}_{\preccurlyeq}}$ are cut- and contraction closed, since cuts between a rule $R_{n,m}$ and R_N or R_T are subsumed by the rule $R_{n-1,m}$. Cut- and contraction closure of $\mathcal{R}_{\mathbb{VW}_{\preccurlyeq}}$ follows since $\mathcal{R}_{\mathbb{V}_{\preccurlyeq}}$ is cut closed and since cuts between $R_{n,m}$ or $W_{n,m}$ and $W_{k,\ell}$ are subsumed by $W_{n+k-1,m+\ell-1}$. For $\mathcal{R}_{\mathbb{VC}_{\preccurlyeq}}$ note that cuts between $R_{n,m}$ and R_{C1} or R_{C2} can be replaced by a number of applications of R_{C2} and R_{C1}. □

Saturation enables a general cut elimination theorem following [5]: (multi-)cuts on context formulae are propagated upwards in the proof trees, and (multi-)cuts on principal formulae can be eliminated using cut and contraction closure.

Theorem 17 (Generic Cut Elimination). *Let $\mathcal{R}$ be a saturated set of shallow rules. Then* Cut *is admissible in* $\mathcal{R}\mathsf{Con}$*, i.e.* $\vdash_{\mathcal{R}\mathsf{Con}} \Gamma \Rightarrow \Delta$ *whenever* $\vdash_{\mathcal{R}\mathsf{ConCut}} \Gamma \Rightarrow \Delta$*.*

Proof. Similar to [10, Prop. 21].

Corollary 18. *For* $\mathcal{L} \in \{\mathbb{V}_{\preccurlyeq}, \mathbb{VN}_{\preccurlyeq}, \mathbb{VT}_{\preccurlyeq}, \mathbb{VW}_{\preccurlyeq}, \mathbb{VC}_{\preccurlyeq}\}$ $\mathsf{G}\mathcal{R}_{\mathcal{L}}\mathsf{Con}$ *has cut elimination.*

Note that contraction closure only allows to eliminate Contraction on *principal formulae*, but not on a principal formula and a context formula. Nevertheless, after establishing cut elimination, admissibility of Contraction and a generic Pspace complexity result are obtained in the modification of the rule set, where in a standard move the principal formulae are copied into the contextual premisses.

Definition 19 (Modified Instances). A *modified instance*

$$\frac{\{\Upsilon\sigma \Rightarrow \Omega\sigma \mid \Upsilon \Rightarrow \Omega \in P_n\} \quad \cup \quad \{\Gamma, \Sigma\sigma, \Theta\sigma \Rightarrow \Delta, \Xi\sigma, \Pi\sigma \mid \Theta \Rightarrow \Xi \in P_c\}}{\Gamma, \Sigma\sigma \Rightarrow \Delta, \Pi\sigma}$$

of a shallow rule $(P_n; P_c; \Sigma \Rightarrow \Delta)$ is given by a substitution σ and a (context) sequent $\Gamma \Rightarrow \Delta$. For the *modification* $\mathcal{R}^*$ of $\mathcal{R}$ the notion of $\mathcal{R}^*$-admissibility and $\mathcal{R}^*$-derivability are as for $\mathcal{R}$ using modified instances of rules in $\mathcal{R}$ instead of instances.

The purpose of modified instances is the elimination of Contraction, where Contraction between context and principal formulae is absorbed by moving principal formulae upwards in the context. Moving to modified instances, e.g. the (standard) instance $\frac{\Gamma,\Theta\Rightarrow\Delta,\Xi \quad \Upsilon\Rightarrow\Omega}{\Gamma,\heartsuit A\Rightarrow\Delta,\clubsuit B}$ is replaced by the modified instance $\frac{\Gamma,\heartsuit A,\Theta\Rightarrow\Delta,\clubsuit B,\Xi, \quad \Upsilon\Rightarrow\Omega}{\Gamma,\heartsuit A\Rightarrow\Delta,\clubsuit B}$. We can now apply the following result from [10] for *tractable* rule sets, i.e., sets where codes of the rules can be computed in space polynomial in the length of the conclusion and where the premisses can be computed in space polynomial in the code of the rule. It can easily be checked that all of the rule sets in Figure 2 as well as the rules of G are tractable.

Theorem 20. *If* $\mathcal{R}$ *is saturated, then* $\vdash_{\mathcal{R}\mathsf{ConCut}} = \vdash_{\mathcal{R}\mathsf{Con}} = \vdash_{\mathcal{R}^*\mathsf{Con}} = \vdash_{\mathcal{R}^*}$*. In particular,* Con *is* $\mathcal{R}^*$*-admissible. If* $\mathcal{R}$ *is also tractable, then backwards proof search in* $\mathcal{R}^*$ *is in* Pspace.

Corollary 21. *For* $\mathcal{L} \in \{\mathbb{V}_{\preccurlyeq}, \mathbb{VN}_{\preccurlyeq}, \mathbb{VT}_{\preccurlyeq}, \mathbb{VW}_{\preccurlyeq}, \mathbb{VC}_{\preccurlyeq}\}$ *we have* $\models_{\mathcal{L}} = \vdash_{(\mathsf{G}\mathcal{R}_{\mathcal{L}})^*}$ *and backwards proof search in* $(\mathsf{G}\mathcal{R}_{\mathcal{L}})^*$ *is in* Pspace.

Remark 22. Theorems 17 and 20 remain valid for languages that do not contain all Boolean connectives. As the propositional rules are shallow, they can be absorbed into the general treatment and it is easy to see that, for every Boolean connective, adding the corresponding left and right rules preserves saturation.

6 Strong and Weak Conditional Implication

For the systems in the language with the strong conditional our strategy for proving soundness and completeness is slightly different.

Theorem 23. *For* $\mathcal{L} \in \{\mathbb{V}_{\boxRight}, \mathbb{VN}_{\boxRight}, \mathbb{VT}_{\boxRight}, \mathbb{VW}_{\boxRight}, \mathbb{VC}_{\boxRight}\}$ *the sequent system* $\mathsf{G}\mathcal{R}_{\mathcal{L}}\mathsf{ConCut}$ *is sound and complete for* $\mathcal{L}$*.*

Proof. Since the strong conditional is defined in terms of entrenchment by the translation $(A \boxRightarrow B) \leftrightarrow \neg((A \wedge \neg B) \preccurlyeq (A \wedge B))$ from [11], we get the translation rules

$$\frac{A \Rightarrow C \quad A, D \Rightarrow \quad C \Rightarrow A, D \quad B \Rightarrow C \quad B \Rightarrow D \quad C, D \Rightarrow B}{\Gamma, (A \preccurlyeq B), (C \boxRightarrow D) \Rightarrow \Delta}\, R_{t1}$$

$$\frac{A \Rightarrow C \quad A, D \Rightarrow \quad C \Rightarrow A, D \quad B \Rightarrow C \quad B \Rightarrow D \quad C, D \Rightarrow B}{\Gamma \Rightarrow \Delta, (A \preccurlyeq B), (C \boxRightarrow D)}\, R_{t2}$$

which together are equivalent to the translation axiom. The rule sets $\mathcal{R}_{\mathbb{V}_{\boxRightarrow}}$, $\mathcal{R}_{\mathbb{VN}_{\boxRightarrow}}$, $\mathcal{R}_{\mathbb{VT}_{\boxRightarrow}}$, $\mathcal{R}_{\mathbb{VW}_{\boxRightarrow}}$ and $\mathcal{R}_{\mathbb{VC}_{\boxRightarrow}}$ arise from the rule sets for the entrenchment connective by cutting every literal of every rule with the appropriate translation rule. The resulting rules have the translation built in which gives completeness and soundness (using Lemma 13). □

Since cuts between the translation rules are subsumed by congruence and since the entrenchment rules are saturated, saturation for these rule sets is not unexpected.

Theorem 24. *For* $\mathcal{L} \in \{\mathbb{V}_{\boxRightarrow}, \mathbb{VN}_{\boxRightarrow}, \mathbb{VT}_{\boxRightarrow}, \mathbb{VW}_{\boxRightarrow}, \mathbb{VC}_{\boxRightarrow}\}$ *the rule set* $\mathsf{G}\mathcal{R}_{\mathcal{L}}$ *is saturated and thus* $\mathsf{G}\mathcal{R}_{\mathcal{L}}\mathsf{Con}$ *has cut elimination. Hence* $\mathcal{L}$ *is decidable in* PSPACE.

Proof. Cut closure is seen analogous to the entrenchment case. E.g. for $\mathcal{R}_{\mathbb{V}_{\boxRightarrow}}$ a cut between rules $R'_{n,m}$ and $R'_{k,\ell}$ is subsumed by the rule $R'_{n+k-1,\, m+\ell-1}$. Note that for some of the premisses of the latter rule we need to cut three of the original premisses and apply Contraction. Contraction closure is straightforward. □

Unfortunately, this technique does not work not work for Lewis' weak conditional $\boxright$, since the translations of $\boxright$ into $\preccurlyeq$ or $\boxRightarrow$ are more subtle. Nevertheless, since the translation $(A \boxright B) \leftrightarrow ((\bot \preccurlyeq A) \vee \neg((A \wedge \neg B) \preccurlyeq (A \wedge B)))$ of $\boxright$ into $\preccurlyeq$ from [11, p.26,53] increases the number of subformulae only by a constant factor, we may represent formulae as directed acyclic graphs instead of trees, to obtain a purely syntactical PSPACE decision procedure of optimal complexity for these logics in the language with $\boxright$.

Theorem 25. *The logics* $\mathbb{V}_{\boxright}, \mathbb{VN}_{\boxright}, \mathbb{VT}_{\boxright}, \mathbb{VW}_{\boxright}, \mathbb{VC}_{\boxright}$ *are decidable in* PSPACE.

Proof. Since the important measure for the backwards proof search procedure from [10] is the nesting depth of connectives and not the size of the formulae, careful inspection of the proofs together with the fact that the translation is linear for the representation of formulae by directed acyclic graphs yields the result. □

7 Applications

Interpolation. The sequent systems presented above enable us to establish Theorem 4 (Craig interpolation) for all logics considered in this paper in a standard way. A logic $\mathcal{L}$ has the *(Craig) interpolation property* (CIP) if whenever $\models_{\mathcal{L}} A \to B$, then there is an *interpolant* I satisfying the *variable condition* $\mathsf{var}(I) \subseteq \mathsf{var}(A) \cap \mathsf{var}(B)$ such that $\models_{\mathcal{L}} A \to I$ and $\models_{\mathcal{L}} I \to B$. We use split sequents [18] to establish the CIP, the intuition being that whenever we split a provable sequent into two, we can find an interpolant:

Definition 26 (split sequent). An expression $\Gamma_1 \mid \Gamma_2 \Rightarrow \Delta_1 \mid \Delta_2$ is a *split sequent*, if $\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2$ is a sequent, and we say that $\Gamma_1 \mid \Gamma_2 \Rightarrow \Delta_1 \mid \Delta_2$ is a *splitting* of $\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2$. A formula I is an *interpolant* in $\mathcal{R}_\mathcal{L}$ for the split sequent $\Gamma_1 \mid \Gamma_2 \Rightarrow \Delta_1 \mid \Delta_2$ if it satisfies the *variable condition* $\mathsf{var}(I) \subseteq \mathsf{var}(\Gamma_1 \Rightarrow \Delta_1) \cap \mathsf{var}(\Gamma_2 \Rightarrow \Delta_2)$ and $\vdash_{\mathcal{R}_\mathcal{L}} \Gamma_1 \Rightarrow \Delta_1, I$ and $\vdash_{\mathcal{R}_\mathcal{L}} I, \Gamma_2 \Rightarrow \Delta_2$. A sequent $\Gamma \Rightarrow \Delta$ *admits interpolation* in $\mathcal{R}_\mathcal{L}$ if all its splittings have an interpolant in $\mathcal{R}_\mathcal{L}$. A shallow rule R *supports interpolation* in $\mathcal{R}_\mathcal{L}$ if whenever all its premisses admit interpolation in $\mathcal{R}_\mathcal{L}$, then so does its conclusion.

It is routine to prove the following lemma by induction.

Lemma 27. *If $\mathsf{G}\mathcal{R}_\mathcal{L}$ is a sound and complete set of shallow rules for a logic $\mathcal{L}$ and all the rules in $\mathsf{G}\mathcal{R}_\mathcal{L}$ support interpolation in $\mathsf{G}\mathcal{R}_\mathcal{L}$, then $\mathcal{L}$ has the interpolation property.*

Theorem 28. *$\mathbb{V}_\preccurlyeq$ has the Craig interpolation property.*

Proof. We need to show that the rules in $\mathsf{G}\mathcal{R}_{\mathbb{V}_\preccurlyeq}$ support interpolation. For the propositional rules this is standard [18]. For $R_{n,m}$ we construct an interpolant for a splitting of the conclusion from interpolants of the corresponding splittings of the premisses. First, consider the rule $R_{2,m}$ and the splitting $\Gamma_1 \mid \Gamma_2 \Rightarrow \Delta_1 \mid \Delta_2$ of its conclusion given by

$$\{(C_i \preccurlyeq D_i) \mid i \in [m], i \text{ odd}\} \mid \{(C_i \preccurlyeq D_i) \mid i \in [n], i \text{ even}\} \Rightarrow (A_1 \preccurlyeq B_1) \mid (A_2 \preccurlyeq B_2)\,.$$

For $k \in [m]$ let I_k be the interpolant for the corresponding splitting of the premiss $C_k \Rightarrow \{D_\ell \mid \ell < k\}, A_1, A_2$ and for $k \in \{1,2\}$ let J_k be the one for the corresponding splitting of the premiss $B_k \Rightarrow \{D_\ell \mid \ell \in [m]\}, A_1, A_2$. For every odd $k \in [m]$ we introduce

$$X_k = \bigvee_{\ell \leq k, \ell \text{ odd}} I_\ell \qquad Y_k = \begin{cases} \neg I_{k+1} \vee \neg J_2 & k = \max\{\ell \in [m] \mid \ell \text{odd}\} \\ \neg I_{k+1} & \text{otherwise} \end{cases} \qquad Z_k = J_1 \vee \bigvee_{\ell \in [m], \ell > k, \ell \text{ odd}} I_\ell$$

$$V_k = (X_k \preccurlyeq Y_k) \qquad W_k = (Y_k \preccurlyeq Z_k) \qquad I = \bigwedge_{k \in [m], k \text{ odd}} (\neg W_k \vee V_k)\,.$$

Claim 1: For every odd $k \in [m]$ we have $\vdash_{\mathcal{R}_{\mathbb{V}_\preccurlyeq}} \Gamma_1, W_k \Rightarrow \Delta_1, V_k$. The idea is to insert W_k instead of $(C_{k+1} \preccurlyeq D_{k+1})$ and V_k instead of $(A_2 \preccurlyeq B_2)$ into the rule pattern. Then using the definitions of W_k and V_k together with applications of Weakening it is straightforward to check that $R_{2,|\{\ell \in [m] | \ell \text{ odd}\}|+1}$ can be applied.
Claim 2: For every partition (F, S) of $\{k \in [m] \mid k \text{ odd}\}$ we have $\vdash_{\mathcal{R}_{\mathbb{V}_\preccurlyeq}} \Gamma_2, \{V_k \mid k \in F\} \Rightarrow \Delta_2, \{W_k \mid k \in S\}$. The idea is to insert the V_k instead of the $(C_k \preccurlyeq D_k)$, and the W_k as positive literals instead of $(A_1 \preccurlyeq B_1)$. Then again it is straightforward to check that we have all the necessary premisses for an application of $R_{|S|+1, |F|+|\{\ell \in [m] | \ell \text{ even}\}|}$.
By propositional reasoning, both claims give $\vdash_{\mathcal{R}_{\mathbb{V}_\preccurlyeq}} \Gamma_1 \Rightarrow \Delta_1, I$ and $\vdash_{\mathcal{R}_{\mathbb{V}_\preccurlyeq}} I, \Gamma_2 \Rightarrow \Delta_2$ and the interpolant I satisfies the variable condition, since all its constituents satisfy it.
For the general case consider the splitting $\Gamma_1 \mid \Gamma_2 \Rightarrow \Delta_1 \mid \Delta_2$ of the conclusion, and write I'_k for the interpolant for the corresponding splitting of the premiss $C_k \Rightarrow \{D_\ell \mid \ell < k\}, \{A_\ell \mid \ell \in [n]\}$ and J'_k for the one for the premiss $B_k \Rightarrow \{A_\ell \mid \ell \in [n]\}, \{D_\ell \mid \ell \in [m]\}$. In the construction of the interpolant above we replace J_1 by $\bigvee_{(A_\ell \preccurlyeq B_\ell) \in \Delta_1} J'_\ell$ and $\neg J_2$ by $\bigvee_{(A_\ell \preccurlyeq B_\ell) \in \Delta_2} \neg J'_\ell$. The formulae I_ℓ in X_k and Z_k are replaced by $\bigvee_{j \in T_\ell} I'_j$ where T_ℓ is the ℓ-th block of consecutive indices j with $(C_j \preccurlyeq D_j) \in \Gamma_1$. The formulae $\neg I_{k+1}$ in Y_k are replaced by $\bigvee_{j \in S_k} \neg I'_j$ where S_k is the k-th block of consecutive indices j with $(C_j \preccurlyeq D_j) \in \Gamma_2$. Then in the proofs of the claims the formulae W_k and V_k are inserted instead of the blocks $\{(C_\ell \preccurlyeq D_\ell) \mid \ell \in T_k\}$ and $\{(C_\ell \preccurlyeq D_\ell) \mid \ell \in S_k\}$. □

Corollary 29. $\mathbb{VT}_{\preccurlyeq}, \mathbb{VN}_{\preccurlyeq}, \mathbb{VW}_{\preccurlyeq}, \mathbb{VC}_{\preccurlyeq}$ *have the CIP.*

Proof. For $\mathbb{VT}_{\preccurlyeq}, \mathbb{VN}_{\preccurlyeq}$ and $\mathbb{VC}_{\preccurlyeq}$ this is immediate since the additional axioms trivially support interpolation. For the rules $W_{n,m}$ of $\mathcal{R}_{\mathbb{VW}_{\preccurlyeq}}$ we only need to modify the proof for the rules $R_{n,m}$ by replacing the interpolants J_1, J_2 in the construction of I by the interpolant J of the contextual premiss and its negation. □

Corollary 30. *For* $* \in \{\Box\!\!\Rightarrow, \Box\!\!\rightarrow\}$ *the logics* $\mathbb{V}_*, \mathbb{VT}_*, \mathbb{VN}_*, \mathbb{VW}_*, \mathbb{VC}_*$ *have the CIP.*

Proof. By translating the formula $A \to B$ into the entrenchment language, and translating the interpolant back into the original language. Since both translations are identity on propositional variables the variable condition holds, and we obtain an interpolant since translating back and forth yields logically equivalent formulae. □

Hybrid Conditional Logic. In [16] a hybridisation of conditional logic $\mathbb{V}_{\Box\!\Rightarrow}$ is proposed to extend Lewis' interpretation of $\Box\!\!\Rightarrow$ in terms of *contextually definite descriptions*. Worlds in a sphere model represent things or individuals, the sphere systems give degrees of salience, and a formula like $\mathsf{Pig} \Box\!\!\Rightarrow \mathsf{Grunting}$ is interpreted as "The (most salient) pig is grunting". *Nominals i* are introduced as names for specific individuals together with the satisfaction operators $@_i A$ stating that A is true for individual i.

Following [12] the sequent system for $\mathbb{V}_{\Box\!\Rightarrow}$ can easily be turned into a sequent system for the hybrid logic $\mathbb{V}_{\mathcal{HC}(@)}$ in the language of the strong conditional. Sphere models are captured coalgebraically as coalgebras for the functor Sp with $\mathsf{Sp}(X) = \{\$ \in \mathcal{PP}(X) \mid \$ \text{ a system of spheres}\}$ and $\mathsf{Sp}(f)$ the double direct image of f. The correct semantics for $\Box\!\!\Rightarrow$ is then given by the predicate lifting $[\![\Box\!\!\Rightarrow]\!]_X(A, B) = \{\$ \in \mathsf{Sp}(X) \mid \exists S \in \$ \text{ s.t. } (S \cap A \neq \emptyset \text{ and } S \cap A \cap B^c = \emptyset)\}$. Our proof of soundness and completeness for $\mathcal{R}_{\mathbb{V}_{\Box\Rightarrow}}$ over $\mathbb{V}_{\Box\!\Rightarrow}$ can be adapted to show that the rules are indeed one-step sound and cut-free complete with respect to the coalgebraic semantics. By [12] this induces a sequent system which is sound and complete with respect to $\mathbb{V}_{\mathcal{HC}(@)}$. In particular, backwards proof search in this system can be implemented in polynomial space.

Theorem 31. *Hybrid conditional logic* $\mathbb{V}_{\mathcal{HC}(@)}$ *is decidable in* PSPACE.

8 Conclusion

We presented the first unlabelled sequent systems for the conditional logics $\mathbb{V}, \mathbb{VN}, \mathbb{VT}$ and $\mathbb{VW}$ in the entrenchment and strong conditional languages and for $\mathbb{VC}$ in the strong conditional language. Since these systems have cut elimination and (after a slight modification) admissibility of contraction, backwards proof search can be implemented in polynomial space, giving the first purely syntactical PSPACE decision procedures for these logics. Furthermore, translating the weak conditional into our systems gives to our knowledge the first purely syntactical PSPACE decision procedures for the logics in the weak conditional language. All the algorithms are of optimal complexity. Moreover, we used our calculi to show that all the logics mentioned have the Craig interpolation property, and to give a PSPACE decision procedure for the hybrid version of $\mathbb{V}_{\Box\!\Rightarrow}$.

Related Work. Our calculus for $\mathbb{VC}_{\preccurlyeq}$ is the sequent version of the tableau calculus in [4,2], but we also systematically cover weaker logics and different languages. The calculi in [8] for the weak conditional language are labelled and thus conceptually more involved, and not complexity optimal. In [1] a system for $\mathbb{V}_{\Box\!\!\rightarrow}$ involving second degree sequents is given, but it is not used for deciding the logic. The complexity results in [3] are obtained via small model theorems which complements our purely syntactical treatment. Calculi for the flat fragments of conditional logics corresponding to logics of the KLM framework are given in [7].

References

1. Crocco, G., Fariñas del Cerro, L.: Structure, consequence relation and logic. In: Gabbay, D.M. (ed.) What is a logical system?, pp. 239–259. Oxford University Press (1994)
2. de Swart, H.C.: A Gentzen- or Beth-type system, a practical decision procedure and a constructive completeness proof for the counterfactual logics VC and VCS. J. Symb. Log. 48(1), 1–20 (1983)
3. Friedman, N., Halpern, J.Y.: On the complexity of conditional logics. In: KR 1994, pp. 202–213 (1994)
4. Gent, I.P.: A sequent- or tableau-style system for Lewis's counterfactual logic VC. Notre Dame J. Form. Log. 33(3), 369–382 (1992)
5. Gentzen, G.: Untersuchungen über das logische Schließen. I. Math. Z. 39(2), 176–210 (1934)
6. Ginsberg, M.L.: Counterfactuals. Artif. Intell. 30, 35–79 (1986)
7. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Analytic tableaux calculi for KLM logics of nonmonotonic reasoning. ACM Trans. Comput. Log. 10(3) (2009)
8. Giordano, L., Gliozzi, V., Olivetti, N., Schwind, C.: Tableau calculus for preference-based conditional logics: PCL and its extensions. ACM Trans. Comput. Log. 10(3), 1–50 (2009)
9. Groeneboer, C., Delgrande, J.P.: Tableau-based theorem proving in normal conditional logics. In: AAAI, pp. 171–176 (1988)
10. Lellmann, B., Pattinson, D.: Cut Elimination for Shallow Modal Logics. In: Brünnler, K., Metcalfe, G. (eds.) TABLEAUX 2011. LNCS, vol. 6793, pp. 211–225. Springer, Heidelberg (2011)
11. Lewis, D.: Counterfactuals. Blackwell (1973)
12. Myers, R., Pattinson, D., Schröder, L.: Coalgebraic Hybrid Logic. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 137–151. Springer, Heidelberg (2009)
13. Nute, D., Cross, C.B.: Conditional logic. In: Gabbay, D.M., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. 4, pp. 1–98. Kluwer (2001)
14. Olivetti, N., Pozzato, G.L., Schwind, C.: A sequent calculus and a theorem prover for standard conditional logics. ACM Trans. Comput. Log. 8(4), 1–51 (2007)
15. Pattinson, D., Schröder, L.: Generic modal cut elimination applied to conditional logics. Log. Methods Comput. Sci. 7(1) (2011)
16. Sano, K.: Hybrid counterfactual logics - David Lewis meets Arthur Prior again. J. Log. Lang. Inf. 18, 515–539 (2009)
17. Schröder, L., Pattinson, D., Hausmann, D.: Optimal tableaux for conditional logics with cautious monotonicity. In: ECAI, pp. 707–712 (2010)
18. Troelstra, A.S., Schwichtenberg, H.: Basic Proof Theory, 2nd edn. Cambridge University Press (2000)

Relevant Minimal Change in Belief Update

Laurent Perrussel[1], Jerusa Marchi[2],
Jean-Marc Thévenin[1], and Dongmo Zhang[3]

[1] Institut de Recherche en Informatique de Toulouse
Université Toulouse I, Toulouse, France
`laurent.perrussel@irit.fr, thevenin@univ-tlse1.fr`
[2] Departamento de Informática e Estatística,
Universidade Federal de Santa Catarina, Florianópolis, Brazil
`jerusa@inf.ufsc.br`
[3] School of Computing and Mathematics
University of Western Sydney, Sydney, Australia
`dongmo@scm.uws.edu.au`

Abstract. The notion of *relevance* was introduced by Parikh in the belief revision field for handling minimal change. It prevents the loss of beliefs that do not have connections with the epistemic input. But, the problem of minimal change and relevance is still an open issue in belief update. In this paper, a new framework for handling minimal change and relevance in the context of belief update is introduced. This framework goes beyond relevance in Parikh's sense and enforces minimal change by first rewriting the Katzuno-Mendelzon postulates for belief update and second by introducing a new relevance postulate. We show that *relevant minimal change* can be characterized by setting agent's preferences on beliefs where preferences are indexed by subsets of models of the belief set. Each subset represents a prime implicant of the belief set and thus stresses the key propositional symbols for representing the belief set.

1 Introduction

Belief updating is the process of incorporating new pieces of information into a set of existing beliefs when the world described by this set has changed. It is usually assumed that this operation follows two principles: (i) the resulting belief set is consistent, and (ii) the change to the original belief set is minimal.

The most influential work within the area is the KM paradigm, which characterizes the belief update operation through a set of plausible axioms, generally referred to as the KM postulates [7]. Despite their popularity, the KM postulates are not sufficient to capture minimal change.

The notion of relevant belief was introduced by Parikh [13] in the context of belief revision. Relevant belief revision ensures that all beliefs in an initial belief set that are not related with the new piece of information are preserved. This notion avoids counter-intuitive changes of beliefs like those performed by the full meet revision operator [1], i.e. removing all statements from the original belief set and keeping only the new piece of information. Relevant change has been investigated in the belief revision context [10,9,14,19]. However, relevance by its nature is a syntactical issue and model-based approaches provide only peripheral solutions. In this sense, approaches based on knowledge compilation [3] and also prime implicates and prime implicants representation have been proposed [4,15].

Particularly, in [15], we propose a relevant belief revision operator based on minimal change to general preference orderings via minimizing prime implicant changes to existing beliefs. This belief operator satisfies Katsuno-Mendelzon's postulates for belief revision as well as Parikh's postulate for relevant revision. However, that proposal was limited to the belief revision context.

The purpose of this paper is to extend our previous work in order to characterize the concept of *Relevant Belief Update*. Such characterization entails not only an adaptation of Parikh's postulate but also a new definition of the KM postulates for belief update in order to capture relevant minimal change. We consider that a belief update process should be performed over set of terms [16] instead of models by only looking at the literals that are concerned with the change issue. A natural way to focus on those literals is to represent the belief set as sets of prime implicants [11].

The paper is organized as follows. Section 2 reviews the notions of implicants and prime implicants and introduces some necessary definitions. Section 3 reviews the results obtained in [15], which are quite related to this work. Section 4 characterizes the class of *relevant minimal change* belief update operators in terms of postulates and constraints on preferences. Section 5 concludes the paper by considering some open issues.

2 Preliminaries

Let $P = \{p_1, \ldots, p_n\}$ be a finite set of propositional symbols and $\mathcal{L}$ be the propositional language associated with P. $\mathsf{Lang} : \mathcal{L} \mapsto 2^P$ is a function that assigns each formula φ in $\mathcal{L}$ the set of the propositional symbols occuring in φ.

Let $LIT = \{L_1, \ldots, L_{2n}\}$ be the set of associated literals: $L_i = p_j$ or $\neg p_j$. A *term* D_i is a *conjunction of literals*: $D_i = L_1 \wedge \cdots \wedge L_k$. Let $\overline{L_i}$ be the complementary literal, s.t. $\overline{L_i} = \neg p_j$ iff $L_i = p_j$ and $\overline{D}$ be the mirror of a term D s.t. $\overline{D} = \overline{L_1} \wedge \cdots \wedge \overline{L_k}$ iff $D = L_1 \wedge \cdots \wedge L_k$. In the following, terms can also be viewed as sets of literals ($D_i = \{L_1, \cdots, L_k\}$) and we will frequently switch between the two notations.

A term D is an *implicant* of an $\mathcal{L}$-formula ψ iff $D \models \psi$, where $\models$ is the satisfiability relation. A term D is said to be a *prime implicant* [17] of ψ if D is an implicant of ψ and for any term D' such that $D' \subset D$, we have $D' \not\models \psi$, i.e., a prime implicant of a formula ψ is an implicant of ψ without any subsumed terms.

Based on P and ψ, four specific sets of terms are considered:

1. $\mathcal{D}$ is the set of *all possible terms that can be built over P*. Since P is finite, $\mathcal{D}$ is also finite, because we only consider terms with non-redundant and non-contradicting literals;

2. PI_ψ is the set of *prime implicants of* ψ. This set is a disjunction of all non-contradictory and non-redundant prime implicants of ψ such that $\psi \equiv PI_\psi$. This set is unique and minimal in the sense that it consists of the smallest sets of terms closed for inference and without any subsumed terms;
3. $\mathcal{D}_\psi$ is the set of all *implicants of* ψ. This set is a disjunction of all non-contradictory and non-redundant implicants of ψ;
4. $\Gamma(\psi)$ is the set of *all possible terms based on* ψ defined as follows: for every $D_\psi \in PI_\psi$ and for every term $D \in \mathcal{D}$, a new term is obtained by adding to D all the literals of D_ψ which are non-conflicting with the literals of D. Formally:
$$\Gamma(\psi) = \{D \cup (D_\psi - \overline{D}) | D_\psi \in PI_\psi \text{ and } D \in \mathcal{D}\}$$

Figure 1 illustrates the inclusion relation between these sets: first prime implicants of ψ, then implicants of ψ, then terms that differ on some symbols with the implicants of ψ and finally all possible terms.

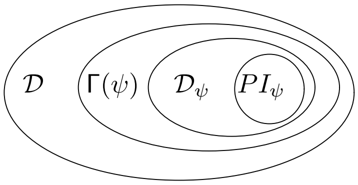


Fig. 1. Inclusion relation of sets of terms

In the sequel we omit "non-contradictory" and "non-redundant" when we mention prime implicants, implicants or terms.

Example 1. Consider that $P = \{p_1, p_2, p_3\}$ is the set of propositional symbols and a formula $\psi \in \mathcal{L}(P)$ such that $\psi = (p_1 \wedge p_2)$. The following sets of terms can be obtained from P and ψ:

$$\begin{aligned}
PI_\psi &= \{\{p_1, p_2\}\} \\
\mathcal{D}_\psi &= \{\{p_1, p_2\}, \{p_1, p_2, p_3\}, \{p_1, p_2, \neg p_3\}\} \\
\Gamma(\psi) &= \{\{p_1, p_2\}, \{\neg p_1, p_2\}, \{p_1, \neg p_2\}, \\
&\quad \{\neg p_1, \neg p_2\}\{p_1, p_2, p_3\}, \{p_1, p_2, \neg p_3\}, \{\neg p_1, p_2, p_3\}, \\
&\quad \{\neg p_1, p_2, \neg p_3\}, \{p_1, \neg p_2, p_3\}, \{p_1, \neg p_2, \neg p_3\} \\
&\quad \{\neg p_1, \neg p_2, p_3\}, \{\neg p_1, \neg p_2, \neg p_3\}\} \\
\mathcal{D} &= \{\{\}, \{p_1\}, \{p_2\}, \{p_3\}, \{\neg p_1\}, \{\neg p_2\}, \{\neg p_3\}, \\
&\quad \{p_1, p_2\}, \{p_1, \neg p_2\}, \{p_1, p_3\}, \cdots \{\neg p_2, \neg p_3\}, \\
&\quad \{p_1, p_2, p_3\}, \cdots, \{\neg p_1, \neg p_2, \neg p_3\}\}
\end{aligned}$$

The cardinalities of these sets are: $\mid PI_\psi \mid = 1$, $\mid \mathcal{D}_\psi \mid = 3$, $\mid \Gamma(\psi) \mid = 12$ and $\mid \mathcal{D} \mid = 27$. Let us stress that terms without the two propositional symbols involved in the prime implicants of ψ could not belong to $\Gamma(\psi)$.

3 Relevance Criterion in Belief Change

In literature, *Belief Change* refers to two different but related theories: Belief Revision and Belief Update [1,7]. Each of these activities is guided by a set of postulates that expresses some pre-requisites for belief change functions and describe how these functions should behave. In both theories, consistency maintenance and minimal change play a key role. However, Parikh observed that none of the theories follow the principle of minimal change. Ideally, if a statement φ in a belief base ψ does not share any propositional symbol with incoming information μ, then φ should belong to the resulting belief base after either the belief revision or belief update operation has been performed.

Formally, upon letting $\circ$ denote a belief revision operator, the following postulate captures the idea of relevant revision [13]:

(P) Let $\psi = \varphi \wedge \varphi'$ s.t. $\mathsf{Lang}(\varphi) \cap \mathsf{Lang}(\varphi') = \emptyset$. If $\mathsf{Lang}(\mu) \subseteq \mathsf{Lang}(\varphi)$, then $\psi \circ \mu \equiv (\varphi \circ' \mu) \wedge \varphi'$, where $\circ'$ is the revision operator restricted to language $\mathsf{Lang}(\varphi)$.

An open question, as stressed in [14], concerns the local revision operator mentioned in postulate **(P)**: this operator must be context-independent. Suppose there are two belief sets ψ and ψ' such that $\psi \equiv \varphi \wedge \varphi'$, $\psi' \equiv \varphi \wedge \varphi''$, $\mathsf{Lang}(\varphi) \cap \mathsf{Lang}(\varphi') = \emptyset$ and $\mathsf{Lang}(\varphi) \cap \mathsf{Lang}(\varphi'') = \emptyset$. Then only a single version of the local revision operator $\circ'$ should exist such that $\psi \circ \mu \equiv (\varphi \circ' \mu) \wedge \varphi'$ and $\psi' \circ \mu \equiv (\varphi \circ' \mu) \wedge \varphi''$ for any μ s.t. $\mathsf{Lang}(\mu) \subseteq \mathsf{Lang}(\varphi)$. Hereafter, we also commit to this *strong* version of **(P)**.

A relevant belief revision operator which minimizes the existing belief prime implicant change was proposed in [15]. That operator, denoted $\circ_{PI}$, satisfies Katsuno-Mendelzon's postulates for belief revision as well as Parikh's postulate for relevant revision. The first step in capturing the notion of relevance is to represent the belief base as its set of prime implicants. Prime implicants facilitate the splitting stage when performing the change by providing a canonical representation and the minimal language for representing belief base ψ.

Satisfaction of postulate (**P**) is assured then by the definition of faithful assignment, where preferences are defined within a subset of terms rather than on the whole set of possible models as required in [6,8]. The pre-order is only required to be set over the set of terms that can be built from $\Gamma(\psi)$. Let $\leqslant_\psi$ be a preference relation defined over the set of all possible terms in $\Gamma(\psi)$: $D \leqslant_\psi D'$ states that D is at least as close as D' w.r.t. ψ. The notion of faithful assignment is defined as follows.

Definition 1. *[15] A* faithful assignment *$\mathcal{F}_\psi$ is a function which maps every formula ψ to a pre-order over $\Gamma(\psi)$ s.t.:*[1]

(C1-T) *if $D, D' \in \mathcal{D}_\psi$, then $D \not<_\psi D'$.*
(C2-T) *if $D \in \mathcal{D}_\psi$ and $D' \notin \mathcal{D}_\psi$, then $D <_\psi D'$.*
(C3-T) *if $\psi \equiv \varphi$, then $\leqslant_\psi = \leqslant_\varphi$.*
(C4-T) *For all $D, D' \notin \mathcal{D}_\psi$, if $(D \subseteq D')$ then $D \sim_\psi D'$.*

[1] $D \sim_\psi D'$ stands for $D \leqslant_\psi D'$ and $D' \leqslant_\psi D$

Constraint **(C4-T)** states that preferences should not favor too specific terms. This is the first step towards the enforcing the notion of relevance.

Operator $\circ_{PI}$ commits to the *strong* version of postulate **(P)** by setting a constraint on faithful assignment. Suppose that $\psi \equiv \varphi \wedge \varphi'$ such that $\mathsf{Lang}(PI_\varphi) \cap \mathsf{Lang}(PI_{\varphi'}) = \emptyset$. Local revision operator $\circ'_{PI}$ used in (**P**) requires that there is only one pre-order $\leqslant_\varphi$ associated to φ. Suppose two terms D and $D' \in \Gamma(\varphi)$ such that $D \leqslant_\varphi D'$. Pre-order $\leqslant_\psi$ should also reflect these preferences; extending terms D and D' with any prime implicants belonging to $PI_{\varphi'}$ must not change preferences. The following constraint expresses the strong notion of relevance by considering multiple pre-orders.

(CS-T) Suppose φ and a faithful assignment $\mathcal{F}'_\psi$ s.t. $\mathcal{F}'_\psi(\varphi) = \leqslant_\varphi$. Faithful assignment $\mathcal{F}_\psi$ mapping each belief set ψ to a pre-order $\leqslant_\psi$ is said to be *relevant* iff for any φ, φ' s.t. $\psi \equiv \varphi \wedge \varphi'$ and $\mathsf{Lang}(PI_\varphi) \cap \mathsf{Lang}(PI_{\varphi'}) = \emptyset$; for any $D, D' \in \Gamma(\varphi)$: $D \leqslant_\varphi D'$ iff $D \cup D_{\varphi'} \leqslant_\psi D' \cup D'_{\varphi'}$ s.t. $D_{\varphi'}, D'_{\varphi'} \in PI_{\varphi'}$ and $D \cup D_{\varphi'}, D' \cup D'_{\varphi'} \in \Gamma(\psi)$.

Revising a belief set ψ by μ is then defined as selecting the preferred terms w.r.t. $\leqslant_\psi$. It has been shown that the resulting revision operator $\circ_{PI}$ defined by faithful assignment $\mathcal{F}_\psi$ satisfies postulate **(P)** if faithful assignment $\mathcal{F}_\psi$ satisfies constraint **(CS-T)**:

Theorem 1. *[15] Let $\mathcal{F}'_\psi$ be a faithful assignment that maps each belief set ψ' to a total pre-order $\leqslant'_\psi$. Let $\circ'_{PI}$ be the revision operator defined by $\mathcal{F}'_\psi$. Let $\mathcal{F}_\psi$ be a faithful assignment that maps each belief set ψ to a total pre-order $\leqslant_\psi$. Let $\circ_{PI}$ be the revision operator defined by $\mathcal{F}_\psi$.*

*If $\mathcal{F}_\psi$ satisfies constraint **(CS-T)** w.r.t. $\mathcal{F}'_\psi$ then $\circ_{PI}$ satisfies **(P)** w.r.t. revision operator $\circ'_{PI}$.*

The result of relevance is rooted in two key aspects: defining the revision operator $\circ_{PI}$ and committing to the strong version of the relevance postulate. Hence, the prime implicant based revision operator exactly characterizes the notion of relevant belief revision.

Dalal's operator and Relevant Revision

According to [11], $\circ_{PI}$ revision operator is equivalent to Dalal's revision operator [2]. Considering that $\circ_{PI}$ is relevant, we show below that Dalal's revision operator is also relevant. The notion of distance used by Dalal can be rephrased with respect to distance between terms belonging to $\Gamma(\psi)$. Every term that belongs to $\Gamma(\psi)$ can be rewritten as $D \cup (D_\psi - \overline{D})$ s.t. $D_\psi \in PI_\psi$ and $D \in \mathcal{D}$. Hence, the set $D \cap \overline{D_\mu}$, where D_μ are the terms of the new information μ, represents the contradicting literals between the belief base ψ and the new information μ. We introduce function κ that returns the set of propositional symbols associated with this set of contradicting literals and which allows us to rephrase Dalal's pre-order $\leqslant_\psi^{\mathsf{Da}}$.

Definition 2 **(κ).** *Let $D_1 \in \mathcal{D}$, $D_\psi \in PI_\psi$ and $D \in \Gamma(\psi)$ s.t. $D = D_1 \cup (D_\psi - \overline{D_1})$: $\kappa(D) = \{p \in P | p \in (D_\psi \cap \overline{D_1})$ or $\neg p \in (D_\psi \cap \overline{D_1})\}$*

Definition 3 **($\leqslant_\psi^{\mathsf{Da}}$).** *Let $D, D' \in \Gamma(\psi)$: $D \leqslant_\psi^{\mathsf{Da}} D' \iff |\kappa(D)| \leqslant_{\mathbb{N}} |\kappa(D')|$*

Let us state that Dalal's revision operator is relevant.

Proposition 1. *Let $\mathcal{F}_\psi^{\mathsf{Da}}$ be a function mapping a total pre-order $\leqslant_\psi^{\mathsf{Da}}$ to each belief set ψ. Function $\mathcal{F}_\psi^{\mathsf{Da}}$ is a faithful assignment which satisfies constraint (**CS-T**) w.r.t. faithful assignment $\mathcal{F}'_\psi = \mathcal{F}_\psi^{\mathsf{Da}}$.*

Proof. (sketch): It is straightforward to prove that (**C1-T**)–(**C3-T**) hold. Constraint (**C4-T**): suppose $D, D' \notin \mathcal{D}_\psi$ s.t. $D \subseteq D'$; suppose l s.t. $l \in \kappa(D')$ and $l \notin \kappa(D)$: either (i) $D \cup \{l\}$ is not consistent and thus $D \cup \{l\} \notin \Gamma(\psi)$ or (ii) l is consistent with D and thus $D \cup \{l\} \in \Gamma(\psi)$ then $\kappa(D) = \kappa(D \cup \{l\})$ and thus $\kappa(D) = \kappa(D')$. Hence (**C4-T**) holds. Constraint (**CS-T**): suppose it does not hold. Then it follows that $\exists \varphi, \varphi'$ s.t. $\psi \equiv \varphi \wedge \varphi'$, $\mathsf{Lang}(PI_\varphi) \cap \mathsf{Lang}(PI_{\varphi'}) = \emptyset$ and $\exists D, D' \in \Gamma(\varphi)$ s.t. $D \leqslant_\varphi D'$ and $D \cup D_{\varphi'} \nleqslant_\psi D' \cup D'_{\varphi'}$. Since $\mathsf{Lang}(PI_\varphi) \cap \mathsf{Lang}(PI_{\varphi'}) = \emptyset$ it follows that $D_{\varphi'}$ is consistent with D and D'; hence $\kappa(D) = \kappa(D \cup D_{\varphi'})$ and $\kappa(D') = \kappa(D' \cup D_{\varphi'})$ which contradicts $D \cup D_{\varphi'} \nleqslant_\psi D' \cup D'_{\varphi'}$. Since $D \cup D_{\varphi'}$, $D' \cup D'_{\varphi'} \in \Gamma(\psi)$, (**CS-T**) holds.

4 Relevant Belief Update

In this section we present the KM framework and we present how KM postulates are changed in order to consider sets of terms. We also show that Forbus' operator is relevant in the sense of Parikh, but it is not minimal. We present how a relevant and minimal operator can be obtained considering terms instead of models and we demonstrate how to achieve *Relevant Minimal Change.*

4.1 KM's Framework of Belief Update

Belief update concerns consistently inserting a new piece of information μ into a belief set ψ. The update operator is usually denoted by $\diamond$ and the resulting belief set is denoted $\psi \diamond \mu$. The KM postulates provide an axiomatic characterization of belief update operators in the context of finite propositional beliefs [7]:

(U1) $\psi \diamond \mu$ implies μ.
(U2) If ψ implies μ then $\psi \diamond \mu$ is equivalent to ψ.
(U3) If both ψ and μ are satisfiable then $\psi \diamond \mu$ is also satisfiable.
(U4) If $\psi_1 \equiv \psi_2$ and $\mu_1 \equiv \mu_2$ then $\psi_1 \diamond \mu_1 \equiv \psi_2 \diamond \mu_2$.
(U5) $(\psi \diamond \mu) \wedge \varphi$ implies $\psi \diamond (\mu \wedge \varphi)$.
(U6) If $\psi \diamond \mu_1$ implies μ_2 and $\psi \diamond \mu_2$ implies μ_1 then $\psi \diamond \mu_1 \equiv \psi \diamond \mu_2$.
(U7) If ψ is complete then $(\psi \diamond \mu_1) \wedge (\psi \diamond \mu_2)$ implies $\psi \diamond (\mu_1 \vee \mu_2)$.
(U8) $(\psi_1 \vee \psi_2) \diamond \mu \equiv (\psi_1 \diamond \mu) \vee (\psi_2 \diamond \mu)$.

Updating ψ by μ consists of choosing the closest models of μ with respect to each model of ψ [8,7]. Let $\preceq_w$ be a pre-order representing preferences defined over $\mathcal{W}$, where $\mathcal{W}$ is the set of all propositional interpretations defined over P. The closeness criterion: $w' \preceq_w w''$ states that w' is at least as close as w'' w.r.t. w. Faithful assignment represents preferences related to w, i.e, the most preferred model is w:[2]

Definition 4. *A faithful assignment $\mathcal{F}_w$ is a function that maps each interpretation w to a partial pre-order $\preceq_w$ s.t.:*

(C1) *for all $w' \in \mathcal{W}$ if $w \neq w'$ then $w \prec_w w'$*

Let $[\![\psi]\!]$ be the set of propositional interpretations that satisfy ψ, i.e., the models of ψ. Updating a belief set is then defined by selecting the preferred models of μ w.r.t. each $\preceq_w$.

Theorem 2. *[8] An update operator $\diamond$ satisfies* **(U1)**–**(U8)** *if and only if there exists a faithful assignment $\mathcal{F}_w$ that maps each interpretation w to a partial pre-order $\preceq_w$ s.t. $[\![\psi \diamond \mu]\!] = \bigcup_{w \in [\![\psi]\!]} \min([\![\mu]\!], \preceq_w)$.*

One of the simplest ways to set preferences is to consider the propositional symbols that may change. This has been proposed by Dalal in [2] and is applied to belief update in [18,5]. It consists of characterizing a belief change operator as a function which changes the minimal number of propositional symbol truth values in each ψ model so that incoming information can be added without entailing inconsistency.

4.2 Relevance Criterion on Belief Update

Since Dalal's operator is a relevant belief revision operator, the immediate question becomes: is it also the case for Dalal's belief update counter-part, the Forbus' operator [5]?

To get the answer, we first need to rephrase the Parikh's postulate for belief update. A naive translation is:

(P-U) Let $\psi = \varphi \wedge \varphi'$ s.t. $\mathsf{Lang}(\varphi) \cap \mathsf{Lang}(\varphi') = \emptyset$. If $\mathsf{Lang}(\mu) \subseteq \mathsf{Lang}(\varphi)$, then $\psi \diamond \mu \equiv (\varphi \diamond' \mu) \wedge \varphi'$, where $\diamond'$ is the update operator restricted to language $\mathsf{Lang}(\varphi)$.

Let us consider one example that illustrates the relevance issue with Forbus' belief update operator.

Example 2. Consider belief base $\psi = (p_2 \wedge p_3 \wedge p_5) \vee (p_4 \wedge p_5)$ and new piece of information $\mu = (p_1 \wedge p_2 \wedge \neg p_3) \vee (\neg p_1 \wedge \neg p_2 \wedge \neg p_3)$. Performing the update process

[2] $\prec_w$ is defined from $\preceq_w$ as usual, i.e., $w' \prec_w w''$ iff $w' \preceq_w w''$ but not $w' \preceq_w w''$.

using Forbus' operator means to calculating distances and preferences between models of ψ and μ: $w' \preceq_w w''$ iff $|d(w,w')| \leqslant_{\mathbb{N}} |d(w,w'')|$, where function d gives the set of propositional symbols that differ between w and w'. The resulting belief base is given by the models of μ that are the closest to each model of ψ:

$$\begin{aligned}[\![\psi \diamond \mu]\!] = \{&\{p_1, p_2, \neg p_3, p_4, p_5\}, \{\neg p_1, \neg p_2, \neg p_3, p_4, p_5\}\\ &\{p_1, p_2, \neg p_3, \neg p_4, p_5\}, \{\neg p_1, \neg p_2, \neg p_3, \neg p_4, p_5\}\}\end{aligned}$$

that corresponds to the following implicants:

$$\psi \diamond \mu = (p_1 \wedge p_2 \wedge \neg p_3 \wedge p_5) \vee (\neg p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge p_5)$$

The belief base ψ can be split into φ and φ' such that $\mathsf{Lang}(\varphi) = \{p_1, p_2, p_3, p_4\}$ and $\mathsf{Lang}(\varphi') = \{p_5\}$, such that $\mathsf{Lang}(\varphi) \cap \mathsf{Lang}(\varphi') = \emptyset$. Literal p_5 is preserved in the resulting belief base, and thus the update process performed using Forbus' operator seems relevant in the sense of Parikh.

However, we face two caveats. First, it is not minimal: literal p_4 appears in one implicant of ψ but not in the representation of μ and p_4 is also concerned with the update operation (p_4 no longer explicitly appears in the resulting belief base); second, the constraint relating languages is too strong if we want to perform update as suggested by the example.

Since each prime implicant of ψ stresses up the relevant literals for representing ψ, update should also focus on these relevant literals. It means that update should be performed by considering each prime implicant of ψ rather than considering each model of ψ.

To enforce this new way to update a belief set, we extend the definition of $\Gamma(\psi)$ so that we consider one term D and a formula μ such that every prime implicant of μ is extended with the maximal consistent part of term D:

$$\Gamma(D, \mu) = \{D_\mu \cup (D - \overline{D_\mu}) | D_\mu \in PI_\mu\}$$

Hence, the "relevant minimal change" operator should pick up terms $D_{(\psi,\mu)}$ of set $\bigcup_{D_\psi \in PI_\psi} \Gamma(D_\psi, \mu)$ that are the closest to each prime implicant D_ψ in PI_ψ as illustrated in Figure 2.

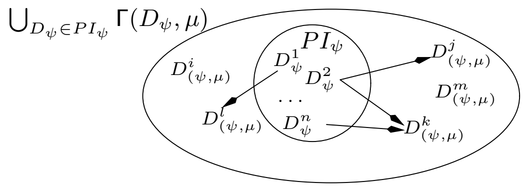


Fig. 2. Belief update performed over terms of PI_ψ

4.3 Belief Update in Prime Implicants

Our aim is to develop a theorem similar to theorems 1 and 2 that describes belief update operation in terms of preferences over terms. As explained, preferences are now indexed by the prime implicants D_ψ of ψ rather than by the models of ψ. First we rewrite constraint (**C1**) that characterizes preferences overs models: all terms which are entailed by an implicant D are strictly preferred to the terms that are not entailed by D. Secondly, we rewrite constraint (**C4-T**), which avoids setting preferences that favor too specific terms (cf. section 3). These two constraints characterize the notion of faithful assignment defined over terms.

Definition 5. *A* faithful assignment $\mathcal{F}_D$ *is a function which maps every* $D \in \mathcal{D}$ *to a partial pre-order* $\leqslant_D$ *defined over* $\Gamma(D)$ *s.t.:*

(CU1-T) *For all* $D', D'' \in \mathcal{D}$*, if* $D' \in \Gamma(D)$*,* $D'' \notin \Gamma(D)$ *then* $D' <_D D''$
(C4U-T) *For all* $D', D'' \in \Gamma(D)$*, if* $D' \subseteq D''$ *then* $D' \sim_D D''$*.*

Since preferences are indexed by terms instead of models, postulates **(U1)**–**(U8)** that characterize the notion of update by applying change to each model of the initial belief set have to be reformulated in order to accommodate the notion of change that leads to relevance: changes should be applied to each *prime implicant* of the initial belief set. In fact, all postulates are identical to postulates **(U1)**–**(U8)** except postulates **(U7-T)** and **(U9-T)**. Let $\psi \diamond_{PI} \mu$ denote the updated belief base. The following postulates characterize $\diamond_{PI}$:

(U1-T) $\psi \diamond_{PI} \mu$ implies μ.
(U2-T) If ψ implies μ then $\psi \diamond_{PI} \mu$ is equivalent to ψ.
(U3-T) If both ψ and μ are satisfiable then $\psi \diamond_{PI} \mu$ is also satisfiable.
(U4-T) If $\psi_1 \equiv \psi_2$ and $\mu_1 \equiv \mu_2$ then $\psi_1 \diamond_{PI} \mu_1 \equiv \psi_2 \diamond_{PI} \mu_2$.
(U5-T) $(\psi \diamond_{PI} \mu) \wedge \varphi$ implies $\psi \diamond_{PI} (\mu \wedge \varphi)$.
(U6-T) If $\psi \diamond_{PI} \mu_1$ implies μ_2 and $\psi \diamond_{PI} \mu_2$ implies μ_1 then $\psi \diamond_{PI} \mu_1 \equiv \psi \diamond_{PI} \mu_2$.
(U7-T) If $PI_\psi = \{D_\psi\}$ then $(\psi \diamond_{PI} \mu_1) \wedge (\psi \diamond_{PI} \mu_2)$ implies $\psi \diamond_{PI} (\mu_1 \vee \mu_2)$.
(U8-T) $(\psi_1 \vee \psi_2) \diamond_{PI} \mu \equiv (\psi_1 \diamond_{PI} \mu) \vee (\psi_2 \diamond_{PI} \mu)$.
(U9-T) If $PI_\psi = \{D_\psi\}$ and $PI_\mu = \{D_\mu\}$ then $\psi \diamond_{PI} \mu = D_\mu \cup (D_\psi - \overline{D_\mu})$.

Postulate **(U7-T)** rephrases the condition "ψ is complete" as "ψ is represented by only one prime implicant". Combining **(U8-T)** and **(U7-T)** leads to update ψ by considering each prime implicant alternately. Postulate **(U9-T)** stresses up the second key difference between $\diamond$ and $\diamond_{PI}$: the result given by $\diamond_{PI}$ is a subset of the set $\bigcup_{D_\psi \in PI_\psi} \Gamma(D_\psi, \mu)$ while the result given by $\diamond$ is a subset of $\mathcal{W}$. Following [7], we now show that whenever constraints **(CU1-T)** and (**C4U-T**) hold, the nine update postulates are satisfied:

Theorem 3 (Update operator). *Let* $\mathcal{F}_D$ *be a faithful assignment that maps each term* $D \in \mathcal{D}$ *to a partial pre-order* $\leqslant_D$*. PI update operator* $\diamond_{PI}$ *defined by* $\mathcal{F}_D$ *satisfies* **(U1-T)**–**(U9-T)** *if*

$$\psi \diamond_{PI} \mu =_{def} \bigcup_{D_\psi \in PI_\psi} \min(\Gamma(D_\psi, \mu), \leqslant_{D_\psi})$$

Proof. (Sketch) The proof is almost a direct translation of the proof of theorem 2 given in [7] and theorem 5 in [11]. (**U1-T**)–(**U4-T**) and (**U8-T**) are consequences of the definitions of Γ and $\diamond_{PI}$. Constraint (**C4U-T**) enforces postulates (**U5-T**)–(**U7-T**). Let us focus on (**U5-T**): suppose the case where ψ and $(\psi \diamond_{PI} \mu) \wedge \varphi$ are consistent, then there exists $D \in (\psi \diamond_{PI} \mu)$ and D_φ s.t. $D \wedge D_\varphi$ is consistent. It follows that there exists D_ψ s.t. D is minimal w.r.t. $\leqslant_{D_\psi}$. Constraint (**C4U-T**) entails that $D \cup D_\varphi$ is also minimal. Hence (**U5-T**) holds. Postulate (**U9-T**) holds since the results given by $\diamond_{PI}$ is a subset of $\bigcup_{D_\psi \in PI_\psi} \Gamma(D_\psi, \mu)$.

4.4 Relevant Belief Update

Postulates (**U7-T**) and (**U9-T**) represent the first key step to handling *Relevant Minimal Change*. The second step is to rewrite Parikh's postulate in the context of belief update. Relevance has to be set by constraining faithful assignments. Consider a term D which can be split in a conjunction of two terms which do not share any symbols: $D \equiv D_1 \wedge D_2$. Suppose one pre-order $\leqslant_{D_1}$ defined by faithful assignment $\mathcal{F}'_D$. Now, suppose two terms $D, D' \in \Gamma(D_1, \mu)$ such that $D \leqslant_{D_1} D'$. Relevance states that adding D_2 to D and D' should not switch the preferences about D and D' since D_2 is expressed with symbols that differ from the symbols of D_1; that is $D \cup D_2 \leqslant_D D' \cup D_2$ (provided that $D \cup D_2$ and $D' \cup D_2$ are consistent, i.e. they belong to $\Gamma(D)$).

(CUS-T) Suppose D_1 and a faithful assignment $\mathcal{F}'_D$ s.t. $\mathcal{F}'_D(D_1) = \leqslant_{D_1}$. Faithful assignment $\mathcal{F}_D$ mapping each $D \in \mathcal{D}$ to a pre-order $\leqslant_D$ is *relevant* iff for any D, D_2 s.t. $D \equiv D_1 \wedge D_2$ and $\mathsf{Lang}(D_1) \cap \mathsf{Lang}(D_2) = \emptyset$; for any $D', D'' \in \Gamma(D_1)$: $D' \leqslant_{D_1} D''$ iff $D' \cup D_2 \leqslant_D D'' \cup D_2$ s.t. $D' \cup D_2, D'' \cup D_2 \in \Gamma(D)$.

Now, we show that operator $\diamond_{PI}$ characterizes relevant belief update by satisfying postulate based on **(P)**. The constraint (**CUS-T**) stating relevance by considering multiple assignments stresses that changes should be performed by handling implicants. Hence, the postulate for relevance should explicitly mention operator $\diamond_{PI}$ in its definition. We rephrase Parikh's postulate in terms of the prime implicant representation of belief since it enables the clear separation of relevant and non-relevant literals used to represent ψ:[3]

(PU-T) Let $PI_\psi = PI_\varphi \times PI_{\varphi'}$. If (i) $\mathsf{Lang}(PI_\mu) \cap \mathsf{Lang}(PI_{\varphi'}) = \emptyset$ and (ii) $\forall \varphi'', \varphi'''$ s.t. $PI_\psi = PI_{\varphi''} \times PI_{\varphi'''}$ and $\mathsf{Lang}(PI_\mu) \cap \mathsf{Lang}(PI_{\varphi'''}) = \emptyset$, $\mathsf{Lang}(PI_{\varphi'''}) \subseteq \mathsf{Lang}(PI_{\varphi'})$; then $\psi \diamond_{PI} \mu \equiv (\varphi \diamond'_{PI} \mu) \wedge \varphi'$, where $\diamond'_{PI}$ is the PI update operator restricted to the language $\mathsf{Lang}(PI_\varphi)$.

The definition of the constraint states that if there exist φ and φ' s.t. $\psi = \varphi \wedge \varphi'$ and φ' is the formula that has the largest set of symbols (condition (ii)) which are not shared with those of μ (condition (i)), then $\diamond_{PI}$ should not change φ'.

If a faithful assignment satisfies constraint **(CUS-T)**, then operator $\diamond_{PI}$ satisfies the relevance postulate for update.

[3] $PI_\varphi \times PI_{\varphi'}$ is the Cartesian product of sets PI_φ and $PI_{\varphi'}$.

Theorem 4. *Suppose PI update operator $\diamond'_{PI}$ defined by the faithful assignment $\mathcal{F}'_D$. Let $\mathcal{F}_D$ be a faithful assignment that maps each $D \in \mathcal{D}$ to a partial pre-order $\leqslant_D$. PI update operator $\diamond_{PI}$ defined by $\mathcal{F}_D$ satisfies **(PU-T)**, w.r.t. operator $\diamond'_{PI}$, if $\mathcal{F}_D$ satisfies **(CUS-T)** w.r.t. faithful assignment $\mathcal{F}'_D$.*

Proof. (sketch) If it is not the case, there exist φ and φ' s.t. $PI_\psi = PI_{\varphi \wedge \varphi'}$, $\mathsf{Lang}(PI_\mu) \cap \mathsf{Lang}(PI_{\varphi'}) = \emptyset$ and $\psi \diamond_{PI} \mu \not\equiv (\varphi \diamond'_{PI} \mu) \wedge \varphi'$. Suppose that $\psi \diamond_{PI} \mu \not\Rightarrow (\varphi \diamond'_{PI} \mu) \wedge \varphi'$. It entails, because of the definition of $\diamond_{PI}$, that there exist D_ψ and $D \in \min(\Gamma(D_\psi, \mu), \leqslant_\psi)$ s.t. $D \not\models (\varphi \diamond'_{PI} \mu) \wedge \varphi'$. There also exist $D' \in \Gamma(\varphi)$ and $D_{\varphi'} \in PI_{\varphi'}$ s.t. $D = D' \cup D_{\varphi'}$ because of the definition of Γ and $\mathsf{Lang}(PI_\varphi) \cap \mathsf{Lang}(PI_{\varphi'}) = \emptyset$. Condition **(CUS-T)** entails that D' belongs to $\min(\Gamma(D_\varphi, \mu), \leqslant_\varphi)$ and thus $D \models (\varphi \diamond'_{PI} \mu) \wedge \varphi'$. Contradiction. Proof for the case $(\varphi \diamond'_{PI} \mu) \wedge \varphi' \Rightarrow \psi \diamond_{PI} \mu$ is similar.

Let us look at the opposite way: suppose an update operator $\diamond_{PI}$ which satisfies postulates **(U1-T)**–**(U9-T)** and **(PU-T)**; the question becomes "is there a relevant faithful assignment that can produce the same result?" If the answer is positive then it means that in fact operator $\diamond_{PI}$ characterizes the family of belief update operators that produces minimal relevant change. The following theorem shows that it is in fact the case if we focus on the strong meaning of relevance:

Theorem 5. *Suppose PI update operator $\diamond'_{PI}$ s.t. **(U1-T)**–**(U9-T)**; Suppose PI update operator $\diamond_{PI}$ s.t. **(U1-T)**–**(U9-T)** and **(PU-T)** hold w.r.t. PI update operator $\diamond'_{PI}$. Then (i) there exists a faithful assignment $\mathcal{F}'_D$ that maps every $D \in \mathcal{D}$ to a pre-order $\leqslant'_D$ s.t.*

$$\psi \diamond'_{PI} \mu =_{def} \bigcup_{D_\psi \in PI_\psi} \min(\Gamma(D_\psi, \mu), \leqslant'_{D_\psi})$$

*and (ii) there exists a relevant faithful assignment $\mathcal{F}_\mathcal{D}$ satisfying constraint **(CUS-T)** w.r.t. to faithful assignment $\mathcal{F}'_D$ s.t.*

$$\psi \diamond_{PI} \mu =_{def} \bigcup_{D_\psi \in PI_\psi} \min(\Gamma(D_\psi, \mu), \leqslant_{D_\psi})$$

Proof. (Sketch) Suppose $\psi \diamond_{PI} \mu$ s.t. postulates (**U1-T**)–(**U9-T**) and (**PU-T**) hold; let us define preferences of faithful assignment $\mathcal{F}_D$ as follows: for any terms D, D' and $D'' \in \mathcal{D}$, there exist D_1 and $D_2 \in \mathcal{D}$ s.t. $D' = D_1 \cup (D - \overline{D_1})$ and $D'' = D_2 \cup (D - \overline{D_2})$. We set $D' \leqslant_D D''$ iff $D \subseteq D'$ or $D \diamond_{PI} (D_1 \vee D_2) = \{D'\}$. Reflexivity and transitivity are proven as in [11]. (**CU1-T**) holds because : (i) for all terms D' subsumed by D, it holds that $D' \leqslant_D D''$ and (ii) (**U2-T**) entails that $D'' \not\leqslant_D D'$ for all $D'' \not\subseteq D$. Constraint (**C4-T**) holds because of postulate (**U5-T**) and also (**U7-T**). Finally, (**U5-T**), (**U7-T**)–(**U9-T**) entails that $D_\psi \diamond_{PI} \mu = \min(\Gamma(D_\psi, \mu), \leqslant_{D_\psi})$ which then entails that $\psi \diamond_{PI} \mu = \cup_{D_\psi \in PI_\psi} \min(\Gamma(\psi, \mu), \leqslant_{D_\psi})$. Finally, we prove that constraint (**CUS-T**) holds: suppose $\mathcal{F}'_D$ is defined in a similar way to $\mathcal{F}_D$ and based on $\diamond'_{PI}$. For all $D \in \mathcal{D}$, assume $D \equiv D_3 \wedge D_4$ and let us go back to the way we set preferences: $D \diamond_{PI} (D_1 \vee D_2) = \{D'\}$ and by (**PU-T**), it holds that $D' \equiv D_3 \diamond'_{PI} (D_1 \vee D_2) \wedge D_4$. Consequently $D_3 \diamond'_{PI} (D_1 \vee D_2) \equiv D' - D_4$. Moreover D' is minimal and also $D' - D_4$ (see above). Hence (**CUS-T**) holds.

We conclude the characterization of $\diamond_{PI}$ by showing that Forbus-based PI update is minimal and relevant:

Proposition 2. *Let $\mathcal{F}^{\mathsf{Fo}}_D$ be a function mapping a pre-order $\leqslant^{\mathsf{Da}}_D$ to each $D \in \mathcal{D}$ (cf. Def. 2 and 3). Function $\mathcal{F}^{\mathsf{Fo}}_D$ is a faithful assignment which satisfies (**CUS-T**) w.r.t. faithful assignment $\mathcal{F}'_D = \mathcal{F}^{\mathsf{Fo}}_D$.*

The proof is similar to the proof of Proposition 1. Let us illustrate the proposition by reconsidering Example 2:

Example 3. Consider belief base ψ and new piece of information μ as presented in Example 2 and represented as $PI_\psi = (p_2 \wedge p_3 \wedge p_5) \vee (p_4 \wedge p_5)$ and $PI_\mu = (p_1 \wedge p_2 \wedge \neg p_3) \vee (\neg p_1 \wedge \neg p_2 \wedge \neg p_3)$. Definitions 2 and 3 give the following faithful assignment $\mathcal{F}^{\mathsf{Fo}}_D$ with pre-orders $\leqslant^{\mathsf{Da}}_{D_\psi}$:

$$\begin{array}{lll} \{p_1, p_2, \neg p_3, p_5\} & <^{\mathsf{Da}}_{\{p_2,p_3,p_5\}} & \{\neg p_1, \neg p_2, \neg p_3, p_5\} \\ \{p_1, p_2, \neg p_3, p_4, p_5\} & \leqslant^{\mathsf{Da}}_{\{p_4,p_5\}} & \{\neg p_1, \neg p_2, \neg p_3, p_4, p_5\} \end{array}$$

Let $\diamond^{\mathsf{Fo}}_{PI}$ be the PI update operator defined by $\mathcal{F}^{\mathsf{Fo}}_D$. We get:

$$\begin{aligned} \psi \diamond^{\mathsf{Fo}}_{PI} \mu = {} & (p_1 \wedge p_2 \wedge \neg p_3 \wedge p_5) \vee \\ & (p_1 \wedge p_2 \wedge \neg p_3 \wedge p_4 \wedge p_5) \vee \\ & (\neg p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge p_4 \wedge p_5) \end{aligned}$$

As expected, operator $\diamond^{\mathsf{Fo}}_{PI}$ preserves literals of prime implicant $(p_4 \wedge p_5)$.

5 Conclusion

This paper proposed a framework for handling relevant minimal update. We go beyond Parikh to ensure that literals without relation with new information are preserved. Operator $\diamond_{PI}$ is characterized both in terms of postulates and faithful assignment over terms. Performing belief update over terms, i.e, set of models ensures the syntax independence principle. Besides that, since beliefs are represented as sets of prime implicants, the belief update operator $\diamond_{PI}$ is not computationally more complex when applied in a relevant belief update process. In fact, Theorems 3–5 stress that $\diamond_{PI}$ exactly characterizes update operators that produce *relevant minimal change.*

There is a subtle link between relevance belief update and the frame problem [12]. On the one hand, these two problems are closely related. A solution to the frame problem requires separating irrelevant fluents from relevant fluents. If we know which fluents we should update after performing an action, these fluents are relevant and the rest are irrelevant. This means that a solution to relevance updating is a solution to the frame problem. On the other hand, a solution to the frame problem needs to be attached to an action logic, which is normally a high-order logic, either dynamic logic or situation calculus. Prime implicants are not expressive enough to represent actions and their effects. How to apply the techniques we introduced in this paper to an action logic will be a promising research topic for the future.

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: partial meet contraction and revision functions. J. of Symbolic Logic 50(2), 510–530 (1985)
2. Dalal, M.: Investigations into a theory of knowledge base revision: Preliminary report. In: Proc. of AAAI 1988, pp. 475–479 (1988)
3. Darwiche, A., Marquis, P.: A Knowledge Compilation Map. Journal of Artificial Intelligence Research (17), 229–264 (2002)
4. Van de Putte, F.: Prime implicates and relevant belief revision. Journal of Logic and Computation 7, 1–11 (2011), DOI: 10.1093/logcom/exr040
5. Forbus, K.: Introducing actions into qualitative simulation. In: Proc. of IJCAI 1989, pp. 1273–1278 (1989)
6. Grove, A.: Two Modellings for Theory Change. J. of Philosophical Logic 17, 157–170 (1988)
7. Katsuno, H., Mendelzon, A.: On the difference between updating a knowledge base and revising it. In: Proc. of KR 1991, pp. 387–394 (1991)
8. Katsuno, H., Mendelzon, A.: Propositional knowledge base revision and minimal change. Artificial Intelligence 52(3), 263–294 (1991)
9. Kourousias, G., Makinson, D.: Parallel interpolation, splitting, and relevance in belief change. J. Symb. Log. 72(3), 994–1002 (2007)
10. Makinson, D., Kourousias, G.: Respecting relevance in belief change. Análisis Filosófico 26(1), 53–61 (2006)
11. Marchi, J., Bittencourt, G., Perrussel, L.: Prime forms and minimal change in propositional belief bases. Annals of Math. and AI (2010)
12. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence 4, 463–502 (1969)
13. Parikh, R.: Beliefs, belief revision, and splitting languages, vol. 2, pp. 266–278. Center for the Study of Language and Information, Stanford (1999)
14. Peppas, P., Chopra, S., Foo, N.: Distance semantics for relevance-sensitive belief revision. In: Proc. of KR 2004, pp. 319–328 (2004)
15. Perrussel, L., Marchi, J., Zhang, D.: Characterizing Relevant Belief Revision Operators. In: Li, J. (ed.) AI 2010. LNCS, vol. 6464, pp. 42–51. Springer, Heidelberg (2010)
16. Perrussel, L., Marchi, J., Bittencourt, G.: Prime implicants and belief update. In: Proceedings of the Twenty-Second International FLAIRS Conference 2009, pp. 577–598 (2009)
17. Quine, W.V.O.: On cores and prime implicants of truth functions. American Mathematics Monthly 66, 755–760 (1959)
18. Winslett, M.: Reasoning about action using a possible models approach. In: Proc. of AAAI 1988, pp. 89–93 (1988)
19. Wu, M., Zhang, D., Zhang, M.: Language splitting and relevance-based belief change in horn logic. In: AAAI (2011)

Minimal Proof Search for Modal Logic K Model Checking*

Abdallah Saffidine

LAMSADE, Université Paris-Dauphine
abdallah.saffidine@dauphine.fr

Abstract. Most modal logics such as S5, LTL, or ATL are extensions of Modal Logic K. While the model checking problems for LTL and to a lesser extent ATL have been very active research areas for the past decades, the model checking problem for the more basic Multi-agent Modal Logic K (MMLK) has important applications as a formal framework for perfect information multi-player games on its own.

We present Minimal Proof Search (MPS), an effort number based algorithm solving the model checking problem for MMLK. We prove two important properties for MPS beyond its correctness. The (dis)proof exhibited by MPS is of minimal cost for a general definition of cost, and MPS is an optimal algorithm for finding (dis)proofs of minimal cost. Optimality means that any comparable algorithm either needs to explore a bigger or equal state space than MPS, or is not guaranteed to find a (dis)proof of minimal cost on every input.

As such, our work relates to A* and AO* in heuristic search, to Proof Number Search and DFPN+ in two-player games, and to counterexample minimization in software model checking.

1 Introduction

Model checking for temporal logics such as LTL or CTL is a major research area with important applications in software and hardware verification [4]. Model checking for agent logics such as ATL or S5 is now also regarded as an important topic with a variety of applications [17,18,11]. On the other hand, Modal Logic K is usually considered the basis upon which more elaborate modal logics are built, such as S5, PDL, LTL, CTL, or ATL [2,16]. Multi-agent Modal Logic K (MMLK) can also be used directly to model (sequential) perfect information games.

In this article, we put forward a model checking algorithm for MMLK that we call Minimal Proof Search (MPS). As the name indicates, given a model checking problem $q \models \phi$, the MPS algorithm outputs a proof that q satisfies ϕ or a counterexample, this proof/counterexample being minimal for some definition of size. Perfect information games provide at least two motivations for small proofs. In game playing, people are usually interested in "short" proofs, for instance a CHESS player would rather deliver checkmate in three moves than in nine moves

* A longer version of this article is available at http://arxiv.org/abs/1207.1832v1.

even if both options grant them the victory. In game solving, "compact" proofs can be stored and independently checked efficiently.

In CTL model checking, finding a minimal witness/counterexample is NP-complete [3]. MMLK model checking, on the contrary, though PTIME-complete [10], allows finding minimal witnesses/counterexamples relatively efficiently as we shall see in this article.

Our goal is related both to heuristic search and software model checking. On one hand, the celebrated A* algorithm outputs a path of minimal cost from a starting state to a goal state. This path can be seen as the proof that the goal state is reachable, and the cost of the path is the size of the proof. On the other hand, finding small counterexamples is an important subject in software model checking. For a failure to meet a specification often indicates a bug in the program, and a small counterexample makes finding and correcting the bug easier [7].

Like A*, MPS is optimal, in the sense that any algorithm provided with the same information and guaranteed to find a proof of minimal size needs to do as many node expansions as MPS.

The tableau-based model checking approach by Cleaveland for the μ-calculus seems to be similar to ours [6], however it would need to be adapted to handle (dis)proof cost. Also, in our understanding, the proof procedure *check1* presented by Cleaveland can be seen as an unguided depth first search while our approach is guided towards regions of minimal cost.

The two algorithms most closely related to MPS are AO*, a generalization of A* to And/Or trees, and DFPN+ [13], a variant of DFPN, itself a depth-first variant of Proof Number Search (PNS) [1].

While And/Or trees are as expressive as the combination of MMLK and Game Automata (GAs), we believe that the separation of concerns between the logic and the Game Automaton is beneficial in practice. For instance, if the properties to be checked are encoded in the logic rather than in the graph, there is no need to rewrite the rules of CHESS if one is interested in finding helpmates instead of checkmates, or if one just wants to know if any piece can be captured in two moves from a given position. The encoding through an And/Or graph would be different in every such situation while in our approach, only the modal logic formula needs to be adapted. Another advantage of MPS over AO* is that if the problem is not solvable, then MPS finds a minimal disproof while AO* does not provide such a guarantee.[1]

DFPN+ is typically only used to find a winning strategy for either player in two-player games. MPS, on the contrary, can be applied to solve other interesting problems without a cumbersome And/Or graph prior conversion. Example of such problems range from finding ladders in two-player games to finiding paranoid wins in multi-player games. Another improvement over DFPN+ is that we allow for a variety of (dis)proof size definitions. While DFPN+ is set to minimize

[1] Following the convention in Proof Number Search, we use the term proof and disproof instead of witness and counterexample which are more common in the model checking literature.

the total edge cost in the proof, we can imagine minimizing, say, the number of leaves or the depth of the (dis)proof.

In his thesis, Nagai derived the DFPN algorithm from the equivalent best-first algorithm PNS [13]. Similarly, we can obtain a depth-first version of MPS from the best first search version presented here by adapting Nagai's transformation. Such a depth-first version should probably be favoured in practice, however we decided to present the best first version in this article for two main reasons. We believe the best-first search presentation is more accessible to the non-specialists. The proofs seemed to be easier to work through in the chosen setting, and they can later be extended to the depth-first setting.

The remainder of this paper is structured as follows. In Sect. 2 we recall the definitions of Game Automaton (GA) and MMLK and formally define (dis)proofs for the corresponding model checking problem. Section 3 elaborates on the notion of (dis)proof cost and the associated basic admissible heuristic functions, it then proceeds with the presentation of the MPS algorithm. Finally, we prove the correctness of MPS, the minimality of the output (dis)proofs and the optimality of the algorithm in Sect. 4. A short discussion concludes the article.

2 Definitions

We define in this section various formal objects that will be used throughout the paper. The GA is the underlying system which is to be formally verified. The MMLK is the language to express the various properties we want to model check GAs against. Finally, a (dis)proof is a tree structure that shows whether a property is true on a state in a GA.

2.1 Game Automata

A GA is a kind of labelled transition system where both the states and the transitions are labelled. If a GA is interpreted as a perfect information game, then a transition corresponds to a move from one state to the next and its label is the player making that move. The state labels are domain specific information about states, for instance we could have a label for each triple (piece, owner, position) in CHESS-like games. Naturally, it is also possible to give a formal definition of GAs.

Definition 1. *A* Game Automaton *is a 5-tuple $G = (P, \Sigma, Q, \pi, \delta)$ with the following components:*

- P *is a non-empty set of* atoms *(or state labels)*
- Σ *is a non-empty finite set of* agents *(or transition labels)*
- Q *is a set of* game states
- $\pi : Q \to 2^P$ *maps each state q to its labels*
- $\delta : Q \times \Sigma \to 2^Q$ *is a transition function that maps a state and an agent to a set of next states.*

In the following, we will use p, p', p_1, ...for atoms, a for an agent, and q, q', q_1, ...for game states. We write $q \xrightarrow{a} q'$ when $q' \in \delta(q, a)$ and we read *agent a can move from q to q'*. Note that δ returns the set of successors, so it need not be a partial function to allow for states without successors. If an agent a has no moves in a state q, we have $\delta(q, a) = \emptyset$.

2.2 Multi-agent Modal Logic K

Following loosely [2], we define the Multi-agent Modal Logic K over a set of atoms P as the formulas we obtain by combining the negation and conjunction operators with a set of *box* operators, one per agent.

Definition 2. *The set T of well-formed* Multi-agent Modal Logic K (MMLK) *formulas is defined inductively as $\phi := p \mid \neg\phi' \mid \Box_a \phi' \mid \phi_1 \wedge \phi_2$ where ϕ, ϕ', ϕ_1,... stand for arbitrary MMLK formulas.*

We can define the usual syntactic shortcuts for the disjunction and the *diamond* operators $\phi_1 \vee \phi_2 \overset{\text{def}}{=} \neg(\neg\phi_1 \wedge \neg\phi_2)$ and $\Diamond_a \phi \overset{\text{def}}{=} \neg \Box_a \neg\phi$. The box operators convey necessity and the diamond operators convey possibility: $\Box_a \phi$ can be read as *it is necessary for agent a that ϕ*, while $\Diamond_a \phi$ is *it is possible for a that ϕ*.

2.3 The Model Checking Problem

We can now interpret MMLK formulas over GAs via the satisfaction relation $\models$. Intuitively, a state in a GA constitutes the context of a formula, while a formula constitutes a property of a state. A formula might be satisfied in some contexts and not satisfied in other contexts, and some properties hold in a state while others do not. Determining whether a given formula ϕ holds in a given state q (in a given implicit GA) is what is commonly referred to as *the model checking problem.* If it is the case, we write $q \models \phi$, otherwise we write $q \not\models \phi$.

It is possible to decide whether $q \models \phi$ by examining the structure of ϕ, the labels of q, as well as the accessible states.

Definition 3. *The formulas* satisfied *by a state q can be constructed by induction as follows.*

- *If p is a label of q, that is if $p \in \pi(q)$, then $q \models p$;*
- *if $q \not\models \phi$ then $q \models \neg\phi$;*
- *if $q \models \phi_1$ and $q \models \phi_2$ then $q \models \phi_1 \wedge \phi_2$;*
- *if for all q' such that $q \xrightarrow{a} q'$, we have $q' \models \phi$, then $q \models \Box_a \phi$.*

2.4 Proofs and Counterexamples

In practice, we never explicitly construct the complete set of formulas satisfied by a state. So when some computation tells us that a formula ϕ is indeed (not) satisfied by a state q, some sort of evidence might be desirable. In software model checking, a model of the program replaces the GA, and a formula in a temporal logic acts as a specification of the program. If a correct model checker asserts that the program does not satisfy the specification, it means that the program or the specification contained a bug. In those cases, it can be very useful for the programmers to have access to an evidence by the model checker of the mismatch between the formula and the system as it is likely to lead them to the bug.

In this section we give a formal definition of what constitutes a *proof* or a *disproof* for the class of model checking problems we are interested in. It is possible to relate the following definitions to the more general concept of *tree-like* counterexamples used in model checking ACTL [5].

Definition 4. *An* exploration tree *for a formula ϕ in a state q is a tree with root n associated with a pair (q, ϕ) with q a state and ϕ a formula, such that n satisfies the following properties.*

- *If n is associated with (q, p) with $p \in P$, then it has no children;*
- *if n is associated with $(q, \neg\phi)$ then n has at most one child and it is an exploration tree associated with (q, ϕ);*
- *if a node n is associated with $(q, \phi_1 \wedge \phi_2)$ then any child of n (if any) is an exploration tree associated with (q, ϕ_1) or with (q, ϕ_2);*
- *if a node n is associated with $(q, \Box_a \phi)$ then any child of n (if any) is an exploration tree associated with (q', ϕ) for some q' such that $q \xrightarrow{a} q'$.*
- *In any case, no two children of n are associated with the same pair.*

Unless stated otherwise, we will not distinguish between a tree and its root node. In the rest of the paper, n, n', n_1, ... will be used to denote nodes in exploration trees.

Definition 5. *A* proof *(resp. a* disproof*) that $q \models \phi$ is an exploration tree with a root n associated with (q, ϕ) satisfying the following hypotheses.*

- *If $\phi = p$ with $p \in P$, then $p \in \pi(q)$ (resp. $p \notin \pi(q)$);*
- *if $\phi = \neg\phi'$, then n has exactly one child n' and this child is a disproof (resp. proof);*
- *if $\phi = \phi_1 \wedge \phi_2$, then n has exactly two children n_1 and n_2 such that n_1 is a proof that $q \models \phi_1$ and n_2 is a proof that $q \models \phi_2$ (resp. n has exactly one child n' and n' is a disproof that $q \models \phi_1$ or n' is a disproof that $q \models \phi_2$);*
- *if $\phi = \Box_a \phi'$, then n has exactly one child n' for each $q \xrightarrow{a} q'$, and n' is a proof for $q' \models \phi'$ (resp. n has exactly one child n' and n' is a disproof for $q' \models \phi'$ for some $q \xrightarrow{a} q'$).*

3 Minimal Proof Search

Let $q \models \phi$ be a model checking problem and n_1 and n_2 two proofs as defined in Sect. 2.4. Even if n_1 is not a subtree of n_2, there might be reasons to prefer, n_1 over n_2. For instance, we can imagine that n_1 contains fewer nodes than n_2, or that the depth of n_1 is smaller than that of n_2.

3.1 Cost Functions

To remain as general as possible with respect to the definitions of a *small* (dis)proof in the introduction, we introduce a cost function k as well as cost aggregators $A_\wedge$ and $A_\Box$. These functions can then be instantiated in a domain dependent manner to get the optimal algorithm for the domain definition of minimality. This approach has been used before in the context of A* and AO* [14].

We assume given a *base cost function* $k : P \to \mathbb{R}^+$, as well as a *conjunction cost aggregator* $A_\wedge : \mathbb{N}^{\mathbb{R}^+ \cup \{\infty\}} \to \mathbb{R}^+ \cup \{\infty\}$ and a *box modal cost aggregator* $A_\Box : \Sigma \times \mathbb{N}^{\mathbb{R}^+ \cup \{\infty\}} \to \mathbb{R}^+ \cup \{\infty\}$, where $\mathbb{N}^{\mathbb{R}^+ \cup \{\infty\}}$ denotes the set of multisets of $\mathbb{R}^+ \cup \{\infty\}$.

We assume the aggregators are increasing in the sense that adding elements to the input increases the cost. For all costs $x \leq y \in \mathbb{R}^+ \cup \{\infty\}$, multisets of costs $X \in \mathbb{N}^{\mathbb{R}^+ \cup \{\infty\}}$, and for all agents a, we have for the conjunction cost aggregator $A_\wedge(X) \leq A_\wedge(\{x\} \cup X) \leq A_\wedge(\{y\} \cup X)$, and for the box aggregator $A_\Box(a, X) \leq A_\Box(a, \{x\} \cup X) \leq A_\Box(a, \{y\} \cup X)$.

We further assume that aggregating infinite costs results in infinite costs and that aggregating finite numbers of finite costs results in finite costs. For all costs $x \in \mathbb{R}^+$, multisets of costs $X \in \mathbb{N}^{\mathbb{R}^+ \cup \{\infty\}}$, and for all agents a, $A_\wedge(\{\infty\}) = A_\Box(a, \{\infty\}) = \infty$ and that $A_\wedge(X) < \infty \Rightarrow A_\wedge(\{x\} \cup X) < \infty$ and $A_\Box(a, X) < \infty \Rightarrow A_\Box(a, \{x\} \cup X) < \infty$.

Note that in our presentation, there is no cost to a negation. The justification is that we want a proof aggregating over a disjunction to cost as much as a disproof aggregating over a conjunction with children of the same cost, without having to include the disjunction and the diamond operator in the base syntax.

Given k, $A_\wedge$, and $A_\Box$, it is possible to define the *global cost function* for a (dis)proof as shown in Table 1.

Table 1. Cost K of a proof or a disproof for a node n as a function of the base cost function k and the aggregators $A_\wedge$ and $A_\Box$. C is the set of children of n.

Label of n	Children of n	$K(n)$
(q, p)	$\emptyset$	$k(p)$
$(q, \neg\phi)$	$\{c\}$	$K(c)$
$(q, \phi_1 \wedge \phi_2)$	C	$A_\wedge(\{K(c) \mid c \in C\})$
$(q, \Box_a\, \phi)$	C	$A_\Box(a, \{K(c) \mid c \in C\})$

Example 1. Suppose we are interested in the nested depth of the $\Box$ operators in the (dis)proof. Then we define $k = 0$, $A_\wedge = \max$, and $A_\Box(a, X) = 1 + \max X$ for all a.

Example 2. Suppose we are interested in the number of atomic queries to the underlying system (the GA). Then we define $k = 1$, $A_\wedge(X) = \sum X$, and $A_\Box(a, X) = \sum X$ for all a.

We define two *heuristic* functions I and J to estimate the minimal amount of interaction needed with the underlying system to say anything about a formula ϕ. These functions are defined in Table 2, $I(\phi)$ is a lower bound on the minimal amount of interaction to prove ϕ and $J(\phi)$ is a lower bound on the minimal amount of interaction to disprove ϕ.

Table 2. Definition of the heuristic functions I and J

Shape of ϕ	$I(\phi)$	$J(\phi)$
p	$k(p)$	$k(p)$
$\neg\phi'$	$J(\phi')$	$I(\phi')$
$\phi_1 \wedge \phi_2$	$A_\wedge(\{I(\phi_1), I(\phi_2)\})$	$\min_{i\in\{1,2\}} A_\wedge(\{J(\phi_i)\})$
$\Box_a\,\phi'$	$A_\Box(a, \emptyset)$	$A_\Box(a, \{J(\phi')\})$

The heuristics I and J are *admissible*, that is, they never overestimate the cost of a (dis)proof.

Proposition 1. *Given a formula ϕ, for any state q, for any proof n that $q \models \phi$ (resp. disproof), $I(\phi) \leq K(n)$ (resp. $J(\phi) \leq K(n)$).*

3.2 Best First Search Framework

We inscribe the MPS algorithm in a best first search framework inspired by game tree search. We then specify a function for initializing the leaves, a function to update tree after a leaf has been expanded, a selection function to decide which part of the tree to expand next, and a stopping condition for the overall algorithm.

Algorithm 1 develops an exploration tree for a given state q and formula ϕ. To be able to orient the search efficiently towards proving or disproving the model checking problem $q \models \phi$ instead of just exploring, we need to attach additional information to the nodes beyond their (state, formula) label. This information takes the form of two *effort* numbers, called the *minimal proof number* and *minimal disproof number*. Given a node n associated with a pair (q, ϕ), the minimal proof number of n, MPN(n), is an indication on the cost of a proof for $q \models \phi$. Conversely, the minimal disproof number of n, MDN(n), is an indication on the cost of a disproof for $q \models \phi$. For a more precise relationship between MPN(n) and the cost of a proof see Prop. 6.

The algorithm stops when the minimal (dis)proof number reaches ∞ as it corresponds to the exploration tree containing a (dis)proof of minimal cost (see Prop. 4).

The values for the effort numbers in terminal leaves and in newly created leaves are defined in Table 3. The values for the effort numbers of an internal node as a function of its children are defined in Table 4. Finally, the selection procedure base on the effort numbers to decide how to descend the global tree

```
bfs(state $q$, formula $\phi$)
$r \leftarrow$ new node with label $(q, \phi)$;
$r$.info $\leftarrow$ init-leaf($r$);
$n \leftarrow r$;
while $r$ is not solved do
    while $n$ is not a leaf do
        $n \leftarrow$ select-child($n$);
    extend($n$);
    $n \leftarrow$ backpropagate($n$);
return $r$

extend(node $n$)
switch on the label of $n$ do
    case $(q, p)$
        $n$.info $\leftarrow$ info-term($n$);
    case $(q, \phi_1 \wedge \phi_2)$
        $n_1 \leftarrow$ new node with label $(q, \phi_1)$;
        $n_2 \leftarrow$ new node with label $(q, \phi_2)$;
        $n_1$.info $\leftarrow$ init-leaf($n_1$);
        $n_2$.info $\leftarrow$ init-leaf($n_2$);
        Add $n_1$ and $n_2$ as children of $n$;
    case $(q, \neg\phi_1)$
        $n' \leftarrow$ new node with label $(q, \phi_1)$;
        $n'$.info $\leftarrow$ init-leaf($n'$);
        Add $n'$ as a child of $n$;
    case $(q, \Box_a \phi_1)$
        foreach $q'$ in $\{q', q \xrightarrow{a} q'\}$ do
            $n' \leftarrow$ new node with label $(q', \phi_1)$;
            $n'$.info $\leftarrow$ init-leaf($n'$);
            Add $n'$ as child of $n$;

backpropagate(node $n$)
new_info $\leftarrow$ update($n$);
if $new\_info = n.info \vee n = r$ then return $n$;
else
    $n$.info $\leftarrow$ new_info;
    return backpropagate($n.parent$)
```

Algorithm 1. Generic pseudo-code for a best-first search algorithm

Table 3. Values for terminal nodes and initial values for leaves

	Node label	MPN	MDN
`info-term`	(q, p) where $p \in \pi(q)$	$k(p)$	∞
	(q, p) where $p \notin \pi(q)$	∞	$k(p)$
`init-leaf`	(q, ϕ)	$I(\phi)$	$J(\phi)$

Table 4. Determination of values for internal nodes

Node label	Children	MPN	MDN
$(q, \neg\phi)$	$\{c\}$	$\text{MDN}(c)$	$\text{MPN}(c)$
$(q, \phi_1 \wedge \phi_2)$	C	$A_\wedge(\{\text{MPN}(c) \mid c \in C\})$	$\min_C A_\wedge(\{\text{MDN}\})$
$(q, \Box_a \phi)$	C	$A_\Box(a, \{\text{MPN}(c) \mid c \in C\})$	$\min_C A_\Box(a, \{\text{MDN}\})$

Table 5. Selection policy

Node label	Children	Chosen child
$(q, \neg\phi)$	$\{c\}$	c
$(q, \phi_1 \wedge \phi_2)$	C	$\arg\min_C A_\wedge(\{\text{MDN}\})$
$(q, \Box_a \phi)$	C	$\arg\min_C A_\Box(a, \{\text{MDN}\})$

is given in Table 5. The stopping condition, Table 3, 4, and 5, as well as Alg. 1 together define Minimal Proof Search.

The `backpropagate` procedure implements a small optimization known as the *current node enhancement* [1]. Basically, if the information about a node n are not changed, then the information about the ancestors of n will not change either and so the next descend will reach n. Thus, it is possible to shortcut the process and start the next descent at n directly.

4 Properties of MPS

Before studying some theoretical properties of (dis)proofs, minimal (dis)proof numbers, and MPS, let us point out that for any exploration tree, not necessarily produced by MPS, we can associate to each node an MPN and an MDN by using the initialization described in Table 3 and the heredity rule described in Table 4.

4.1 Correctness of the Algorithm

The first property we want to prove about MPS is that the descent does not get stuck in a solved subtree.

Proposition 2. *For any internal node n with finite effort numbers, the child c selected by the procedure described in Table 5 has finite effort numbers.* $\text{MPN}(n) \neq \infty$ *and* $\text{MDN}(n) \neq \infty$ *imply* $\text{MPN}(c) \neq \infty$ *and* $\text{MDN}(c) \neq \infty$.

As a result, each descent ends in a non solved leaf. Either the associated formula is of the form p and the leaf gets solved, or the leaf becomes an internal node and its children are associated with structurally smaller formulas.

Proposition 3. *The MPS algorithm terminates in a finite number of steps.*

Since the algorithm terminates, we know that the root of the tree will eventually be labelled with a infinite minimal (dis)proof number and thus will be *solved*. It remains to be shown that this definition of a solved tree coincides with containing (dis)proof starting at the root.

Proposition 4. *If a node n is associated with (q, ϕ), then* $\text{MDN}(n) = \infty$ *(resp.* $\text{MPN}(n) = \infty$*) if and only if the tree corresponding to n contains a proof (resp. disproof) that $q \models \phi$ as a subtree with root n.*

Theorem 1. *The MPS algorithm takes a formula ϕ and a state q as arguments and returns after a finite number of steps an exploration tree that contains a (dis)proof that $q \models \phi$.*

4.2 Minimality of the (Dis)Proofs

Now that we know that MPS terminates and returns a tree containing a (dis)proof, we need to prove that this (dis)proof is of minimal cost.

The two following propositions can be proved by a simple structural induction on the exploration tree, using Table 3 and the admissibility of I and J for the base case and Table 4 for the inductive case.

Proposition 5. *If a node n is solved, then the cost of the contained (dis)proof is given by the minimal (dis)proof number of n.*

Proposition 6. *If a node n is associated with (q, ϕ), then for any proof m (resp. disproof) that $q \models \phi$, we have* $\text{MPN}(n) \leq K(m)$ *(resp.* $\text{MDN}(n) \leq K(m)$*).*

Since the aggregators for the cost function are increasing functions, then $\text{MPN}(n)$ and $\text{MDN}(n)$ are non decreasing as we add more nodes to the tree n.

Proposition 7. *For each disproved internal node n in a tree returned by the MPS algorithm, at least one of the children of n minimizing the* MDN *is disproved.*

Combining Prop. 5, 6, and 7, we get the following theorem.

Theorem 2. *The tree returned by the MPS algorithm contains a (dis)proof of minimal cost.*

4.3 Optimality of the Algorithm

The MPS algorithm is not optimal in the most general sense because it is possible to have better algorithm in some cases by using transpositions, domain knowledge, or logical reasoning on the formula to be satisfied.

For instance, take $\phi_1 = \Diamond_a(p \wedge \neg p)$ and ϕ_2 some non trivial formula satisfied in a state q. If we run the MPS algorithm to prove that $q \models \phi_1 \vee \phi_2$, it will explore at least a little the possibility of proving $q \models \phi_1$ before finding the minimal proof through ϕ_2. We can imagine that a more "clever" algorithm would recognize that ϕ_1 is never satisfiable and would directly find the minimal proof through ϕ_2.

Another possibility to outperform MPS is to make use of transpositions to shortcut some computations. MPS indeed explores structures according to the MMLK formula shape, and it is well-known in modal logic that bisimilar structures cannot be distinguished by MMLK formulas. It is possible to express an algorithm similar to MPS that would take transpositions into account, adapting ideas from PNS [15,12,8]. We chose not to do so in this article for simplicity reasons.

Still, MPS can be considered optimal among the programs that do not use reasoning on the formula itself, transpositions or domain knowledge. Stating and proving this property formally is not conceptually hard, but we have not been able to find simple definitions and a short proof that would not submerge the reader with technicalities. Therefore we decided only to describe the main ideas of the argument from a high-level perspective.

Definition 6. *A pair (q', ϕ') is* similar *to a pair (q, ϕ) with respect to an exploration tree n associated with (q, ϕ) if q' can substitute for q and ϕ' for ϕ in n.*

Let n associated with (q, ϕ) be an exploration tree with a finite MPN (resp. MDN), then we can construct a pair (q', ϕ') similar to (q, ϕ) with respect to n such that there is a proof that $q' \models \phi'$ of cost exactly $\text{MPN}(n)$ (resp. a disproof of cost $\text{MDN}(n)$).

Definition 7. *An algorithm A is* purely exploratory *if the following holds. Call n the tree returned by A when run on a pair (q, ϕ). For every pair (q', ϕ') similar to (q, ϕ) with respect to n, running A on (q', ϕ') returns a tree structurally equivalent to n.*

Depth first search, if we were to return the explored tree, and MPS are both examples of purely exploratory algorithms.

Proposition 8. *If a purely exploratory algorithm A is run on a problem (q, ϕ) and returns a solved exploration tree n where* $\text{MPN}(n)$ *(resp.* $\text{MDN}(n)$*) is smaller than the cost of the contained proof (resp. disproof), then we can construct a problem (q', ϕ') similar with respect to n such that A will return a structurally equivalent tree with the same proof (resp. disproof) while there exists a proof of cost* $\text{MPN}(n)$ *(resp. disproof of cost* $\text{MDN}(n)$*).*

Note that if the cost of a solved exploration tree n is equal to its MPN (resp. MDN), then we can make MPS construct a solved shared root subtree of n just by influencing the tie-breaking in the selection policy described in Table 5.

Theorem 3. *If a purely exploratory algorithm A returns a solved exploration tree n, either this tree (or a subtree) can be generated by MPS or A is not guaranteed to return a tree containing a (dis)proof of minimal cost on all possible inputs.*

5 Conclusion and Discussion

We presented Minimal Proof Search (MPS), a model checking algorithm for MMLK. MPS has been proven correct, and it has been proved that the (dis)proof returned by MPS was minimizing a generic cost function. The only assumption about the cost function is that it is defined recursively using increasing aggregators. Finally, we have shown that MPS was optimal among the purely exploratory model checking algorithms for MMLK.

Nevertheless, the proposed approach has a few limitations. MPS is a best first search algorithm and is memory intensive; the cost functions addressed in the article cannot represent variable edge cost; and MPS cannot make use of transpositions in its present form. Still, we think that these limitations can be overcome in future work.

We envision a depth-first adaptation of MPS similar to Nagai's transformation of PNS into DFPN. Alternatively, we can draw inspiration from PN^2 [1] and replace the heuristic functions I and J by a nested call to MPS, leading to an MPS^2 algorithm trading time for memory. These two alternative algorithms would directly inherit the correctness and minimality theorems for MPS. The optimality theorem would also transpose in the depth-first case, but it would not be completely satisfactory. Indeed, even though the explored tree will still be admissibly minimal, several nodes inside the tree will have been forgotten and re-expanded multiple times. This trade-off is reminiscent of the one between A* and its depth-first variation IDA* [9].

Representing problems with unit edge costs is already possible within the framework presented in Sect. 3.1. It is not hard to adapt MPS to the more general case as we just need to replace the agent labels on the transitions with (agent, cost) labels. This more general perspective was not developed in this article because the notation would be heavier while it would not add much to the intuition and the general understanding of the ideas behind MPS.

Finding minimal (dis)proofs while taking transpositions into account is more challenging because of the double count problem. While it is possible to obtain a correct algorithm returning minimal (dis)proofs by using functions based on propagating sets of individual costs instead of real values in Sect. 3.1, similarly to previous work in PNS [12], such a solution would hardly be efficient in practice and would not necessarily be optimal. The existing literature on PNS and transpositions can certainly be helpful in addressing efficient handling of transpositions in MMLK model checking [15,12,8].

Beside evaluating and improving the practical performance of MPS, future work can also study to which extent the ideas presented in this article can be applied to model checking problems in more elaborate modal logics and remain tractable.

Acknowledgement. We would like to thank the reviewers for their helpful comments and for pointing out Cleaveland's related work [6].

References

1. Allis, L.V., van der Meulen, M., van den Herik, H.J.: Proof-Number Search. Artificial Intelligence 66(1), 91–124 (1994)
2. Blackburn, P., De Rijke, M., Venema, Y.: Modal Logic, vol. 53. Cambridge University Press (2001)
3. Clarke, E.M., Grumberg, O., McMillan, K.L., Zhao, X.: Efficient generation of counterexamples and witnesses in symbolic model checking. In: Proceedings of the 32nd Annual ACM/IEEE Design Automation Conference, pp. 427–432. ACM (1995)
4. Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. The MIT Press (1999)
5. Clarke, E.M., Jha, S., Lu, Y., Veith, H.: Tree-like counterexamples in model checking. In: Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science, pp. 19–29. IEEE (2002)
6. Cleaveland, R.: Tableau-based model checking in the propositional mu-calculus. Acta Informatica 27(8), 725–747 (1989)
7. Groce, A., Visser, W.: What went wrong: Explaining counterexamples. In: Model Checking Software, pp. 121–136 (2003)
8. Kishimoto, A., Müller, M.: A solution to the GHI problem for depth-first proof-number search. Information Sciences 175(4), 296–314 (2005)
9. Korf, R.E.: Depth-first iterative-deepening: an optimal admissible tree search. Artificial Intelligence 27(1), 97–109 (1985)
10. Lange, M.: Model checking propositional dynamic logic with all extras. Journal of Applied Logic 4(1), 39–49 (2006)
11. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 682–688. Springer, Heidelberg (2009)
12. Müller, M.: Proof-Set Search. In: Schaeffer, J., Müller, M., Björnsson, Y. (eds.) CG 2002. LNCS, vol. 2883, pp. 88–107. Springer, Heidelberg (2003)
13. Nagai, A.: Df-pn algorithm for searching AND/OR trees and its applications. Ph.D. thesis, University of Tokyo (December 2001)
14. Pearl, J.: Heuristics: intelligent search strategies for computer problem solving. Addison Wesley Publishing Company (1984)
15. Schijf, M., Allis, L.V., Uiterwijk, J.W.: Proof-number search and transpositions. ICCA Journal 17(2), 63–74 (1994)
16. Shoham, Y., Leyton-Brown, K.: Multiagent systems: Algorithmic, game-theoretic, and logical foundations. Cambridge University Press (2009)
17. van der Hoek, W., Wooldridge, M.J.: Model checking knowledge and time. In: Bošnački, D., Leue, S. (eds.) SPIN 2002. LNCS, vol. 2318, pp. 95–111. Springer, Heidelberg (2002)
18. van Ditmarsch, H.P., van der Hoek, W., Kooi, B.P.: Concurrent dynamic epistemic logic for MAS. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 201–208. ACM (2003)

Building an Epistemic Logic for Argumentation*

François Schwarzentruber[1], Srdjan Vesic[2], and Tjitze Rienstra[2]

[1] IRISA / ENS Cachan
francois.schwarzentruber@bretagne.ens-cachan.fr
[2] Computer Science and Communication
University of Luxembourg
{srdjan.vesic,tjitze.rienstra}@uni.lu

Abstract. In this paper, we study a multi-agent setting in which each agent is aware of a set of arguments. The agents can discuss and persuade each other by putting forward arguments and counter-arguments. In such a setting, what an agent will do, i.e. what argument she will utter, may depend on what she knows about the knowledge of other agents. For example, an agent does not want to put forward an argument that can easily be attacked, unless she believes that she is able to defend her argument against possible attackers. We propose a logical framework for reasoning about the sets of arguments owned by other agents, their knowledge about other agents' arguments, etc. We do this by defining an epistemic logic for representing their knowledge, which allows us to express a wide range of scenarios.

1 Introduction

Argumentation is the interdisciplinary study of how conclusions can be reached through logical reasoning. In the area of artificial intelligence, argumentation is usually seen as a reasoning approach based on construction and evaluation of arguments. The work of Pollock [10], Vreeswijk [16], and Simari and Loui [13] gave rise to other propositions on how to conceptualise this process. Nowadays, much research on the topic of argumentation is based on the argumentation theory proposed by Dung [4]. It allows to abstract from the origin and the structure of arguments, by representing an argumentation system as a directed graph, whose vertices correspond to arguments and arcs to attacks between them.

It is common that argumentation takes place between multiple agents, having different information and different goals. In such a setting, agents present arguments in order to persuade other agents. Their goal is often to make a certain argument accepted (or rejected). Some efforts were done in studying argumentation dialogues [11,12] by applying game-theoretic notions. However, those approaches do not allow for reasoning about agents' knowledge, which is one of the essential factors in such a setting and influences agent's behaviour in a major way. For example, when deciding which argument

* SV was funded by the National Research Fund, Luxembourg. His work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. This Programme is supported by the Marie Curie Co-funding of Regional, National and International Programmes (COFUND) of the European Commission.

to utter, an agent may take into account his beliefs about whether another agent has an attacker of that argument. Moreover, an agent may want to reason about the knowledge of another agent. For example: what should I do if he knows that I know that he does not have an attacker of this argument?

In this paper, we define a logical framework for this setting. To do so, we use the epistemic modal logic. We define a logic which allows to formalise a broad spectrum of scenarios concerning the knowledge of agents in form of arguments (e.g. attacks between them) but also the knowledge of agents about the knowledge of other agents, and so on. We also provide a method to speak about the fact that an agent is aware of the existence of an argument.

The remainder of the paper is organised as follows. Section 2 introduces the setting and stresses the importance of the notion of awareness. Section 3 provides a logic to express beliefs about awareness. Section 4 extends this logic for expressing beliefs about the structure of the argumentation graph. Section 5 provides a solution for expressing beliefs about properties of a given argument. The last section concludes.

2 Setting

Since we represent the basic knowledge of agents in form of arguments, we first introduce the notion of an argumentation theory [4] which is used in our formalisation.

Definition 1 (Argumentation system). *An argumentation framework is a pair $\mathcal{A} = (A, \rightsquigarrow)$ where A is a set of arguments and $\rightsquigarrow \subseteq A \times A$ a binary relation. For each pair $(a, b) \in \rightsquigarrow$, we say that a attacks b. We will also sometimes use notation $a \rightsquigarrow b$ instead of $(a, b) \in \rightsquigarrow$.*

We model a situation where a set of agents $\{1, \ldots, n\}$ have different knowledge (in terms of arguments) and beliefs (about the knowledge of other agents). We can model this situation in abstract argumentation theory by representing the arguments and the attack relation between them by what we will call a *big argumentation framework*. We denote this framework by $\mathtt{BAF} = (\mathcal{A}_{\mathcal{B}}, \rightsquigarrow_{\mathcal{B}})$. The big argumentation framework contains all arguments relevant to a particular discourse. Here we may imagine, for example, that BAF is constructed from all available knowledge and beliefs on a subject such as nuclear energy, and the issue of whether or not we should build more nuclear power plants, or instead close them. The knowledge and opinions in BAF may come, for example, from books, internet, scientific publications, but they may also be completely personal to an agent. Agents are resource bounded and are, in general, not aware of all arguments that belong to the BAF. An agent is aware of only those arguments that she has acquainted herself with, or that she has formed, in some way, on the basis of personal considerations or a priori knowledge.

We can thus represent the knowledge of an agent i by a set $A_i \subseteq \mathcal{A}_{\mathcal{B}}$ of arguments. We assume, however, that all agents use the same logical language in order to understand each other and that they agree on the attack relation. That is, for every pair of arguments $a, b \in \mathcal{A}_{\mathcal{B}}$, all agents agree on whether or not a is a valid counter-argument to b (or whether a attacks b). So we have a model where all arguments of a particular discourse, and their attack relations, are represented by the big argumentation framework $\texttt{BAF} = (\mathcal{A}_\mathcal{B}, \rightsquigarrow_\mathcal{B})$, and the knowledge of an agent i is represented by a set $A_i \subseteq \mathcal{A}_\mathcal{B}$. This induces, for an agent i, a framework $(A_i, \rightsquigarrow_i)$, with $\rightsquigarrow_i = \rightsquigarrow|_{A_i}$. Note that the formalisation we use, namely the hypothesis that there exists a big argumentation framework and that agents are aware of some arguments from this framework is already present in argumentation literature [11,14]. In the rest of the paper, we develop logics for reasoning in this setting.

3 The First Attempt of an Epistemic Argumentation Logic

In this section, we propose a framework for representing the fact that different agents are aware of different arguments. Let $AGT = \{1, \ldots, n\}$ be a finite set of agents. The language of this logic, denoted by $\mathcal{L}_1$ is generated by the following BNF:

$$\varphi ::= owns(i, a) \mid \neg\varphi \mid \varphi \wedge \varphi \mid B_i\varphi$$

where $i \in AGT$ is an agent, and $a \in \mathcal{A}_\mathcal{B}$ is an argument. For a finite set $S \subseteq \mathcal{A}_\mathcal{B}$, with $S = \{a_1, \ldots, a_k\}$, we define an abbreviation $owns(i, S) \overset{\texttt{def}}{=} owns(i, a_1) \wedge \ldots \wedge owns(i, a_k)$.

A formula $owns(i, a)$ means that agent i is aware of the argument a. The meaning of $\neg$, $\wedge$ and derived connectives $\vee$, $\rightarrow$, $\leftrightarrow$ are as usual. A formula $B_i\varphi$ means that agent i believes that φ holds. Some examples of statements that we can express are:

– $owns(1, \{a, b, c\}) \wedge B_1 owns(2, \{a, b\})$ (Agent 1 is aware of a, b and c and believes that agent 2 is aware of a and b.)
– $owns(1, \{a\}) \wedge B_1 B_2 \neg owns(1, \{a\})$ (Agent 1 is aware of a but believes that agent 2 believes he is not.)

The interpretation of the language is based on Kripke structures where states describe possible configurations of argument awareness for all agents. Formally, a state w and an agent i map to a set $D_i \subseteq A$, where D_i is the set of arguments that agent i is aware of in state w. For every agent i, the accessibility relation R_i captures the 'considers possible' relation. Formally:

Definition 2. *An $\mathcal{L}_1$-epistemic argumentation model is a Kripke structure $\mathcal{M} = (W, R, D)$ where:*

– *W is a non-empty set of* states*;*
– *R maps each agent i to an* accessibility relation *R_i over W;*
– *D maps each world w and each agent i to a set of arguments $D_i(w)$ such that:*
  1. *for all agents i, for all $w, u \in W$, wR_iu implies $D_i(u) = D_i(w)$.*
  2. *for all agents i, j, for all $w, u \in W$, wR_iu implies $D_j(u) \subseteq D_i(w)$.*

We use the familiar interpretation of belief by taking every R_i to be a KD45 relation [8]. That is, R_i is

– serial: $\forall s \in W, \exists t \in W$ s.t. $t \in R_i(s)$,
– transitive: $\forall s, t, u \in W$, $t \in R_i(s)$ and $u \in R_i(t)$ implies $u \in R_i(s)$,
– and Euclidean: $\forall s, t, u \in W$, $t \in R_i(s)$ and $u \in R_i(s)$ implies $t \in R_i(u)$,

The truth conditions are as follows:

- $\mathcal{M}, w \models owns(i, a)$ iff $a \in D_i(w)$;
- $\mathcal{M}, w \models B_i\varphi$ iff for all $u \in R_i(w)$, we have $\mathcal{M}, u \models \varphi$;
- $\mathcal{M}, w \models \varphi \wedge \psi$ iff $\mathcal{M}, w \models \varphi$ and $\mathcal{M}, w \models \psi$;
- $\mathcal{M}, w \models \neg\varphi$ iff it is not the case that $\mathcal{M}, w \models \varphi$

The two conditions from Definition 2 crucially capture our intuition behind awareness of arguments, as described in the previous section. The first condition says that in every world an agent considers possible, she is aware of the same set of arguments that she is aware of in the actual world. This condition corresponds to the following 'argument awareness introspection' axioms:

- $owns(i, a) \rightarrow B_i owns(i, a)$;
- $\neg owns(i, a) \rightarrow B_i \neg owns(i, a)$

The second condition stipulates that, if an agent is not aware of an argument, she believes that no agent is aware of that argument. This condition corresponds to the following axiom:

- $\neg owns(i, a) \rightarrow B_i \neg owns(j, a)$.

Figure 1 shows a model $\mathcal{M}$ where $\mathcal{M}, s \models owns(1, \{a, b, c\}) \wedge B_1 owns(2, \{a, b\})$. Notice that agent 1 has no belief as to whether or not agent 2 knows c, and agent 2 has no beliefs as to whether agent 1 knows a.

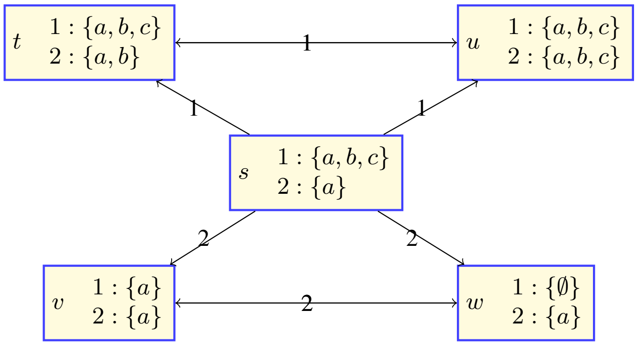


Fig. 1. An $\mathcal{L}_1$-epistemic argumentation logic model

We say that φ is $\mathcal{L}_1$-satisfiable iff there exists an $\mathcal{L}_1$-argumentation epistemic model $\mathcal{M}$ and a world w such that $\mathcal{M}, w \models \varphi$. The satisfiability problem of a formula of $\mathcal{L}_1$ is the following decision problem:

- input: a formula $\varphi \in \mathcal{L}_1$;
- output: yes iff the formula φ is $\mathcal{L}_1$-satisfiable.

Having an algorithm to solve the satisfiability problem enables us to check consistency automatically. We now study the computational properties of the satisfiability problem of a formula of $\mathcal{L}_1$.

Theorem 1. *If there are more than 3 agents, the satisfiability problem of a formula of $\mathcal{L}_1$ is PSPACE-hard.*

Proof. When there is more than three agents, we can embed the satisfiability problem of $KD45_2$ into the satisfiability problem of a formula of $\mathcal{L}_1$. Let φ be a formula of $KD45_2$. Let i, j be two distinct agents such all agents appearing in φ are in $\{i, j\}$. Let k be a third distinct agent. We suppose the set of arguments to be the set of all propositions appearing in φ. We then define a polynomial translation tr from the $KD45_2$ language to $\mathcal{L}_1$ as follows:

– $tr(p) = owns(k, p)$;
– $tr(B_i\varphi) = B_i tr(\varphi)$;
– $tr(B_j\varphi) = B_j tr(\varphi)$.

We have that φ is $KD45_n$ satisfiable iff $tr(\varphi)$ is satisfiable in an epistemic argumentation model. Note that we must take care to verify the condition of Definition 2 in the sense ‘left to right’. We need the extra agent k in order to be able to construct such a epistemic argumentation model. Technical details are left to the reader.

Theorem 2. *The satisfiability problem of a formula of $\mathcal{L}_1$ is in PSPACE.*

Proof. We can embed $\mathcal{L}_1$ into $KD45_n$ and call an optimal procedure for $KD45_n$ plus *distributed belief* which is in PSPACE [8]. Let φ be a formula of $\mathcal{L}_1$. Let m be the modal depth of the formula φ. The embedding works as follows. We add also an operator called *distributed belief* in the language, noted B_{dist}. This operator enables us to express the properties of Definition 3 up to the depth n with a formula of polynomial size in the length of φ. The semantics is defined as follows:

– $\mathcal{M}, w \models B_{dist}\psi$ iff for all $i \in AGT$, for all $u \in R_i(w)$ we have $\mathcal{M}, u \models \psi$.

We denote by $\boldsymbol{B_{dist}}\chi$ the formula $\chi \wedge B_{dist}\chi \wedge B^2_{dist}\chi \wedge \cdots \wedge B^m_{dist}\chi$. It corresponds to common knowledge up to level n, where n is the modal depth of φ. We then consider $tr(\varphi)$ as the conjunction of φ and the following formulas:

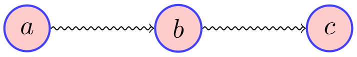
– $\boldsymbol{B_{dist}}(owns(i, a) \rightarrow B_i owns(i, a))$
– $\boldsymbol{B_{dist}}(\neg owns(i, a) \rightarrow B_i \neg owns(i, a))$
– $\boldsymbol{B_{dist}}(\neg owns(i, a) \rightarrow B_i \neg owns(j, a))$.

In that way, $tr(\varphi)$ imposes the $KD45_n$-model to satisfy the properties of Definition 3 up to level m. The formula φ is satisfiable in an epistemic argumentation model iff the formula $tr(\varphi)$ is satisfiable in $KD45_n$ plus distributed belief, where constructions of the form $owns(i, a)$ are considered as atomic propositions in $KD45_n$.

4 Expressing Belief about Properties of Arguments

The formalisation presented until now is only the first step towards describing an argument-based dialogue. There are still simple facts that cannot be expressed in the proposed logic. For example, imagine a non-expert person having an idea in some area. She can believe that her idea is interesting, and she is not aware of any attacker of her argument, but she also believes that there is an argument (from an expert) attacking her argument. The problem here is that to express a property about an argument one is not aware of. The next example formalises this consideration.

Example 1. Let us consider the following BAF:



Imagine that agent 1 is not an expert and she has only argument b. The framework proposed in the previous section does not allow to represent a situation in which she has no beliefs about whether this argument is attacked or not. Namely, according to Definition 2, for every model $\mathcal{M}$, for every world w in $\mathcal{M}$ such that agent 1 has exactly the argument b, i.e. where $D_1(w) = \{b\}$, for every world u in $\mathcal{M}$ such that wR_1u, for every agent j, it holds that $a \notin D_j(u)$. That is, in every world of every possible model where agent 1 is aware, agent 1 believes that b is not attacked.

The previous example shows the formalism from the previous section is not expressive enough since it cannot represent the situation where an agent believes that there exists an attacker of one of her arguments, without being able to construct an attacker herself. We start by defining a new language which is richer and allows to speak about attacks between arguments. The solution we propose consists in mixing epistemic modal logic (that we proposed in the previous section) and a logical framework to speak about argumentation graphs, initially proposed by Grossi [7]. Let ATM be a countable set of atomic propositions. The new language is defined as a combination of those two languages:

$$\varphi ::= \langle U \rangle \psi \mid \neg\varphi \mid \varphi \wedge \varphi \mid B_i\varphi$$
$$\psi ::= p \mid \neg\psi \mid \psi \wedge \psi \mid isarg(a) \mid ownedby(i) \mid [attacks]\psi \mid [is\_attacked]\psi$$

where $p \in ATM$, a is an argument of the BAF and i is an agent. We define the language $\mathcal{L}_2$ as the set of formulas obtained with the rule φ. φ-formulas are epistemic modal logic formulas expressing beliefs about facts. The construction $\langle U \rangle \psi$ is read as 'there exists an argument verifying the property ψ'. Then a ψ-formula describes the property of a given argument. Propositions p are used to describe property of arguments, as for instance 'the current argument is about politics'. $isarg(a)$ states that the argument of which we speak now is argument a. $ownedby(i)$ means that the current argument is owned by i. The construction $[attacks]\psi$ means that all arguments that the current argument attacks verify the property ψ. The construction $[is\_attacked]\psi$ means that all arguments that the current argument is attacked by verify the property ψ. We define the following abbreviations: $\langle attacks \rangle\psi = \neg[attacks]\neg\psi$ and $\langle is\_attacked \rangle\psi = \neg[is\_attacked]\neg\psi$.

Example 2. Now, we can say that agent 1 does not have beliefs about whether argument b is attacked or not. We can write this as 'agent 1 does not believe that b is attacked and agent 1 does not believe that b is not attacked': $\neg B_1(\langle U\rangle(isarg(b) \wedge \langle is\_attacked\rangle\top)) \wedge \neg B_1(\langle U\rangle(isarg(b) \wedge \neg\langle is\_attacked\rangle\top))$. As another example, take the following formula which says that agent 1 believes that there exists an argument about global warming owned by the second agent: $B_1(\langle U\rangle(global\_warming \wedge ownedby(2)))$. We can also say that agent 1 does not have an attacker of b but agent 1 believes that agent 2 has an attacker of b on the subject of global warming. It is written as:
$\langle U\rangle(isarg(b) \wedge [is\_attacked]\neg ownedby(1)) \wedge B_1 isarg(b) \wedge \langle is\_attacked\rangle (ownedby(2) \wedge global\_warming)$.

We now define how to interpret formulas of language $\mathcal{L}_2$.

Definition 3. *A $\mathcal{L}_2$−epistemic argumentation model is a Kripke structure $\mathcal{M} = (W, R, \mathcal{A})$ based on a BAF $= (\mathcal{A}_\mathcal{B}, \rightsquigarrow_\mathcal{B})$ where:*

- *W is a non-empty set of epistemic worlds;*
- *R maps each agent to a serial, transitive and Euclidean relation over W;*
- *$\mathcal{A}$ maps each world w to a labelled argumentation graph $\mathcal{A}_w = (A_w, \rightsquigarrow_w, L_w)$ where:*
  - *A_w is a finite subset of $\mathcal{A}_\mathcal{B} \cup \{?_0, ?_1, \dots\}$;*
  - *$\rightsquigarrow_w \subseteq A_w \times A_w$ is a binary relation such that for $a, b \in \mathcal{A}_\mathcal{B}$, $a \rightsquigarrow_\mathcal{B} b$ if and only if $a \rightsquigarrow_w b$;*
  - *L_w is a map from A_w to $2^{AGT \cup ATM}$.*

Furthermore, we impose:

1. *for all agents i, for all $w, u \in W$, wR_iu implies that $\{a \in A_u \cap \mathcal{A}_\mathcal{B} \mid i \in L_u(a)\} = \{a \in A_w \cap \mathcal{A}_\mathcal{B} \mid i \in L_w(a)\}$;*
2. *for all agents i, for all $w, u \in W$, wR_iu implies that $\{a \in A_u \cap \mathcal{A}_\mathcal{B}\} \subseteq \{a \in A_w \cap \mathcal{A}_\mathcal{B} \mid i \in L_w(a)\}$.*

An example of a model is depicted in Figure 2. The model is still a Kripke model but now, each world w contains an argumentation graph $\mathcal{A}_w = (A_w, \rightsquigarrow_w)$. Each argument of A_w is either an argument from $\mathcal{A}_\mathcal{B}$, or an element of a set $\{?_0, ?_1, \dots\}$. A "question mark argument" denotes an argument the agent is not aware of, that is to say, they can not argue by using it, but they can imagine its existence. The idea is to represent facts like "there is an attacker of argument a" without being able to identify an actual attacker of this argument. $\rightsquigarrow_w$ is the attack relation in $\mathcal{A}_w$. $\rightsquigarrow_w$ is compatible with the attack relation of the BAF in the following sense: if two arguments a and b of the BAF appear in the argumentation graph $\mathcal{A}_w$ of the world w, then the attack relation in $\mathcal{A}_w$ is the same as in the BAF. Intuitively, it says that there is no uncertainty concerning the attack of arguments of the BAF [11]. L is a valuation function that specifies the atomic properties of a (this can be anything, for example a subject of an argument) and the agents that own argument a, for all arguments $a \in A_w$. For example, in Figure 2, in worlds w and u, argument b, owned by agent 1 is about global warming. Condition (1) and condition (2) have the same meaning as condition (1) and (2) in Definition 2.

The truth conditions for φ-formulas are defined as follows:

- $\mathcal{M}, w \models B_i\varphi$ iff for all $u \in R_i(w)$, we have $\mathcal{M}, u \models \varphi$;
- $\mathcal{M}, w \models \langle U \rangle \psi$ iff there exists an argument $a \in A_w$ such that $\mathcal{A}_w, a \models \psi$.

The truth conditions for ψ-formulas are defined as follows:

- $\mathcal{A}_w, a \models p$ iff $p \in ATM$ and $p \in L_w(a)$;
- $\mathcal{A}_w, a \models isarg(b)$ iff $a = b$;
- $\mathcal{A}_w, a \models ownedby(i)$ iff $i \in$ *AGT* and $i \in L_w(a)$;
- $\mathcal{A}_w, a \models [attacks]\psi$ iff for all b such that $a \rightsquigarrow_w b$ we have $\mathcal{A}_w, b \models \psi$;
- $\mathcal{A}_w, a \models [is\_attacked]\psi$ iff for all b such that $b \rightsquigarrow_w a$ we have $\mathcal{A}_w, b \models \psi$;

Example 3 (Example 1 Cont.). The logic $\mathcal{L}_2$ is expressive enough to overcome problems of Example 1. Let $\alpha = \neg B_1(\langle U \rangle (isarg(b) \wedge \langle is\_attacked \rangle \top)) \wedge \neg B_1(\langle U \rangle (isarg(b) \wedge \neg \langle is\_attacked \rangle \top))$. This formula says that agent 1 does not have beliefs about whether argument b is attacked or not. Let $\mathcal{M}$ be the model from Figure 2, then $\mathcal{M}, w \models \alpha$. This means that in model $\mathcal{M}$ and world w, agent 1 does not have beliefs about whether argument b is attacked or not.

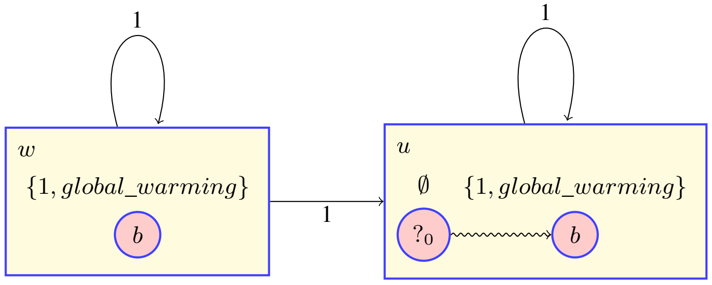


Fig. 2. An $\mathcal{L}_2$-epistemic argumentation logic model

The language $\mathcal{L}_2$ is a conservative extension of the language $\mathcal{L}_1$. Indeed, we can embed $\mathcal{L}_1$ into $\mathcal{L}_2$ by preserving validities with the following translation:

- $tr(owns(i, a)) = \langle U \rangle (ownedby(i) \wedge isarg(a))$.

We define the notion of $\mathcal{L}_2$-satisfiable formula. A formula φ is $\mathcal{L}_2$-satisfiable iff there exists a big argumentation framework $\mathtt{BAF} = (\mathcal{A}_\mathcal{B}, \rightsquigarrow_\mathcal{B})$ and a $\mathcal{L}_2$−epistemic argumentation model $\mathcal{M} = (W, R, \mathcal{A})$ based on BAF, and a world $\in W$, such that $\mathcal{M}, w \models \varphi$. In the same way, we define the satisfiability problem of $\mathcal{L}_2$.

Theorem 3. *Even if there are no occurrences of arguments in the formula we want to check, the satisfiability problem of $\mathcal{L}_2$ is EXPTIME-hard.*

Proof. The global satisfiability problem of modal logic K is defined as follows:

- input: two formulas φ, ψ where there is only one modal operator $\Box$;
- output: yes iff there exists a pointed model $\mathcal{M}, w$ for logic K such that $\mathcal{M}, u \models \varphi$ for all $u \in W$ and $\mathcal{M}, w \models \psi$.

It is EXPTIME-hard. We polynomially reduce the global satisfiability of logic K to the satisfiability of a formula of $\mathcal{L}_2$ in the following way: $\varphi \models_g \psi$ iff $\neg\langle U\rangle\neg tr(\varphi) \wedge \langle U\rangle tr(\psi)$ satisfiable where $tr(\Box\varphi) = [attacks]\varphi$. So the satisfiability problem of $\mathcal{L}_2$ is EXPTIME-hard. All of this works because of the presence of ?-arguments.

Now, concerning the satisfiability problem of $\mathcal{L}_2$, we have a tableau method decision procedure in [3] dealing with nominals (or arguments in our case), a K-operator $\Box$ (or $[attacks]$ in our case) universal modality. In [1], the author explains that the satisfiability problem of the hybrid logic where we add the converse operator and the universal operator is EXPTIME-complete. In our case, it means that given finite sets A, B of formulas of the form $\langle U\rangle\psi$, checking whether $\bigwedge_{\langle U\rangle\psi\in A}\langle U\rangle\psi \wedge \bigwedge_{\langle U\rangle\psi\in B}\neg\langle U\rangle\psi$ is satisfiable can be solved with an EXPTIME procedure. If we combine with a tableau procedure for $KD45_n$ [8] we obtain the following result.

Theorem 4. *The satisfiability problem of $\mathcal{L}_2$ is in EXPTIME.*

Proof. We give the idea of algorithm to solve the satisfiability problem of $\mathcal{L}_2$. Let us consider $\varphi \in \mathcal{L}_2$. Let $arg(\varphi)$ be the set of arguments that appear in the formula φ. We add also an operator called *distributed belief* to the language, denoted B_{dist}, in order to be able to express the properties of Definition 3. The semantics is defined as follows:

– $\mathcal{M}, w \models B_{dist}\varphi$ iff for all $i \in AGT$, for all $u \in R_i(w)$ we have $\mathcal{M}, u \models \varphi$.

We denote by $\boldsymbol{B_{dist}}\chi$ the formula $\chi \wedge B_{dist}\chi \wedge B^2_{dist}\chi \wedge \cdots \wedge B^m_{dist}\chi$. It corresponds to common knowledge up to level m, where m is the modal depth of φ. We denote by $\langle U\rangle SF(\varphi)$ the set of all subformulas of φ of the form $\langle U\rangle\varphi$. Let $Att \in 2^{arg(\varphi)^2}$. We define $T_{Att}(\varphi)$ as the following conjunction, which imposes the constraint of the Definition 3 up to depth m:

– φ;
– $\boldsymbol{B_{dist}}((\langle U\rangle isarg(a) \wedge ownedby(i)) \rightarrow B_i(\langle U\rangle isarg(a) \wedge ownedby(i)))$ for $i \in AGT$;
– $\boldsymbol{B_{dist}}((\langle U\rangle isarg(a) \wedge \neg ownedby(i)) \rightarrow B_i(\neg\langle U\rangle isarg(a)))$ for $i \in AGT$;
– $\boldsymbol{B_{dist}}([U](isarg(a) \rightarrow \neg isarg(b)))$ for all $a, b \in A$ such that $a \neq b$.
– $\boldsymbol{B_{dist}}(\langle U\rangle isarg(a) \wedge \langle U\rangle isarg(b)) \rightarrow \langle U\rangle(isarg(a) \rightarrow \langle attacks\rangle isarg(b))$ for all $(a, b) \in Att$;
– $\boldsymbol{B_{dist}}([U](isarg(a) \rightarrow [attacks]\neg isarg(b))$ for all $(a, b) \notin Att$;

Now we define the algorithm.

```
for $Att \in 2^{arg(\varphi)^2}$
    for $A, B \subseteq \langle U\rangle SF(T_{Att}(\varphi))$
        $PROP[A, B] =$
            $satsolver\_K^{U,converse}(\bigwedge_{\langle U\rangle\varphi\in A}\langle U\rangle\psi \wedge \bigwedge_{\langle U\rangle\varphi\in B}\neg\langle U\rangle\varphi))$
    endFor
    if $(modified\_KD45_n\_tableau\_method(T_{Att}(\varphi), PROP))$
        return sat
    endIf
endFor
return unsat
```

We loop on all possible attack relations Att over arguments that appear in the formula φ. Somehow, we browse all possible BAF. Our aim is then to check if $T_{Att}(\varphi)$ is satisfiable.

The first step consists in computing which subformulas of $T_{Att}(\varphi)$ of the form $\langle U\rangle\psi$ are consistent. Thus, $PROP[A, B]$ will contain 'sat' if $\bigwedge_{\langle U\rangle\psi\in A}\langle U\rangle\psi \wedge \bigwedge_{\langle U\rangle\psi\in B}\neg\langle U\rangle\psi$ is satisfiable and 'unsat' otherwise. The procedure $satsolver\_K^{U,converse}$ is an EXPTIME procedure to solve the satisfiability problem of K plus converse, plus universal modality and hybrid logic. Indeed, arguments are considered as nominals (it is impossible to have two different nodes labelled by the same arguments).

The second step is now to run a tableau method for $KD45_n$ logic on the formula $T_{Att}(\varphi)$. For that, we use the tableau method described in [8]. This tableau method runs as usual for the Boolean connectives and beliefs operators but it considers formulas of the form $\langle U\rangle\psi$ as atoms. We extend this tableau method with a new rule applied on a node w, when all the other rules have already been applied:

- Let A be the set of $\langle U\rangle\psi$ formula written in the node w. Let B be the set of $\langle U\rangle\psi$ such that $\neg\langle U\rangle\psi$. If $PROP[A, B]$ is unsat, we close the current tableau branch.

This modified version of the tableau method runs in PSPACE $\subseteq$ EXPTIME. So the global algorithm runs in EXPTIME. The proof of completeness and soundness of this algorithm are classical.

5 Expressing Properties of Arguments Containing Belief

The logical language developed in Section 4 is powerful enough to enable to speak about arguments without specifying them, but some facts about the framework we study in this paper still cannot be expressed in it. Consider the following example.

Example 4. Agent i owns a and believes that there exists an argument attacked by a which is owned by agent k, but agent j believes that this argument is not owned by k.

Languages $\mathcal{L}_1$ and $\mathcal{L}_2$ do not allow to express statement from the previous example.

We now present a preliminary but promising approach for expressing properties over arguments but also beliefs about those properties. The approach we present here is inspired by other applied logics mixing:

- knowledge and time [5], [9]: the author speaks about moments in time and knowledge about properties of the moment;
- time and space [2]: the author speaks about evolution of objects in the time.

The language $\mathcal{L}_3$ is defined by the following rule:

$$\varphi ::= p \mid isarg(a) \mid ownedby(i) \mid B_i\varphi \mid \langle U\rangle\varphi \mid [attacks]\varphi \mid [is\_attacked]\varphi$$

where $p \in ATM$, $a \in \mathcal{A}_{\mathcal{B}}$ and $i \in \{1, \ldots, n\}$ is an agent. In $\mathcal{L}_3$, we can mix doxastic operator B_i and speaking about arguments. The reading of $B_i\varphi$ is now 'agent i believes φ about the current argument'.

Example 5. The formula $\langle U\rangle(isarg(a) \wedge ownedby(i) \wedge B_i\langle attacks\rangle(ownedby(k) \wedge B_j\neg ownedby(k))$ is in $\mathcal{L}_3$ and captures the sentence of Example 4.

Definition 4. *$\mathcal{F} = (W, R)$ is a Kripke epistemic frame if and only if W is a non-empty set of possible worlds and $\mathcal{R}$ is a function mapping each agent i to a serial, transitive and Euclidean relation R_i over W. $\mathcal{MA} = (\mathcal{M}, \mathcal{A}, L)$ is a world/argument model if and only if:*

- *$\mathcal{M} = (W, R)$, is a Kripke epistemic frame;*
- *$\mathcal{A} = (A, \rightsquigarrow)$ is an argumentation graph such that $A = \mathcal{A}_{\mathcal{B}} \cup \{?_0, ?_1, \dots\}$ such that for all $a, b \in \mathcal{A}_{\mathcal{B}}$, $a \rightsquigarrow_{\mathcal{B}} b$ iff $a \rightsquigarrow b$;*
- *L maps all couples $(w, a) \in W \times A$ to elements of $2^{AGT \cup ATM}$.*

The truth conditions are:

- $\mathcal{MA}, (w, a) \models p$ iff $p \in ATM$ and $p \in L(w, a)$;
- $\mathcal{MA}, (w, a) \models isarg(b)$ iff $a = b$;
- $\mathcal{MA}, (w, a) \models ownedby(i)$ iff $i \in \{1, \dots, n\}$ and $i \in L(w, a)$;
- $\mathcal{MA}, (w, a) \models B_i\varphi$ iff for all $u \in R_i(w)$, we have $\mathcal{MA}, (u, a) \models \varphi$;
- $\mathcal{MA}, (w, a) \models \langle U\rangle\varphi$ iff there exists $b \in A$, we have $\mathcal{MA}, (w, b) \models \varphi$;
- $\mathcal{MA}, (w, a) \models [attacks]\varphi$ iff for all $b \in A$, $a \rightsquigarrow b$ implies $\mathcal{MA}, (w, b) \models \varphi$;
- $\mathcal{MA}, (w, a) \models [is\_attacked]\varphi$ iff for all $b \in A$, $b \rightsquigarrow a$ implies $\mathcal{MA}, (w, b) \models \varphi$.

Example 6. Figure 3 shows a model for the formula from Examples 4 and 5, namely: $\langle U\rangle(isarg(a) \wedge ownedby(i) \wedge B_i\langle attacks\rangle(ownedby(k) \wedge B_j\neg ownedby(k))$. We consider a model (on the right) built from the epistemic frame on the left and the `BAF` in the middle.

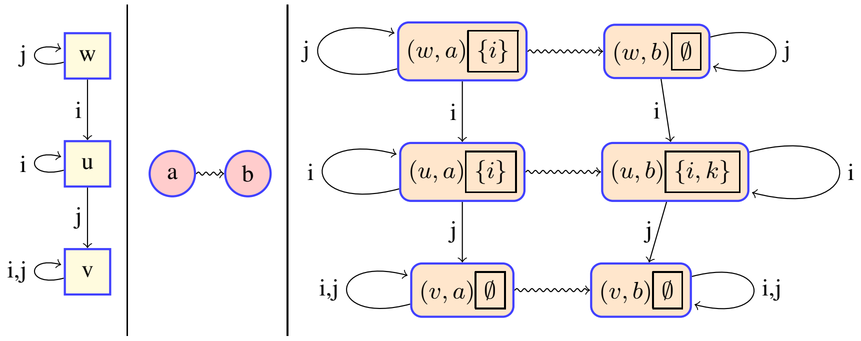


Fig. 3. A world / argument model

In previous sentence, the context is a couple (w, a) where w is a possible epistemic situation and a is an argument. Table 1 illustrates how to follow the nodes of a graph in Figure 3 when analysing the formula from our running example.

Note that this framework contains $KD45 \times K$. The satisfiability problem of $S5 \times K$ is NEXPTIME-complete [6, page 339]. We conjecture that the satisfiability of the logic $KD45 \times K$ is also NEXPTIME-complete so the satisfiability problem in our setting may be NEXPTIME-hard.

Table 1. A world / argument model

Part of the sentence (syntax)	context world/argument (semantics)
i owns a and	(w, a)
i believes that	all (t, a) such that wR_it, in our case: only (u, a)
there exists an argument x attacked by a...	there exists (u, x) such that $a \rightsquigarrow x$, here: (u, b)
agent j believes that ...	all (t, b) such that uR_jt, in our case: only (v, b)

6 Summary

In this paper, we provide three languages to deal with argumentation and beliefs. The first one $(\mathcal{L}_1)$ enables us to speak about beliefs about awareness of arguments. The second one $(\mathcal{L}_2)$ is a conservative extension of $\mathcal{L}_1$ and enables to speak about beliefs about the structure of the argumentation graph. The third logic $(\mathcal{L}_3)$ enables to speak about beliefs about a specific argument. The third logic has many promising features, but is not a conservative extension of $\mathcal{L}_2$. A part of our future work will be to investigate whether it is possible to slightly change $\mathcal{L}_3$ in order to make it a conservative extension of $\mathcal{L}_2$.

This paper presents a landscape of incremental logics, in the sense that every logic is more expressive than the previous one. As expected, the complexity of consistency checking of a formula increases. For $\mathcal{L}_1$ it is PSPACE-complete, for $\mathcal{L}_2$, it is EXPTIME-complete and for $\mathcal{L}_3$ we conjecture it to be NEXPTIME-hard. As this complexity is high, a part of our future work will be to study their syntactic fragments.

The paper presents the first attempt to formalise agents' beliefs in a multi-agent argumentation setting. We are inspired by the work of Grossi [7]. That paper shows that an argumentation framework can be seen as a Kripke structure. In our paper, we "import" those ideas on argumentation level (inside of each possible world), and develop a framework for reasoning about agents' beliefs. The solution presented in this paper is the first attempt to model awareness about arguments. An urgent extension of our work is to make a detailed comparison with the logic of awareness about propositional facts [15].

Growing interest in game theoretic investigations of argument-based dialogues [11,12,14] shows that a logical framework is needed to represent knowledge and beliefs of agents in such a setting. We believe that our work is the first step in that direction.

References

1. Areces, C., de Rijke, M.: From description to hybrid logics, and back. Advances in Modal Logic 3, 17–36 (2001)
2. Bennett, B., Cohn, A.G., Wolter, F., Zakharyaschev, M.: Multi-dimensional modal logic as a framework for spatio-temporal reasoning. Applied Intelligence 17(3), 239–251 (2002)
3. Bolander, T., Braüner, T.: Tableau-based decision procedures for hybrid logic. Journal of Logic and Computation 16(6), 737–763 (2006)
4. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence Journal 77, 321–357 (1995)

5. Fagin, R., Moses, Y., Halpern, J., Vardi, M.: Reasoning about knowledge. The MIT Press (2003)
6. Gabbay, D.M.: Many-dimensional modal logics: theory and applications, vol. 148. North-Holland (2003)
7. Grossi, D.: On the logic of argumentation theory. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), pp. 409–416. IFAAMAS (2010)
8. Halpern, J.Y., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artificial intelligence 54(3), 319–379 (1992)
9. Halpern, J.Y., Vardi, M.Y.: The complexity of reasoning about knowledge and time. i. lower bounds. Journal of Computer and System Sciences 38(1), 195–237 (1989)
10. Pollock, J.: How to reason defeasibly. Artificial Intelligence Journal 57, 1–42 (1992)
11. Rahwan, I., Larson, K.: Argumentation and game theory, pp. 321–339. Springer (2009)
12. Riveret, R., Prakken, H., Rotolo, A., Sartor, G.: Heuristics in argumentation: A game theory investigation. In: COMMA, pp. 324–335 (2008)
13. Simari, G.R., Loui, R.P.: A mathematical treatment of defeasible reasoning and its implementation. Artificial Intelligence Journal 53, 125–157 (1992)
14. Thimm, M., Garcia, A.J.: Classification and strategical issues of argumentation games on structured argumentation frameworks. In: Proceedings of the Ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems 2010, AAMAS 2010 (2010)
15. van Ditmarsch, H., French, T.: Becoming aware of propositional variables. In: Logic and Its Applications, pp. 204–218 (2011)
16. Vreeswijk, G.: Abstract argumentation systems. Artificial Intelligence Journal 90, 225–279 (1997)

A Unifying Perspective on Knowledge Updates

Martin Slota and João Leite

CENTRIA & Departamento de Informática
Universidade Nova de Lisboa
Quinta da Torre
2829-516 Caparica, Portugal

Abstract. We introduce an abstract update framework based on viewing a knowledge base as the *set of sets of models* of its elements and performing updates by introducing additional interpretations – *exceptions* – to the sets of models of elements of the original knowledge base. In [36], an instantiation of this framework for performing rule updates has been shown to semantically characterise one of the syntax-based *rule update semantics*. In this paper we show that the framework can also capture a wide range of both model- and formula-based belief update operators which constitute the formal underpinning of existing approaches to *ontology updates*. Exception-driven operators thus form a unifying perspective on both ontology and rule updates, opening new possibilities for addressing *updates of hybrid knowledge bases* consisting of both an ontology and a rule component.

1 Introduction

In this paper we propose a novel generic method for specifying update operators. By viewing a knowledge base as the *set of sets of models of its elements*, and seeing updates as *adding new interpretations* to those sets, we are able to capture a range of model- and formula-based belief update operators. When coupled with the results of [36] in which an instantiation of this framework was shown to characterise a syntax-based rule update semantics, our findings imply that exception-driven operators are the first approach that embraces these two seemingly irreconcilable approaches to updates.

Throughout the last decade, standardisation efforts gave rise to widely accepted knowledge representation languages such as the Web Ontology Language (OWL)[1] and Rule Interchange Format (RIF),[2] based on Description Logics [4] and Logic Programming [16], respectively. This has fostered a large number of ontologies and rule bases with different levels of complexity and scale. Whereas ontologies provide the logical underpinning of intelligent access and information integration, rules are widely used to represent business policies, regulations and declarative guidelines about information.

Since both ontologies and rules offer important features for knowledge representation, considerable effort has been invested in identifying a unified hybrid knowledge framework where expressivity of both formalisms could be seamlessly combined. This task turned out to be very challenging because of the inherent semantic differences between the two knowledge representation paradigms.

[1] `http://www.w3.org/TR/owl-overview/`
[2] `http://www.w3.org/2005/rules/wiki/RIF_Working_Group`

Over the years, work on hybrid knowledge bases has matured significantly and fundamental semantic as well as computational problems were addressed successfully (see [20] for an overview). The recent formalisms, based on an embedding to a unifying non-monotonic formalism, such as the Autoepistemic Logic [6] or the Logic of Minimal Knowledge and Negation as Failure (MKNF) [27], provide a tight and semantically neat integration of ontologies and rules, allowing predicates to be defined concurrently in the ontology as well as by rules. Nevertheless, they only deal with *static knowledge*.

One of the main challenges for knowledge engineering and information management is to efficiently and plausibly deal with the incorporation of new, possibly conflicting knowledge and beliefs. In other words, support for *knowledge dynamics* is essential. This topic has been extensively addressed in the context of both Description Logics and Logic Programs, when taken separately.

Ontology Updates. The area of research called *ontology change* encompasses a number of strongly related though distinguishable subareas, such as ontology matching, ontology integration and merging, or ontology translation (a survey can be found in [14]). The purest type of change, concerned with modifications to a single ontology, is generally referred to as *ontology evolution*. Approaches to ontology evolution with a firm semantic underpinning, thus amenable to a formal analysis of their behaviour and properties, are based on research in the area of *belief change*, initiated by the seminal work of Alchourrón, Gärdenfors and Makinson (AGM) [1] who proposed a set of desirable properties of change operators on monotonic logics, now called *AGM postulates*.

Subsequently, *revision* and *update* were distinguished as two very related but ultimately different belief change operations [39,21,28]. While revision deals with incorporating new information about a *static world* into a knowledge base, update takes place when a knowledge base needs to be brought up to date when the *modelled world changes*. While AGM postulates were deemed appropriate for describing revision, a different set of postulates was suggested for belief update [21,28].

Update operators based on these postulates, usually referred to as *model-based*, were later used to partially address ontology updates [25,10], namely to update the part of the ontology with assertions about individuals (the *ABox*). On the other hand, model-based operators are considered inappropriate for updating ontological axioms that define the terminology (the *TBox*) [7,34]. Their antipole, *formula-based operators*, which manipulate the knowledge base at a syntactic level and are strongly related to *base revision operators*, were adopted for performing TBox updates instead [7].

Rule Updates. When updates were tackled in the context of Logic Programming, it was only natural to consider adapting the belief update postulates and operators to deal with them. However, this led to counterintuitive results because the model-based approach fails to capture the essential relationships between literals encoded in rules [23], and the formula-based approach is too crude as it does not allow rules to be reactivated when reasons for their suppression disappear [40]. Although state-of-the-art approaches to rule updates are guided by the same basic intuitions and aspirations as belief updates, they build upon fundamentally different principles and methods.

Many of them are based on the *causal rejection principle* [23,3,12,2] which states that a rule is *rejected* only if it is directly contradicted by a more recent rule. This essentially means that inertia and minimal change, applied at the level of *literals* in model-based belief update operators, is instead applied to *rules* and the truth values of literals follow from the set of unrejected rules. Causal rejection semantics are useful in a number of practical scenarios and their behaviour is intuitively predictable. Alternative approaches to rule updates employ syntactic transformations and other methods, such as abduction [31], prioritisation and preferences [40,11], or dependencies on default assumptions [32,22]. The main feature of all these approaches is that they need to refer to the syntactic structure of a logic program: the individual rules and, in most cases, also the literals in heads and bodies of these rules. These properties render them seemingly irreconcilable with belief updates since ontology axioms and formulae in Classical Logic simply have no heads and bodies.

Towards Updates of Hybrid Knowledge Bases. The question that arises, then, is: How can we combine methods used for updating ABoxes, TBoxes and rules in a single framework that allows us to *update hybrid knowledge bases*?

In [34,37] we provided partial solutions to this problem but the inherent differences between the distinct approaches to updates have prevented us from suggesting a universal hybrid update semantics. Subsequently, in [33,35,36] we looked for a suitable semantic foundation of rule updates which would be independent of rule syntax and, at the same time, would retain the fundamental properties of existing rule update semantics. This led to the study of *exception-driven rule update operators* and a definition of particular operators that semantically capture the justified update semantics for rule updates [23] and enjoy a number of syntactic as well as semantic properties.

In this paper we go one step beyond and show that exception-driven operators also capture a range of belief update operators, both model- and formula-based. In other words, they form a common basis for both ontology and rule updates and create room for their cross-fertilisation, ripening and further development.

Our main contributions in this paper are as follows:

- We define abstract exception-driven operators for any knowledge representation formalism with a model-theoretic semantics.
- We show that they capture a wide range of belief update operators.
- We discuss the relationship between belief set and belief base revision operators, on the one hand, and exception-driven operators, on the other.

This work has the following structure: In Sect. 2 we introduce update operators for first-order knowledge bases which form the basis for ontology updates. Sections 3 and 4 introduce abstract exception-driven operators and show how they are able to characterise belief updates. We discuss the relationship between our update framework and revision operators and point at interesting future research directions in Sect. 5.

2 Preliminaries

In this section we introduce model- and formula-based update operators for first-order knowledge bases which underlie the formal approaches to ontology updates [25,10,7,24].

One of the main issues with ontology updates is the *expressibility* of the result of an update which arises due to the fact that Description Logics are *fragments* of first-order logic, so the result of an update operator may not be expressible in the DL used to encode the original ontology and its update [5]. Nevertheless, in this paper we abstract away from this problem, noting that Description Logics for which expressibility is guaranteed have been identified [25,10], approximation techniques for the updated ontology also constitute a viable solution to this problem [10], and the recent work on belief revision within Horn and other fragments of classical logic may show this problem from a new viewpoint (see e.g. [8] and references therein).

Throughout the remainder of this paper we thus assume to be using a function-free first-order language consisting of disjoint non-empty sets of constant and predicate symbols $\boldsymbol{C}$ and $\boldsymbol{P}$. First-order formulae are defined in the standard way and by a *(first-order) knowledge base* we mean a set of first-order sentences.

From a semantic viewpoint we adopt first-order interpretations under the *standard names assumption* in order to simplify comparison between first-order interpretations, problematic when the interpretation of constants may vary [39,10]. More formally, we assume that the set of constant symbols $\boldsymbol{C}$ is infinite and all first-order interpretations are over the universe $\boldsymbol{C}$ where every constant is interpreted by itself. In addition, we assume that the equality predicate $\approx$ is allowed to be interpreted by any congruence relation on $\boldsymbol{C}$ that allows for replacement of equals by equals, enabling us to support updates of equality assertions. The set of all interpretations satisfying these conditions is denoted by $\boldsymbol{I}$. Note that due to Theorems 5.9.4 and 9.3.9 in [13], the semantics we adopt preserves the standard first-order consequences of all finite knowledge bases.

Furthermore, every interpretation $I \in \boldsymbol{I}$ directly corresponds to the set of ground atoms that it entails; in the following we use these two notions interchangeably. We denote the set of models of a knowledge base $\mathcal{B}$ by $[\![\mathcal{B}]\!]$ and say that $\mathcal{B}$ is *consistent* if $\mathcal{B}$ has a model. Given two knowledge bases $\mathcal{B}$, $\mathcal{C}$, we say that $\mathcal{B}$ *entails* $\mathcal{C}$, denoted by $\mathcal{B} \models \mathcal{C}$, if $[\![\mathcal{B}]\!] \subseteq [\![\mathcal{C}]\!]$, and that $\mathcal{B}$ *is equivalent to* $\mathcal{C}$, denoted by $\mathcal{B} \equiv \mathcal{C}$, if $[\![\mathcal{B}]\!] = [\![\mathcal{C}]\!]$.

We liberally define an update operator as any function that takes the original knowledge base and its update as inputs, and returns the updated knowledge base.

Definition 1 (Update Operator). *A* (first-order) update operator *is a binary function over the set of all knowledge bases. Any update operator $\diamond$ is inductively generalised to finite sequences of knowledge bases $\langle \mathcal{B}_i \rangle_{i<n}$ as follows:*

$$\diamond\langle \mathcal{B}_0 \rangle = \mathcal{B}_0 \ , \qquad\qquad \diamond\langle \mathcal{B}_i \rangle_{i<n+1} = (\diamond\langle \mathcal{B}_i \rangle_{i<n}) \diamond \mathcal{B}_n \ .$$

In the following we consider two complementary ways of further specifying an update operator. While the first one puts constraints on the models of knowledge bases produced by it, the second directly defines the resulting knowledge base by performing modifications at the syntactic level.

Model-Based Update Operators. The basic idea underlying model-based update operators is that models of the original knowledge base are viewed as *alternative states* of the modelled world, only one of which is the true one. Given this perspective, it is natural to perform an update with $\mathcal{U}$ by updating each of the alternatives independently of the others, making it consistent with $\mathcal{U}$, and thus obtaining a new set of interpretations – the models of the updated knowledge base. Formally this is captured by the equation

$$[\![\mathcal{B} \diamond \mathcal{U}]\!] = \bigcup_{I \in [\![\mathcal{B}]\!]} \mathsf{incorporate}([\![\mathcal{U}]\!], I) \ , \tag{1}$$

where $\mathsf{incorporate}(\mathcal{M}, I)$ returns the members of $\mathcal{M}$ closer to I so that the original information in I is preserved as much as possible. A natural way of defining this set is by assigning a preorder $\leq^I$ over $\boldsymbol{I}$ to each interpretation I and taking the minima of $\leq^I$ within $\mathcal{M}$, i.e. $\mathsf{incorporate}(\mathcal{M}, I) = \min(\mathcal{M}, \leq^I)$. In the following we first formally establish the concept of an *order assignment*; thereafter we define when an update operator is *characterised by an order assignment.*

Given a set $\mathcal{S}$, a *preorder over* $\mathcal{S}$ is a reflexive and transitive binary relation over $\mathcal{S}$; a *strict preorder over* $\mathcal{S}$ is an irreflexive and transitive binary relation over $\mathcal{S}$. Given a preorder $\leq$ over $\mathcal{S}$, we denote by $<$ the strict preorder induced by $\leq$, i.e. $s < t$ if and only if $s \leq t$ and not $t \leq s$. For any subset $\mathcal{T}$ of $\mathcal{S}$, the set of minimal elements of $\mathcal{T}$ w.r.t. $\leq$ is denoted by $\min(\mathcal{T}, \leq)$. A *preorder assignment over* $\mathcal{S}$ is any function ω that assigns a preorder $\leq^s_\omega$ over $\mathcal{S}$ to each $s \in \mathcal{S}$. A preorder assignment ω is *faithful* if for all $s, t \in \mathcal{S}$ with $s \neq t$, $s <^s_\omega t$.

Definition 2 (Model-Based Update Operator [21]). *Let $\diamond$ be a first-order update operator and ω a preorder assignment over $\boldsymbol{I}$. We say that $\diamond$ is* characterised by ω *if for all knowledge bases $\mathcal{B}$, $\mathcal{U}$,*

$$[\![\mathcal{B} \diamond \mathcal{U}]\!] = \bigcup_{I \in [\![\mathcal{B}]\!]} \min\left([\![\mathcal{U}]\!], \leq^I_\omega\right) \enspace .$$

An operator $\diamond$ is model-based *if it is characterised by some faithful preorder assignment.*

The model-based operator that underlies the work on ABox updates [25,10] is Winslett's operator which compares interpretations based on the sets of ground atoms that they interpret differently than the original interpretation.

Definition 3 (Winslett's Operator [39]). *The preorder assignment W is defined for all interpretations $I, J, K \in \boldsymbol{I}$ as $J \leq^I_{\mathsf{W}} K$ if and only if $(J \div I) \subseteq (K \div I)$, where $\div$ denotes the set-theoretic symmetric difference.* Winslett's operator $\diamond_{\mathsf{W}}$ *is a fixed update operator that is characterised by W.*

Formula-Based Operators. The traditional formula-based update operators that manipulate a knowledge base syntactically are Set-Of-Theories, Cross-Product and WIDTIO (see [39] and references therein). The central notion for these operators is that of a *possible remainder* which is a maximal set of formulae from the original knowledge base that is consistent with the update. Formally, given knowledge bases $\mathcal{B}$ and $\mathcal{U}$, the set of possible remainders $\mathsf{rem}(\mathcal{B}, \mathcal{U})$ is the set of maximal subsets $\mathcal{B}'$ of $\mathcal{B}$ such that $\mathcal{B}' \cup \mathcal{U}$ is consistent. The distinct formula-based operators differ in how they deal with multiple possible remainders. The Set-Of-Theories operator returns the set of all alternative results, i.e. the set of knowledge bases $\mathcal{B}' \cup \mathcal{U}$ for every $\mathcal{B}' \in \mathsf{rem}(\mathcal{B}, \mathcal{U})$. Assuming that the initial knowledge base is finite, the Cross-Product operator compiles these different remainders into a single formula and returns a single knowledge base that is equivalent to the "disjunction" of knowledge bases returned by the Set-Of-Theories operator.

Definition 4 (Cross-Product Operator). *The formula-based operator $\diamond_{\mathsf{CP}}$ is defined for all finite knowledge bases $\mathcal{B}, \mathcal{U}$ as $\mathcal{B} \diamond_{\mathsf{CP}} \mathcal{U} = \mathcal{U} \cup \{ \psi \}$ where ψ is the formula*

$$\bigvee_{\mathcal{B}' \in \mathsf{rem}(\mathcal{B}, \mathcal{U})} \bigwedge_{\phi \in \mathcal{B}'} \phi \enspace .$$

On the other hand, the operator WIDTIO (When In Doubt, Throw It Out [39]) takes the safe path – it keeps exactly those formulae that belong to the intersection of all remainders and throws away the rest.

Definition 5 (WIDTIO Operator). *The formula-based operator* $\diamond_{\mathsf{WIDTIO}}$ *is defined for all knowledge bases* $\mathcal{B}$, $\mathcal{U}$ *as* $\mathcal{B} \diamond_{\mathsf{WIDTIO}} \mathcal{U} = \mathcal{U} \cup \bigcap \mathsf{rem}(\mathcal{B}, \mathcal{U})$.

Recently, an operator inspired by WIDTIO was defined in [24] to tackle ABox updates. Additionally, in [7] the new formula-based operator *Bold* was suggested for performing TBox updates because of the counterintuitive behaviour of model-based operators when used for this purpose. The Bold operator solves the problem of multiple remainders by using a selection function to choose one and commit to it.

Definition 6 (Bold Operator [7]). *A* remainder selection function *is a function* s *that assigns to every set of remainders* $\mathcal{R}$ *a remainder* $\mathsf{s}(\mathcal{R}) \in \mathcal{R}$.

Given a remainder selection function s, *the formula-based operator* $\diamond^{\mathsf{s}}_{\mathsf{BOLD}}$ *is for all knowledge bases* $\mathcal{B}$, $\mathcal{U}$ *defined as* $\mathcal{B} \diamond^{\mathsf{s}}_{\mathsf{BOLD}} \mathcal{U} = \mathcal{U} \cup \mathsf{s}(\mathsf{rem}(\mathcal{B}, \mathcal{U}))$.

3 Exception-Driven Operators

In order to show how belief- and formula-based operators can be characterised in a unified manner, we define an abstract framework for *exception-driven operators*, usable for any knowledge representation formalism with a monotonic model-theoretic semantics. We also demonstrate how the *justified update semantics* (or *JU-semantics*) for rule updates [23] was characterised semantically in [36] using exception-driven operators.

Abstract Exception-Driven Operators. Throughout this subsection we assume to be using some knowledge representation formalism in which a *knowledge base* is a subset of the set of all *knowledge atoms* Ω and $\mathcal{Z}$ denotes the set of all *semantic structures* among which the *models* of knowledge atoms are chosen. The set of models of a knowledge atom α is denoted by $[\![\alpha]\!]$. The *semantic characterisation* of a knowledge base $\mathcal{K}$ is the *set of sets of models* of its knowledge atoms: $\langle\!\langle \mathcal{K} \rangle\!\rangle = \{ [\![\alpha]\!] \mid \alpha \in \mathcal{K} \}$. The models of $\mathcal{K}$ are the models of all its elements, i.e. $[\![\mathcal{K}]\!] = \bigcap \langle\!\langle \mathcal{K} \rangle\!\rangle$.

An exception-driven operator views a knowledge $\mathcal{K}$ through its semantic characterisation $\langle\!\langle \mathcal{K} \rangle\!\rangle$ and *introduces exceptions* to its knowledge atoms by adding new semantic structures to their original sets of models. The formalisation of this idea is straightforward: an exception-driven update operator is characterised by an *exception function* that, given the set of models of a knowledge atom α and the semantic characterisations of the original and updating knowledge base, returns the set of semantic structures that are to be introduced as exceptions to α.

Definition 7 (Exception Function). *An* exception function *is any function*

$$\varepsilon : 2^{\mathcal{Z}} \times 2^{2^{\mathcal{Z}}} \times 2^{2^{\mathcal{Z}}} \rightarrow 2^{\mathcal{Z}} \enspace .$$

Given such an exception function and knowledge bases $\mathcal{K}$, $\mathcal{U}$, it naturally follows that the semantic characterisation resulting from updating $\mathcal{K}$ by $\mathcal{U}$ should consist of sets of models of each knowledge atom α from $\mathcal{K}$, each augmented with the respective exceptions, and also the unmodified sets of models of knowledge atoms from $\mathcal{U}$. In other words, we obtain the set of sets of models

$$\{ [\![\alpha]\!] \cup \varepsilon([\![\alpha]\!], \langle\!\langle \mathcal{K} \rangle\!\rangle, \langle\!\langle \mathcal{U} \rangle\!\rangle) \mid \alpha \in \mathcal{K} \} \cup \langle\!\langle \mathcal{U} \rangle\!\rangle \ . \tag{2}$$

Turning to the syntactic side, an *update operator* is binary function over 2^{Ω} that takes the original knowledge base and its update as inputs and returns the updated knowledge base. An *exception-driven update operator* is then formalised as follows:

Definition 8 (Exception-Driven Update Operator). *We say that an update operator $\oplus$ is* exception-driven *if for some exception function ε, $\langle\!\langle \mathcal{K} \oplus \mathcal{U} \rangle\!\rangle$ is equal to* (2) *for all $\mathcal{K}, \mathcal{U} \subseteq \Omega$. In that case we also say that $\oplus$ is ε*-driven.

Before we begin formally comparing model- and formula-based operators with exception-driven ones, we briefly illustrate how the results of [36], where the JU-semantics for rule updates was semantically characterised, fit within our abstract framework for exception-driven operators. Our main intention in doing so is to provide the reader with a broader picture of exception-driven operators; the technical details left out in what follows can be found in [36].

Exception-Driven Rule Updates. We adopt the standard syntax and the stable models semantics of propositional logic programs [16]. In particular, given a set of atoms $\boldsymbol{A}$, a *literal* is an atom $p \in \boldsymbol{A}$ or its default negation $\sim\!p$, a *rule* consists of a pair of sets of literals $(H(\pi), B(\pi))$, usually written $(H(\pi) \leftarrow B(\pi).)$, and a *program* is a set of rules. An *interpretation* is a subset of $\boldsymbol{A}$ that naturally assigns truth values to atoms and a *model* of a rule is an interpretation that satisfies the rule when interpreted as a classical implication. Models of a program $\mathcal{P}$ are the models of all its rules and an interpretation J is a *stable model* of $\mathcal{P}$ if it is a subset-minimal model of its Gelfond-Lifschitz reduct $\mathcal{P}^J$ [16]. The set of stable models of $\mathcal{P}$ is denoted by $[\![\mathcal{P}]\!]_{\mathsf{SM}}$.

The goal of rule update semantics [26,23,3,12,2,31,40,11,22,32] is to generalise the definition of stable models to *pairs* or *sequences of programs* where each component represents an update of the preceding ones. These semantics are usually constrained to *finite sequences of non-disjunctive programs* which we call *dynamic logic programs* (DLPs). Typically, they are defined by referring to the syntactic structure of the programs in a DLP. As a consequence, analysis of their semantic properties is very daunting and most of them do exhibit undesirable behaviour, e.g. by being sensitive to *tautological updates* which is counterintuitive in the context of updates – a tautology cannot encode a *change* in the modelled world because it is always true. The historically first semantics for DLPs is the JU-semantics [23]. We denote the set of all JU-models of a DLP $\mathcal{D}$ by $[\![\mathcal{D}]\!]_{\mathsf{JU}}$.

The operators introduced in [36] can be seen as an instantiation of the abstract framework introduced above. In this context, the set of knowledge atoms Ω consists of all *rules and programs* and the set of semantic structures $\mathcal{Z}$ of *three-valued interpretations*. A *knowledge base* (or *rule base*) is thus any set of rules and programs and its elements are perceived as atomic pieces of knowledge. Note that a program is a special case of a rule base. The reason why we allow for programs *inside* a rule base is that when a rule is updated, by adding exceptions to its set of models, the resulting set of models is usually not expressible by a rule, only by a program. Note also that the notion of a *stable model* can be naturally generalised to rule bases by introducing models of a rule base as the models of all its elements and defining the Gelfond-Lifschitz reduct of a rule base $\mathcal{R}$ as $\mathcal{R}^J = \{ \Pi^J \mid \Pi \in \mathcal{R} \}$.

In [36], the semantics assigned to each rule or program in a rule base is given by a refinement of *SE-models* [38], dubbed *RE-models*, which can distinguish additional classes of rules, indispensable in the context of updates. An exception function, here denoted by $\varepsilon_{\mathsf{JU}}$, is then defined. Details about RE-models and $\varepsilon_{\mathsf{JU}}$ can be found in [36].

The main property of $\varepsilon_{\mathsf{JU}}$ is that stable models of the rule base produced by an $\varepsilon_{\mathsf{JU}}$-driven operator, when applied to a DLP $\mathcal{D}$, coincide with its JU-models. This holds whenever $\mathcal{D}$ does not contain *local cycles*, i.e. rules π with both $\{ p, {\sim}p \} \cap H(\pi) \neq \emptyset$ and $\{ p, {\sim}p \} \cap B(\pi) \neq \emptyset$ for some $p \in \boldsymbol{A}$.

Theorem 9 ([36]). *Let $\mathcal{D}$ be a DLP without local cycles, J an interpretation and $\oplus$ an $\varepsilon_{\mathsf{JU}}$-driven rule update operator. Then $[\![\bigoplus \mathcal{D}]\!]_{\mathsf{SM}} = [\![\mathcal{D}]\!]_{\mathsf{JU}}$.*

This means that up to the marginal case of local cycles, $\varepsilon_{\mathsf{JU}}$ can be seen as a semantic characterisation of the JU-semantics: it leads to stable models that coincide with JU-models. In case the DLP contains local cycles, *less* stable models than JU-models are found [36]. Local cycles correspond to two different kinds of rules: tautological rules and rules with the negation of their head in the body. The different behaviour in the presence of tautological rules is a strict improvement over JU-models, as it introduces immunity to tautological updates. The other differences are a consequence of treating constraints such as $(p \leftarrow {\sim}p.)$ and $(\leftarrow {\sim}p.)$ uniformly while the JU-semantics treats them differently under certain circumstances.

This tight relationship allowed us to study the semantic properties of JU-models under a range of different notions of program equivalence and entailment, and to shed new light on the problem of *state condensing* since $\varepsilon_{\mathsf{JU}}$-driven operators compress any DLP into a single equivalent rule base.

Even more importantly, these results, along with the developments in this paper, show that exception-driven operators form a common semantic basis for both ontology and rule updates, and so create room for addressing updates of hybrid knowledge bases.

4 Belief Updates Using Exception-Driven Operators

Concrete exception-driven operators for first-order knowledge bases are obtained from the abstract framework developed in Sect. 3 by identifying the set of knowledge atoms Ω with the set of first-order sentences and the set of semantic structures $\mathcal{Z}$ with first-order interpretations under the standard names assumption, as introduced in Sect. 2.

In [21] it was shown that *propositional* model-based update operators are exactly those that satisfy a collection of eight *update postulates*. These postulates express basic desirable properties of update operators and most of them can be directly generalised to the first-order case. Here we use the following three basic properties of first-order update operators and prove results about the class of all operators satisfying them. The properties are formulated for an update operator $\diamond$ and quantified over all knowledge bases $\mathcal{B}, \mathcal{C}, \mathcal{U}, \mathcal{V}$.[3]

[3] Their numbers are as in [21,19].

(U1) $\mathcal{B} \diamond \mathcal{U} \models \mathcal{U}$.

(U2.1) $\mathcal{B} \cup \mathcal{U} \models \mathcal{B} \diamond \mathcal{U}$.

(U4) If $\mathcal{B} \equiv \mathcal{C}$ and $\mathcal{U} \equiv \mathcal{V}$, then $\mathcal{B} \diamond \mathcal{U} \equiv \mathcal{C} \diamond \mathcal{V}$.

The intuitive reading of (U1) is that information from the update must be retained in the updated knowledge base, also known as the *principle of primacy of new information* [9]; (U2.1) expresses that models of $\mathcal{B}$ that are also models of $\mathcal{U}$, and thus need not be updated, are kept as models of the updated knowledge base; (U4) specifies that the operator must be *syntax-independent*, i.e. it must provide equivalent results given equivalent inputs. All model-based update operators, including Winslett's, satisfy these principles:

Proposition 10 (Properties of Model-Based Updates). *Every model-based update operator satisfies (U1), (U2.1) and (U4).*

Furthermore, any operator satisfying these three principles can be faithfully modelled by an exception-driven operator. Formally:

Theorem 11 (Model-Based Updates Using Exception-Driven Operators). *If $\diamond$ is an update operator that satisfies (U1), (U2.1) and (U4), then there exists an exception function ε such that for every ε-driven update operator $\oplus$ and all finite sequences of knowledge bases $\mathcal{D}$, $[\![\diamond\mathcal{D}]\!] = [\![\bigoplus\mathcal{D}]\!]$.*

Similar results can be achieved for formula-based update operators. First we introduce the following principles, counterparts of the respective belief update postulates, which are satisfied by many formula-based operators. We denote by $\langle\!\langle \mathcal{B} \rangle\!\rangle^{\boldsymbol{I}}$ the set $\langle\!\langle \mathcal{B} \rangle\!\rangle \cup \{ \boldsymbol{I} \}$ for any knowledge base $\mathcal{B}$. The principles are as follows:

(F1) $\langle\!\langle \mathcal{B} \diamond \mathcal{U} \rangle\!\rangle \supseteq \langle\!\langle \mathcal{U} \rangle\!\rangle$.

(F2.1) $\langle\!\langle \mathcal{B} \cup \mathcal{U} \rangle\!\rangle \supseteq \langle\!\langle \mathcal{B} \diamond \mathcal{U} \rangle\!\rangle$.

(F4) If $\langle\!\langle \mathcal{B} \rangle\!\rangle^{\boldsymbol{I}} = \langle\!\langle \mathcal{C} \rangle\!\rangle^{\boldsymbol{I}}$ and $\langle\!\langle \mathcal{U} \rangle\!\rangle^{\boldsymbol{I}} = \langle\!\langle \mathcal{V} \rangle\!\rangle^{\boldsymbol{I}}$, then $\langle\!\langle \mathcal{B} \diamond \mathcal{U} \rangle\!\rangle^{\boldsymbol{I}} = \langle\!\langle \mathcal{C} \diamond \mathcal{V} \rangle\!\rangle^{\boldsymbol{I}}$.

We can see that (F1) and (F2.1) are *stronger* versions of (U1), and (U2.1), respectively. While (F1) requires that the sets of models of formulae in $\mathcal{U}$ be retained in the semantic characterisation of $\mathcal{B} \diamond \mathcal{U}$, (F2.1) states that every formula in $\mathcal{B} \diamond \mathcal{U}$ be equivalent to some formula in $\mathcal{B} \cup \mathcal{U}$. Intuitively, this means that $\mathcal{B} \diamond \mathcal{U}$ is obtained from $\mathcal{B} \cup \mathcal{U}$ by deleting some of its elements, modulo equivalence. Finally, (F4) is a reformulation of (U4) that is satisfied by formula-based operators – it can be seen as syntax-independence w.r.t. the set of sets of models of a knowledge base, modulo the presence of tautologies, instead of the overall set of models as in (U4). In some ways it is *weaker* than (U4) as its antecedent is much stronger.

The WIDTIO operator satisfies all of these principles, and so does the Bold operator if it is based on a remainder selection function that selects remainders with the same semantic characterisation when given sets of remainders with the same sets of semantic characterisations. More formally:

Definition 12 (Regular Bold Operator). *Let $\mathcal{R}$ be a set of remainders. We denote the set $\{ \langle\!\langle \mathcal{B}' \rangle\!\rangle^{\boldsymbol{I}} \mid \mathcal{B}' \in \mathcal{R} \}$ by $(\!(\mathcal{R})\!)^{\boldsymbol{I}}$.*

We say that the Bold operator $\diamond^{\mathsf{s}}_{\mathsf{BOLD}}$ is regular *if for all sets of remainders $\mathcal{R}_1$, $\mathcal{R}_2$ such that $(\!(\mathcal{R}_1)\!)^{\boldsymbol{I}} = (\!(\mathcal{R}_2)\!)^{\boldsymbol{I}}$ it holds that $\langle\!\langle \mathsf{s}(\mathcal{R}_1) \rangle\!\rangle^{\boldsymbol{I}} = \langle\!\langle \mathsf{s}(\mathcal{R}_2) \rangle\!\rangle^{\boldsymbol{I}}$.*

The regularity condition guarantees a certain degree of independence of syntax, e.g. given the sets of remainders $\mathcal{R}_1 = \{ \{ p \}, \{ q \} \}$ and $\mathcal{R}_2 = \{ \{ p \wedge p \}, \{ q \vee q \} \}$, a regular Bold operator either selects $\{ p \}$ from $\mathcal{R}_1$ and $\{ p \wedge p \}$ from $\mathcal{R}_2$, or it selects $\{ q \}$ from $\mathcal{R}_1$ and $\{ q \vee q \}$ from $\mathcal{R}_2$. A non-regular one might select, say, $\{ p \}$ from $\mathcal{R}_1$ and $\{ q \vee q \}$ from $\mathcal{R}_2$. Thus the regularity condition ensures that the operator is independent of the syntax of individual formulae in the knowledge base.

The Cross-Product operator satisfies (F1), (U2.1) and (F4), but not (F2.1).

Proposition 13 (Properties of Formula-Based Updates). *The WIDTIO and regular Bold operators satisfy (F1), (F2.1) and (F4). The Cross-Product operator satisfies (F1), (U2.1) and (F4) but does not satisfy (F2.1).*

The following result establishes that formula-based operators such as WIDTIO and regular Bold can be fully captured by exception-driven operators. In addition, operators such as Cross-Product can be captured for the case of a single update.

Theorem 14 (Formula-Based Updates Using Exception-Driven Operators). *If $\diamond$ is an update operator that satisfies (F1), (F2.1) and (F4), then there exists an exception function ε such that for every ε-driven update operator $\oplus$ and all finite sequences of knowledge bases $\mathcal{D}$, $[\![\diamond\mathcal{D}]\!] = [\![\bigoplus\mathcal{D}]\!]$.*

If $\diamond$ is an update operator that satisfies (F1), (U2.1) and (F4), then there exists an exception function ε such that for every ε-driven update operator $\oplus$ and all knowledge bases $\mathcal{B}$, $\mathcal{U}$, $[\![\mathcal{B} \diamond \mathcal{U}]\!] = [\![\mathcal{B} \oplus \mathcal{U}]\!]$.

5 Discussion

We have introduced exception-driven operators for first-order knowledge bases and shown that they can fully capture update operators that form the basis of ontology updates, such as the model-based Winslett's operator, or the formula-based WIDTIO and Bold operators [25,10,7,24]. The Cross-Product operator can be captured when a single update is performed. Furthermore, the same can be said about the Set-Of-Theories operator since for a single update it is equivalent to the Cross-Product operator [39], with alternative knowledge bases interpreted disjunctively. However, neither of these two operators offers a viable alternative for updating ontologies. Cross-Product requires that disjunctions of ontology axioms be performed, which is typically not supported in DLs, and Set-Of-Theories produces a disjunctive ontology which is impractical and deviates from mainstream DL research.

An interesting point regarding the results of Sect. 4 is that the principles (U1), (U2.1) and (U4) are not specific to update operators, they are also satisfied by *AGM revision* operators. These operators are developed for the case of revising a *belief set* which is a set of formulae closed w.r.t. a logical consequence operator Cn. A revision operator $\star$ takes an original belief set $\mathcal{T}$ and a formula μ representing its revision and produces the revised belief set $\mathcal{T} \star \mu$. The typical properties satisfied by AGM revision operators include *success*, *inclusion* and *extensionality* [18], formalised, respectively, as

$$\mu \in \mathcal{T} \star \mu \ , \qquad \mathcal{T} \star \mu \subseteq \mathsf{Cn}(\mathcal{T} \cup \{ \mu \}) \ , \qquad \text{If } \mu \equiv \nu \text{, then } \mathcal{T} \star \mu = \mathcal{T} \star \nu .$$

These three properties directly imply that (U1), (U2.1) and (U4) are satisfied by AGM revision operators if the initial knowledge base is a belief set and each of its updates a single formula. This essentially means that Theorem 11 directly applies to AGM revision operators as well. Note that the operator adopted for ABox updates in [24], inspired by WIDTIO, performs a deductive closure of the ABox before updating it, so it corresponds to the standard *full meet AGM revision operator*.

Similarly, principles (F1), (F2.1) and (F4) are closely related with the properties of *base revision operators* [15,18], of which direct instances are the WIDTIO and Bold operators. In particular, two types of base revision are identified in [18], the *internal* and *external base revision*. Both of them satisfy base revision counterparts of *success* and *inclusion* and, in addition, internal revision operators satisfy a property called *uniformity*. These three principles together entail that internal revision operators satisfy (F1), (F2.1) and one half of (F4); the other half can be achieved by putting additional constraints on the two-place selection function that generates the revision operator, similar to the *regularity* condition we imposed on the Bold operator above. Such regular internal revision operators are thus directly subject to Theorem 14. The same however does not hold for regular external revision operators as they need not satisfy *uniformity*. Note also that the WIDTIO and Bold operators coincide with *internal full meet base revision* and *internal maxichoice base revision* operators, respectively.

To sum up, in this paper we introduced the abstract framework for *exception-driven operators* which view a knowledge base or program as the *set of sets of models* of its elements, and perform updates by adding new interpretations – *exceptions* – to the sets of models of elements in the original knowledge base or program. The most important feature of this approach is that it provides a common basis for a wide range of model- and formula-based belief update operators as well as for the JU-semantics, a traditional syntax-based approach to rule updates. In other words, exception functions and exception-driven operators offer a uniform framework that bridges two very distinct approaches to updates, previously considered irreconcilable.

Along with this, new possibilities for addressing updates of hybrid knowledge bases arise. The different methods used for dealing with ABox, TBox and rule updates can be viewed uniformly by looking at their associated exception functions. When coupled with a counterpart of SE- or RE-models in the context of hybrid knowledge bases, this can lead to universal hybrid update semantics which in turn can further improve our understanding of the relation between the distinct update paradigms.

Our discussion of the expressivity of exception-driven operators w.r.t. *revision* operators, on both belief sets and belief bases, can be used to tackle and unify approaches to *ontology revision* [29,17,30]. This seems relevant even in the context of ontology *updates* since it has been suggested in the literature that the strict distinction between revision and update is not suitable in the context of ontologies [7].

Furthermore, exception-driven characterisations of additional rule update semantics need to be investigated. This poses a number of challenges due to the need to detect non-tautological irrelevant updates [2,32]. Insights gained by obtaining exception-driven characterisations of various rule update semantics may also shed light on the problem of *updating disjunctive programs* which has received very little attention up until now.

Acknowledgements. We would like to thank the reviewers for their valuable comments. The authors were partially supported by the FCT funded project ERRO – Efficient Reasoning with Rules and Ontologies (PTDC/EIA-CCO/121823/2010).

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. J. Symb. Log. 50(2), 510–530 (1985)
2. Alferes, J.J., Banti, F., Brogi, A., Leite, J.A.: The refined extension principle for semantics of dynamic logic programming. Studia Logica 79(1), 7–32 (2005)
3. Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C.: Dynamic updates of non-monotonic knowledge bases. J. Log. Program. 45(1-3), 43–70 (2000)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications, 2nd edn. Cambridge Univ. Press (2007)
5. Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: Veloso, M.M., Kambhampati, S. (eds.) Procs. AAAI 2005, pp. 572–577. AAAI/MIT Press (2005)
6. de Bruijn, J., Eiter, T., Polleres, A., Tompits, H.: Embedding nonground logic programs into autoepistemic logic for knowledge-base combination. ACM Trans. Comput. 12(3), 20 (2011)
7. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of *DL-Lite* Knowledge Bases. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 112–128. Springer, Heidelberg (2010)
8. Creignou, N., Papini, O., Pichler, R., Woltran, S.: Belief revision within fragments of propositional logic. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Procs. KR 2012, pp. 126–136. AAAI Press (2012)
9. Dalal, M.: Investigations into a theory of knowledge base revision. In: Procs. AAAI 1988, pp. 475–479. AAAI/MIT Press (1988)
10. De Giacomo, G., Lenzerini, M., Poggi, A., Rosati, R.: On instance-level update and erasure in description logic ontologies. J. Log. Comput. 19(5), 745–770 (2009)
11. Delgrande, J.P., Schaub, T., Tompits, H.: A Preference-Based Framework for Updating Logic Programs. In: Baral, C., Brewka, G., Schlipf, J. (eds.) LPNMR 2007. LNCS (LNAI), vol. 4483, pp. 71–83. Springer, Heidelberg (2007)
12. Eiter, T., Fink, M., Sabbatini, G., Tompits, H.: On properties of update sequences based on causal rejection. TPLP 2(6), 721–777 (2002)
13. Fitting, M.: First-Order Logic and Automated Theorem Proving, 2nd edn. Graduate texts in computer science. Springer, Berlin (1996)
14. Flouris, G., Makanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: classification and survey. Knowledge Eng. Review 23(2), 117–152 (2008)
15. Gärdenfors, P.: Belief Revision: An Introduction. In: Belief Revision, pp. 1–28. Cambridge Univ. Press (1992)
16. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Kowalski, R.A., Bowen, K.A. (eds.) Procs. ICLP/SLP 1988, pp. 1070–1080. MIT Press (1988)
17. Halaschek-Wiener, C., Katz, Y.: Belief base revision for expressive description Logics. In: Grau, B.C., Hitzler, P., Shankey, C., Wallace, E. (eds.) Procs. OWLED 2006. CEUR Workshop Proceedings, vol. 216 (2006), CEUR-WS.org
18. Hansson, S.O.: Reversing the Levi identity. J. Philosophical Logic 22(6), 637–669 (1993)

19. Herzig, A., Rifi, O.: Propositional belief base update and minimal change. Artif. Intell. 115(1), 107–138 (1999)
20. Hitzler, P., Parsia, B.: Ontologies and rules. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, 2nd edn. International Handbooks on Information Systems, pp. 111–132. Springer, Berlin (2009)
21. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: Allen, J.F., Fikes, R., Sandewall, E. (eds.) Procs. KR 1991, April 22-25, pp. 387–394. Morgan Kaufmann Publishers, Cambridge (1991)
22. Krümpelmann, P., Kern-Isberner, G.: On belief dynamics of dependency relations for extended logic programs. In: Procs. NMR 2010, Toronto, Canada (2010)
23. Leite, J., Moniz Pereira, L.: Generalizing Updates: From Models to Programs. In: Dix, J., Moniz Pereira, L., Przymusinski, T.C. (eds.) LPKR 1997. LNCS (LNAI), vol. 1471, pp. 224–246. Springer, Heidelberg (1998)
24. Lenzerini, M., Savo, D.F.: On the evolution of the instance level of DL-Lite knowledge bases. In: Rosati, R., Rudolph, S., Zakharyaschev, M. (eds.) Procs. DL 2011. CEUR Workshop Proceedings, vol. 745 (2011), CEUR-WS.org
25. Liu, H., Lutz, C., Miličić, M., Wolter, F.: Updating description logic ABoxes. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Procs. KR 2006, pp. 46–56. AAAI Press (2006)
26. Marek, V.W., Truszczynski, M.: Revision programming. Theor. Comput. Sci. 190(2), 241–277 (1998)
27. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5), 93–154 (2010)
28. Peppas, P., Nayak, A.C., Pagnucco, M., Foo, N.Y., Kwok, R.B.H., Prokopenko, M.: Revision vs. update: Taking a closer look. In: Wahlster, W. (ed.) Procs. ECAI 1996, pp. 95–99. John Wiley and Sons, Chichester (1996)
29. Qi, G., Yang, F.: A Survey of Revision Approaches in Description Logics. In: Calvanese, D., Lausen, G. (eds.) RR 2008. LNCS, vol. 5341, pp. 74–88. Springer, Heidelberg (2008)
30. Ribeiro, M.M., Wassermann, R.: Base revision in description logics - preliminary results. Procs. IWOD 2007, 69–82 (2007)
31. Sakama, C., Inoue, K.: An abductive framework for computing knowledge base updates. TPLP 3(6), 671–713 (2003)
32. Šefránek, J.: Static and dynamic semantics: Preliminary report. In: MICAI, pp. 36–42 (2011)
33. Slota, M., Leite, J.: On semantic update operators for answer-set programs. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) Procs. ECAI 2010. Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 957–962. IOS Press (2010)
34. Slota, M., Leite, J.: Towards Closed World Reasoning in Dynamic Open Worlds. TPLP Special Issue 10(4-6), 547–564 (2010)
35. Slota, M., Leite, J.: Back and Forth between Rules and SE-Models. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 174–186. Springer, Heidelberg (2011)
36. Slota, M., Leite, J.: Robust equivalence models for semantic updates of answer-set programs. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Procs. KR 2012, pp. 158–168. AAAI Press (2012)
37. Slota, M., Leite, J., Swift, T.: Splitting and updating hybrid knowledge bases. TPLP Special Issue 11(4-5), 801–819 (2011)
38. Turner, H.: Strong equivalence made easy: nested expressions and weight constraints. TPLP 3(4-5), 609–622 (2003)
39. Winslett, M.: Updating Logical Databases. Cambridge Univ. Press, New York (1990)
40. Zhang, Y.: Logic program-based updates. ACM Trans. Comput. Log. 7(3), 421–472 (2006)

Verifying Brahms Human-Robot Teamwork Models

Richard Stocker, Louise Dennis, Clare Dixon, and Michael Fisher

Department of Computer Science, University of Liverpool, U.K.
R.S.Stocker@liverpool.ac.uk

Abstract. Collaboration between robots and humans is an increasingly important aspect of industrial and scientific settings. In addition, significant effort is being put into the development of robot helpers for more general use in the workplace, at home, and in health-care environments. However, before such robots can be fully utilised, a comprehensive analysis of their safety is necessary. Formal verification techniques are regularly used to exhaustively assess system behaviour. Our aim is to apply such techniques to Brahms, a human-agent-robot modelling language. We show how to translate from Brahms scenarios, using a formal semantics for Brahms, into the input language of a model checker. We illustrate the approach by defining, translating, and verifying a domestic robot helper example.

1 Introduction

As autonomous devices are increasingly being developed for, and deployed in, both domestic and industrial scenarios, there is an increasing requirement for humans to at least interact with, and often work cooperatively with, such devices. While the autonomous devices in use at present are just simple sensors or embedded hardware, a much wider range of systems are being developed. These consist not only of devices performing solo tasks, such as the automated vacuum cleaners we see already, but are likely to include robots working cooperatively with humans. For example, there will be robot 'helpers' to assist the elderly and incapacitated in their homes [1, 2], there will be manufacturing robots which will help humans to make complex artifacts [3], and there will be robots tasked with ensuring that humans working in dangerous areas remain safe. All these are being developed, many will be with us in the next ten years, and all involve varying degrees of cooperation and teamwork.

The above examples highlight robots deployed in both domestic and safety-critical industrial situations where human safety can be compromised. Thus, it is vital to carry out as much analysis as is possible not only to maximize the safety of the humans involved, but to ascertain whether the humans and robots together 'can', 'should', or 'will' achieve the goals required of the team activity.

There are several *challenges* facing such analysis. One is to to accurately describe human, and indeed robot, behaviour. Even when we have described such behaviours, how can we exhaustively assess the possible interactions between the humans and robots? While some work has been carried out on the safety analysis of low-level human-robot interactions [4], a detailed analysis of the *high-level* behaviours within such systems has not yet been achieved.

In this paper, we tackle the general problem of matching a set of requirements (which could concern safety, capabilities, or interactions) against scenarios involving humans, robots, and agents. Within this, we use important work on high-level modelling of human-agent-robot teamwork that has already been carried out using the Brahms framework [5]. Thus, we assume that the key interactions and behaviours of any human-agent-robot scenario have been captured within a Brahms model. We also assume that a set of informal requirements have been constructed. The work described in this paper essentially describes the *solid* arrows within Fig. 1.

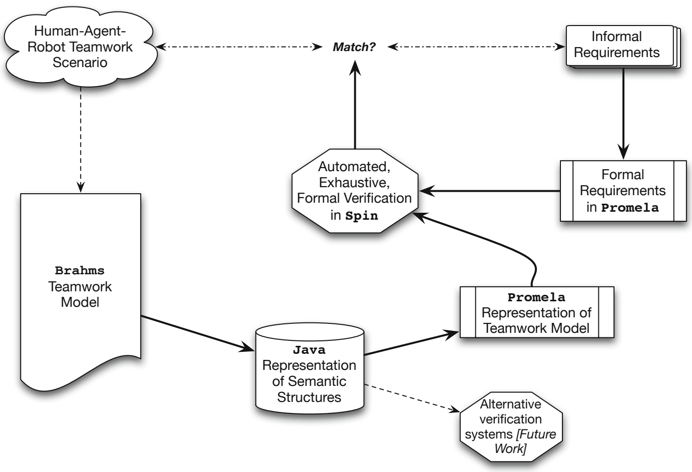

Human-Agent-Robot Teamwork Scenario
Match?
Informal Requirements
Automated, Exhaustive, Formal Verification in Spin
Formal Requirements in Promela
Brahms Teamwork Model
Promela Representation of Teamwork Model
Java Representation of Semantic Structures
Alternative verification systems [Future Work]

Fig. 1. Overview of the processes described in this paper

Thus, given a Brahms model of a human-agent-robot scenario, we use the formal semantics described in [6] to generate a `Java` representation of the semantic structures relevant to this scenario. We then translate these structures into `Promela` process descriptions, which represent partial instantiations of the semantics, suitable for input to the `Spin` model checker [7]. In parallel, we translate the informal requirements (where possible) to `Promela` code representing these properties. Finally, we apply the `Spin` model checker to the `Promela` descriptions and feed the results back to the high-level scenarios for evaluation. This, then, provides a mechanism for formally verifying properties of human-agent-robot teamwork scenarios.

Our paper describes this framework and exhibits its use on a specific domestic scenario, where a helper robot and a house agent work together with a person to to monitor, remind and assist a person with their daily activities in a home environment.

2 Background

2.1 Brahms

Brahms is a multi-agent modelling, simulation and development environment devised by Sierhuis [5] and subsequently developed at NASA Ames Research Center. Brahms is designed to model both human and robotic activity using *rational agents*. Rational agents can be seen as autonomous entities, able to make their own choices and carry out actions in a *rational* and *explainable* way [8]. As Brahms was developed in order to represent people's activities in real-world contexts, it allows the representation of artifacts, data, and concepts in the form of classes and objects. Both agents and objects can be located in a model of the world giving agents the ability to detect both objects and other agents, have beliefs about the objects and move between locations. For a more detailed description of the Brahms language we refer the reader to [5] and [9]. The *key aspects* of Brahms are:

– *activities*: actions (with durations) an agent can perform;
– *beliefs*: each agent's own personal perceptions of itself, the environment and other agents;
– *facts*: the actual state of the agents and the environment (which agents/objects observe using *detectables*);
– *detectables*: allowing facts to be brought into the an agent's belief base and determining how the agent will react in response;
– *workframes*: sequences of events required to complete a task, together with any belief updates resulting from the task completion;
– *thoughtframes*: reasoning/thought processes, e.g. "I believe it is raining therefore I believe I need an umbrella";
– *time*: Brahms models incorporate a time-line of events and belief changes.

Brahms has been judged ideal for describing human-agent-robot teamwork, for example astronaut-robot teamwork on Mars [10].

A Brahms simulation contains a set of agents (representing robots, humans or actual agents) and a scheduling system which manages a clock recording global time within the simulation. Since agent actions have durations, the scheduler will examine each agent to see how much longer any action the agent is performing will take and then advance the clock to the next significant point in time, typically when the agent action finishes. When two agents finish actions at the same time (and at the start of the simulation) the scheduler also manages the order in which the agents execute their reasoning processes in order to determine their next action. In such cases the agents have a pre-determined order of priority within the scheduler.

2.2 Brahms Formal Semantics

In [6], we provided a formal *operational semantics* for Brahms. This provides the theoretical basis for our verification. A Brahms model is represented as a 5-tuple:

$$\langle Ags, ag_i, B_\xi, F, T_\xi \rangle$$

Where Ags is the set of all agents, ag_i the agent currently under consideration, B_ξ the belief base of the system (used to synchronise the agents, e.g. agent i's next event finishes in 1000 seconds), F the set of facts in the environment (e.g. temperature is $20°C$) and T_ξ is the current time of the system.

The agents (Ags, and ag_i) have a 9-tuple representation:

$$\langle ag_i, \mathcal{T}, W, stage, B, F, T, TF, WF \rangle$$

Here ag_i is the identification of the agent; $\mathcal{T}$ the current thoughtframe; W the current workframe; $stage$ the current stage of the agent's reasoning cycle; B the agent's beliefs; F the set of facts about the world; T the agent's internal time; TF the agent's thoughtframes; and WF is the agent's set of workframes. In addition, $stage$ controls which rules in the operational semantics are currently applicable to the agent or if the agent is in a finish (*fin*) or idle ($idle$) stage. The (operational) semantics is then represented as a set of transition rules of the form

$$\langle StartingTuple \rangle \xrightarrow[ConditionsRequiredForActions]{ActionsPerformed} \langle ResultingTuple \rangle$$

Here, '*ConditionsRequiredForActions*' refers to conditions which must hold before the rule can be applied, while '*ActionsPerformed*' represents changes to the agent, object or system state which, for presentational reasons, can not be easily represented in the resulting tuple. Finally, it is assumed that all agents and objects can see and access everything in the overall system's tuple, e.g. T_ξ.

The semantics is split into two groups of rules: the first group concerns the global system and represents the functioning of the scheduler; the other acts upon individual agents. Rules for the scheduler act as a global arbiters instructing agents when to start, suspend, or terminate. Rules for the individual agents choose actions and update beliefs, etc. An agent first processes *thoughtframes*, then *detectables* (both of which may update the beliefs), and then *workframes* which may initiate actions, referred to as *activities*.

For example, there are rules informing an agent on how to select a thoughtframe based on whether its beliefs match the thoughtframe guard conditions and whether the thoughtframe's priority is sufficiently high. The rules governing activities communicate with the system to inform it of the activity's duration. When no agent can apply any more operational rules, control returns to the scheduler which examines all the agents' activities to determine which will conclude first and at what time it will finish. The scheduler then moves the global clock forward accordingly, and hands control to the rules governing the behaviour of the individual agents once more.

2.3 Formal Verification, `Promela` and `SPIN`

Formal verification represents a family of techniques aimed at assessing whether a system always/ever satisfies its specification. We consider a fully automated, algorithmic technique known as *model checking* [11]. A model checker takes a description of the system together with some requirement expressed in a formal logic. The model checker exhaustively checks the formal requirement against *all* paths through the system. If a path is found in which the property does not hold then a trace of that path is provided.

In this paper we use the `Spin` model checker [7]. `Promela` (Process/Protocol Meta Language) is the input language for `Spin`. `Promela` was designed to be a simple multi-process language, allowing the models generated to be small. Processes are a key part of `Promela`. They are asynchronous and are declared by the key word `proctype`. `Promela` provides three basic control flow constructs: case selection; repetition; and unconditional jump.

`Spin` itself is an on-the-fly reachability analysis system [7]. It accepts specifications in the form of *linear temporal logic* properties, which are translated into *Büchi automata*

— finite automata over infinite input sequences. `Spin` then examines all possible runs through the `Promela` program, running the *Büchi automaton* in parallel in order to assess whether the temporal requirements are satisfied.

3 Case Study: "Domestic Home Care"

We will describe our translation and verification procedure through the development of one specific example; further details are available in the extended technical report [12]. We first describe the, necessarily very simple, scenario and outline its Brahms implementation. Though we can only provide an English overview here, we provide a sample Brahms workframe in Fig. 2; the full Brahms implementation is provided in [12].

3.1 Overview of Scenario

In this scenario there is a person, a helper robot, a human care worker and a house agent. The helper robot is mobile and can move about the house assisting the person with various tasks. The house agent has the role of detecting information, informing the person and issuing reminders where necessary. The care worker is called when the robot/agent are unable to assist. Such domestic health-care scenarios typically involve assisting the elderly or infirm; see for example [1, 2].

The helper robot: fetches drinks, cooks food, and delivers meals to the person; collects dirty dishes and puts them in the dishwasher; fetches medicines; records whether the person has taken these; informs the person of what to do in case of an emergency, e.g. a fire; and communicates with the house agent. The house agent: informs the helper robot of the person's location; issues reminders to the person (e.g. to flush the toilet); and monitors the person's location. The care worker is called for when the person fails to take their medication. We assume the care worker is always successful in administering the medication. The person is modelled very simply, watching TV, requesting food, eating and going to the toilet at regular intervals.

3.2 Brahms Representation

This scenario is modelled using five agents and one object: *Robot*, *House*, *Care-Worker*, *Environment* and *Bob* (our elderly person) are the agents and *Clock* is the object. The *Clock* is used for termination of the simulation (i.e. after 20 hours) and provides the notion of time used by the simulation e.g. governing when the human's hunger increases. The *Environment* is a simple agent that decides if, and when, a fire alarm will occur.

Bob's role is to watch television and perform simple everyday tasks such as eating and going to the toilet. Thoughtframes are used to update beliefs about how hungry he is and how much he needs the toilet. When his hunger reaches a certain threshold a workframe activates and *Bob* requests food. A similar workframe will trigger a visit to the toilet. These workframes have a higher priority than the workframe for watching television, so when they become active the 'television' workframe *suspends*. The workframe for going to the toilet activates other workframes to flush the toilet and wash his hands once finished. Two versions of these workframes exist: representing whether or not he remembers to perform the task, each have the same guard conditions and priority so only one will execute at random. *Bob* also has workframes for taking his medication and thoughtframes that govern whether or not he chooses to do so.

The helper *Robot* remains idle until it receives a command or it detects *Bob* requires attention. When *Bob* requests food, the *Robot* prepares and delivers it. There is a detectable in the *Robot's* "wait for instructions" workframe which detects when *Bob* has finished eating; this triggers a belief update which in turn triggers a workframe to clear the plates. The *Robot* also has workframes to deliver medicine to *Bob*; activated at pre-allocated times. The *Robot* places the drugs on *Bob's* tray and then monitors them hourly to check if they have been taken. The workframe governing this is shown in Fig. 2. A detectable `takenMedicationC` aborts the workframe if the drugs have been taken and then updates the *Robot's* beliefs. If the drugs have not been taken the workframe reminds *Bob* to take his medication. The *Robot* counts the number of times it reminds *Bob*, and after 2 reminders it notifies the *House*. The *Robot* also instructs *Bob*to evacuate the house in the case of fire and answers the door to the *Care Worker*.

```
workframe wf_checkMedicationC {
   repeat: true;
   priority: 3;
   detectables:
      detectable takenMedicationC{
         when(whenever)
         detect((Bob.hasMedicationC = false),
          dc:100)
         then abort; }
   when(knownval(current.perceivedtime > 14)and
      knownval(Bob.hasMedicationC = true) and
      knownval(current.checkMedicationC = true))
   do {
      checkMedication();
      remindMedicationC();
      conclude((current.checkMedicationC = false));
      conclude((current.missedMedicationC =
       current.missedMedicationC + 1)); }}
```

Fig. 2. The *Robot's* workframe to remind *Bob*about medication

The *House* is 'intelligent'. It is responsible for monitoring *Bob*, giving him instructions based on his location, and detecting any fire. The *House's* default workframe monitors *Bob*, and has detectables which update the *House's* beliefs about *Bob's* location. When *Bob's* location is at the toilet a new workframe is fired, containing an 'abort' detectable which quits the activity when *Bob* leaves the toilet and activates a new workframe which detects *Bob's* location and uses this to decide whether or not *Bob* has left without flushing the toilet. *Bob*is then reminded if necessary. The default monitoring workframe also has a detectable for a fire, this aborts the current activity and activates a workframe which sounds an alarm and notifies the *Robot* and *Bob*. Finally, while the *House* is notified that *Bob* has failed to take his medicine, it informs the *Care Worker*.

The *Care Worker* performs outside activities which are abstracted into a single "busy" activity. When the *Care Worker* is called he/she will only make their way to the house once they have finished their current activity. When the *Care Worker* arrives they ring the door bell. Once they have been let in by the *Robot* they administer the medication and inform the *Rpbpt* that the patient has now taken the medication. The *Care Worker* then leaves and continues with their outside activities.

Note that each of the agent's behaviours are here deliberately chosen to to be simple. We can, of course, add much more complex behaviour though our aim here is just to use this scenario to exhibit the overall approach.

4 From Brahms to `Promela`

We automatically build a `Promela` version of the Brahms scenario. In practice we translate Brahms into `Java` data structures corresponding to the semantic configurations (i.e., the various tuples mentioned in section 2.2) [6]. This is to allow us to (later) target several different model-checkers from the same intermediate representation. Here, however, we just discuss the final `Promela` code and do not detail the intermediate `Java` representation. `Promela`'s restrictive data types and control structures make it difficult to model the operational semantics for Brahms directly. Agents, workframes, thoughtframes and the tuples representing the system model all have to be represented via arrays. This makes it complex to write generic code that will apply to *any* model. As such we choose to generate a *partial instantiation* of the operational rules tailored for a particular model of interest. This partial instantiation is generated automatically from the `Java` representation.

4.1 From the Scheduler to a Promela Process

Representing the Scheduler in `Promela`. Given a specific model, we generate partial instantiations of the scheduler rules which act, not on a list of unknown agents, Ags, but upon the specific agents we know to exist in the model. The only variables used by the scheduler are an integer to represent its current time, '`cntEnvionment`'; an enumeration, '`turn`', which can be either an object/agent's name or the Environment; and a Boolean, '`EnvironmentActive`', which decides when the system is to terminate.

The `Promela` translation imitates the Brahms system scheduler by representing it as a `proctype`. The global clock is represented by an integer. Agents are also represented using `proctypes` and the scheduler determines their order of execution through '`turn`'. Once an agent has executed, '`turn`' is re-assigned to the scheduler.

Matching the Scheduler's Rules. The `Promela` code captures all the scheduler rules in a loop containing a conditional expression with one condition representing the guard for each rule. If the condition evaluates to true then code representing the rule's semantics is executed. We give an example of one of the scheduler rules, Sch_run, and discuss its instantiation as `Promela` code.

<u>RULE:</u> Sch_run

$$\langle Ags, ag_i, B_\xi, F, T_\xi\rangle \qquad \xrightarrow[\forall ag \in Ags | stage_{ag} \in \{fin, idle\}, (T_\xi \neq -1)]{stage_{ag_i} = Set\_Act} \qquad \langle Ags, ag_i, B_\xi, F, T_\xi\rangle$$

Sch_run becomes active if all the agents are either finished (in the *fin* stage) or idle (the *idle* stage) and the simulation hasn't finished ($T_\xi \neq -1$). In `Promela`:

- a set of Boolean variables represent when agents are *idle* (e.g., '`RobotActive`') is set to *false* if the *Robot* is *idle*); '
- a set of integers representing the time remaining for each agent's current activity are used to judge whether an agent is in the *fin* stage. (e.g., if '`Robot_timeRemaining`' is zero then the *Robot* is in the *fin* stage); and
- `Promela` will terminate if the simulation has concluded so it isn't necessary to check explicitly for $T_\xi = -1$.

The condition representing the rule's guard checks all these variables ('`RobotActive`', '`Robot_timeRemaining`' etc.) explicitly. In the generic rule, Sch_run sets the $stage$ of ag_i to Set_Act. In `Promela` the value of the agent's enumeration '`turn`' represents the agent's stage and this is set accordingly.

4.2 From Agent Semantics to Promela Processes

Representing the Agent's Data Structures in `Promela`. The components of the 9-tuple that represent an agent are primarily represented by arrays. These arrays are referred to by name in the partial instantiations of the operational rules.

For instance, $\mathcal{T}$, the agent's current thoughtframe is represented as a one-dimensional array and treated as a stack. The array is labelled '`tf_stack`' followed by the agent's name e.g. '`tf_stackRobot`'. The current workframe is represented in a similar fashion. The first six indices (three in the case of the current thoughtframe) of the array (elements 0-5) are used to store the workframe header data. Below the header information are a stack of *deeds* which may represent belief updates or activities. Sets of thoughtframes and workframes are stored in the same format but in two-dimensional arrays where the first index represents the thoughtframe or workframe and the second represents the elements of the thoughtframe or workframe. These are named '`tf`' or '`wf`' followed by the name of the agent. e.g. an agent *Robot* may have a set of workframes as follows:

Index	Workframe at index 0	index 1
0	Workframe ID number = 0	ID = 1
1	Boolean guard condition = 1 (workframe is active)	Guard = 0
2	Priority of the workframe = 4	Priority = 10
3	Repeat = 3 (always repeat)	Repeat = 0 (never repeat)
4	Boolean to flag a communication or move activity = 0	Comm/Move = 0
5	Boolean to flag if workframe is in impasse = 0	impasse = 1
6 . .	Last deed on stack . .	Last deed on stack . .
i	Top deed on stack	Top deed on stack

We do not represent the current stage of the agent's reasoning cycle explicitly, but do so implicitly by the order in which rules are represented in the `Promela` code.

Beliefs and facts in Brahms are tied to the `attributes` and `relations` of an agent; where attributes are defined properties of agents and relations are connections between agents. So agent `Robot` could believe agent `Bob's` attribute `AskedForFood` is true or that `Bob` has the relation of `isPatientOf` with the `Carer`. To model this in `Promela` every agent is assigned a belief about every `attibute` and `relation`, even if it does not own that attribute. This belief is represented as a Boolean array. The name of the belief is the name of the agent followed by the name of the attribute, e.g. `RobotAskedForFood` represents the `Robot's` beliefs about the attribute `AskedForFood`. The index of the array is the ID number of the agent whom the belief concerns, e.g.

0 = Robot's ID	*Robot* believes the Robot askedForFood = false
1 = Clock's ID	*Robot* believes the Clock askedForFood = false
2 = Bob's ID	*Robot* believes that *Bob* AskedForFood = true
3 = House's ID	Robot believes the House AskedForFood = false

Matching the Agent's Semantic Rules in `Promela`. When the scheduler's '`turn`' enumeration is an agent name then control passes to the agent rules. Like the scheduler rules these are represented by a loop that checks the rule pre-conditions in turn. To explain how the `Promela` translation matches Brahms we show how one of the operational semantic rules is represented in `Promela`.

RULE: *Wf_Select*

$$\langle ag_i, \emptyset, \emptyset, \mathit{Wf\_*}, B_i, F, T_i, \mathit{TF}_i, \mathit{WF}_i\rangle \xrightarrow[\exists W \in \mathit{WF}_i | B_i \models W^g]{\beta = Max_{pri}(W \in \mathit{WF}_i | B \models W^g)} \langle ag_i, \emptyset, \beta, \mathit{Wf\_(true/false/once)}, B_i, F, T_i, \mathit{TF}_i, \mathit{WF}_i\rangle$$

Wf_Select determines which workframe is to be selected. For the rule to be activated there needs to exist a workframe in the set of workframes whose guard conditions evaluates to true with respect to the belief base ($\exists W \in \mathit{WF}_i | B_i \models W^g$). At the start of each cycle the agent first identifies which workframes have guard conditions that evaluate to true: those which are active have a 1 entered at index 1 in the 2-dimensional array of workframes above, those that are not have a 0. The rule also states that the "current workframe" entry in the tuple must be empty, which is represented in `Promela` by a pointer to the current workframe's top element. If this is -1 then there is no current workframe. If the current workframe is empty and some workframe is active then Wf_Select will be selected.

Wf_Select performs a selection process to find the active workframe with the highest priority ($\beta = Max_{pri}(W \in \mathit{WF}_i | B \models W^g)$). The `Promela` translation loops through the array of workframes, checks the guard condition and the priority of each workframe; index 1 and 2 in the workframe array shown earlier. It builds a temporary array of workframes that share the maximum priority among all the active workframes. Finally the `Promela` code arbitrarily selects one workframe from this temporary array. For a further comparison with the semantics rules we refer the reader to the technical report [12].

4.3 Correctness Issues

As can be seen, we have not implemented the Brahms semantics directly in `Promela`. At present, analysis of this implementation consists of an informal comparison of the `Promela` arrays against the complex data structures of the semantics and an informal analysis of the operational rules against the partial instantiations produced for the specific example of the "Home Care" system. Parts of this analysis have been reproduced here and the full version can be found in [12]. In future work we intend to produce a more general, though still informal, discussion of the translation mechanisms themselves. It should be noted that there is also no proof that the operational semantics accurately capture Brahms. So both systems can be viewed separately as mechanisms for exploring models of human-agent teamwork even if they are not provably equivalent.

5 "Home Care" Verification

We next consider the actual verification of human-agent-robot teamwork; again we focus on the "home care" scenario.

5.1 Requirements

We develop a range of logical requirements for the scenario; recall that in *temporal logic*, $\Diamond\phi$ means that "ϕ will be true at *some* moment in the future", while $\Box\phi$ means that "ϕ will be true at *all* future moments". We describe some of the properties verified and classify these just by the core aspect they represent, i.e. properties labelled F*n* relate to the fire alarm; labelled by T*n* relate to the toilet; H*n* relate to hunger and M*n* relate to medicine. For space reasons, we only provide the temporal formulae in the case of the fire alarm. The axioms used in the properties are all based on the beliefs of the agents or facts in the system. We expect all of these properties to hold apart from M1.

F1: If a fire actually occurs then, eventually, *House* will generate a fire alarm. Logical requirement is: $\Box(\mathrm{a} \Rightarrow \Diamond\ \mathrm{b})$ where
  a = there is a fire
  b = *House* believes there is a fire alarm

F2: If fire alarm is sounding, and *Bob* leaves *House* then fire alarm finishes. Logical requirement is: $\Box((a \wedge b) \Rightarrow \Diamond\neg a)$ where
  a = *House* believes there is a fire alarm
  b = *Bob* believes he has evacuated the house

F3: If fire alarm is sounding and *Bob* has not left *House*, then *Robot* reminds *Bob*. Logical requirement is: $\Box((a \wedge b) \Rightarrow \Diamond\neg c)$ where
  a = *House* believes there is a fire alarm
  b = *Bob* believes he has evacuated the house
  c = *Robot* believes it has alerted *Bob* of the fire 0 times

T1: Eventually *Bob* will go to toilet.

T2: If *Bob* goes to the toilet he can forget to flush it and, if so, he will be reminded by the House. So, if *Bob* goes the toilet then eventually he will flush the toilet.

H1: If *Bob* requests food then eventually *Robot* will deliver the food within an hour.

H2: Once *Bob* has finished eating, *Robot* will then retrieve the dishes and place in the dishwasher.
M1: Either *Bob* always takes his medication or the *Robot* never reminds him to do so. (This should be false since *Bob* may not take his medication even if reminded).
M2: If *Bob* has medication, but not taken it, then *Robot* will eventually remind *Bob* to take it.
M3: *Bob* takes medicine or *House* is informed that *Bob* has not taken it.
M4: If *Care Worker* is informed that *Bob* has not taken his medication then the *Care Worker* is with *Bob* within 2 hours and *Bob* takes his medication.

5.2 Verification Results

The properties F1, F2, F3, T1, T2, H1, H2, M2, M3 were all verified using `Spin` (i.e. the property holds on all paths from every initial state) in times ranging from T1 of 29.9 seconds to H1 of 848 seconds. As expected `Spin` shows that the property M1 is false and the time taken to find a trace in the model was 421 seconds. The property M4 was run multiple times to observe how changing the duration of the *Care Worker's* other duties affected the outcome. `Spin` was able to verify M4 so long as the *Care Worker's* other duties took less than 2 hours.

6 Conclusions

In this paper we have presented an overview of our work in verifying human-agent teamwork using the `Spin` model checker and the Brahms teamwork modelling system. Brahms enables the description of human-agent teamwork scenarios where the defining factors are the actions taken, their timing, duration and results. It has proven useful in the analysis of such scenarios via simulation. By adding verification to Brahms we extend its usefulness by allowing *all* possible simulations (with fixed time granularities) to be explored, thus ensuring that undesirable outcomes *can not* arise within the model.

A simple case study was presented, demonstrating the kind of human-agent teamwork scenarios we intend to verify. This case study included sample verified properties. The case study demonstrates most of the core capabilities of Brahms: multi-tasking by suspending actions in favour of higher priority ones; detecting changes in the environment; aborting actions; choosing between actions of equal priority; and communicating to coordinate actions.

The properties we verified were necessarily simple. However, it should be clear that as long as the properties can be represented in a straight-forward temporal language, then model-checking can be carried out. When humans are involved, we abstract their behaviour within the Brahms model and describe their requirements in logical terms. Whether human participants live up to these requirements is, of course, up to others to assess.

6.1 Related and Future Work

There are relatively few tools available for the analysis of human-agent teamwork. Brahms is one of the few that is used in the analysis of real systems. At present Brahms is, essentially, a testing tool and is used to examine multiple simulations of a model in a search of undesirable outcomes (e.g. Extra Vehicular Activities in space [10, 13]).

As far as we are aware there is no tool for the formal analysis via model checking of such scenarios. However BDI-style agent programming languages are a natural tool for creating such models, with their emphasis on modelling the reasoning of autonomous agents in terms of their beliefs and goals. A number of systems have been developed for model-checking programs in agent languages [14–16] though none of these have yet been applied to human-agent-robot teamwork.

In future we aim to improve the efficiency of the verification and to analyse more complex scenarios with multiple agents, cooperating and coordinating efforts in a much larger team. Scalability of the verification will be tested on these new scenarios. Scenarios under consideration include search and rescue; factory work; and hospitals. We also aim to investigate the verification of Brahms models in other model checkers, particularly ones with input languages which let us capture the operational semantics in a more intuitive fashion. This would provide a better guarantee of equivalence to Brahms simulations and it would also provide a point of comparison for evaluating the efficiency of the model checkers. The Java Pathfinder system [17] is an obvious candidate for this, either by implementing the Brahms semantics directly in Java or by using the AIL tool-kit for modelling agent languages and AJPF, its associated JPF based model checker [16].

Acknowledgements. This work was funded by EPSRC through EP/F033567 and EP/F037201.

References

1. Montemerlo, M., Pineau, J., Roy, N., Thrun, S., Verma, V.: Experiences with a mobile robotic guide for the elderly. In: Eighteenth National Conference on Artificial Intelligence, pp. 587–592. American Association for Artificial Intelligence, Menlo Park (2002)
2. Pineau, J., Montemerlo, M., Pollack, M., Roy, N., Thrun, S.: Towards robotic assistants in nursing homes: Challenges and results. Robotics and Autonomous Systems 42, 271–281 (2003)
3. Lenz, C., Nair, S., Rickert, M., Knoll, A., Rosel, W., Gast, J., Bannat, A.: Joint-action for Humans and Industrial Robots for Assembly Tasks. In: Proc. 17th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 130–135 (2008)
4. CHRIS: Cooperative Human Robot Interaction Systems (2011), `http://www.chrisfp7.eu`
5. Sierhuis, M.: Modeling and Simulating Work Practice. BRAHMS: a multiagent modeling and simulation language for work system analysis and design. PhD thesis, Social Science and Informatics (SWI), University of Amsterdam, SIKS Dissertation Series No. 2001-10, Amsterdam, The Netherlands (2001)
6. Stocker, R., Sierhuis, M., Dennis, L., Dixon, C., Fisher, M.: A Formal Semantics for Brahms. In: Leite, J., Torroni, P., Ågotnes, T., Boella, G., van der Torre, L. (eds.) CLIMA XII 2011. LNCS, vol. 6814, pp. 259–274. Springer, Heidelberg (2011)
7. Holzmann, G.: The Spin Model Checker: Primer and Reference Manual. Addison-Wesley (2003)
8. Wooldridge, M.: An Introduction to Multiagent Systems. John Wiley & Sons (2002)

9. Sierhuis, M.: Multiagent Modeling and Simulation in Human-Robot Mission Operations (2006), http://ic.arc.nasa.gov/ic/publications
10. Clancey, W., Sierhuis, M., Kaskiris, C., van Hoof, R.: Advantages of Brahms for Specifying and Implementing a Multiagent Human-Robotic Exploration System. In: Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS), pp. 7–11. AAAI Press (2003)
11. Clarke, E., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (2000)
12. Stocker, R., Dennis, L., Dixon, C., Fisher, M.: Spin Verification of Brahms Human-Robot Teamwork Models (2012), http://www.csc.liv.ac.uk/~rss/Publications.html
13. Hirsh, R., Tyree, K., Johnson, N., Johnson, N.: Intelligence for human-assistant planetary surface robots. In: IntelZigence for Space Robotics, pp. 261–279. TSI Press (2006)
14. Jongmans, S.-S.T.Q., Hindriks, K.V., van Riemsdijk, M.B.: Model Checking Agent Programs by Using the Program Interpreter. In: Dix, J., Leite, J., Governatori, G., Jamroga, W. (eds.) CLIMA XI. LNCS, vol. 6245, pp. 219–237. Springer, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-14977-1_17
15. Bordini, R.H., Fisher, M., Pardavila, C., Wooldridge, M.: Model checking AgentSpeak. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS (2003)
16. Dennis, L.A., Fisher, M., Webster, M., Bordini, R.: Model Checking Agent Programming Languages. Automated Software Engineering 19, 5–63 (2012)
17. Visser, W., Havelund, K., Brat, G.P., Park, S., Lerda, F.: Model Checking Programs. Automated Software Engineering 10, 203–232 (2003)

On Satisfiability in ATL with Strategy Contexts

Nicolas Troquard[1] and Dirk Walther[2]

[1] Laboratory for Applied Ontology (ISTC-CNR), Trento, Italy
[2] Universidad Politécnica de Madrid, Spain

Abstract. This paper is a study of Brihaye *et al.*'s ATL with strategy contexts. We focus on memory-less strategies and establish that the resulting logic is undecidable. An immediate corollary follows that the problem of satisfiability checking of every variant of ATL with strategy context introduced by Brihaye *et al.* is undecidable. We also relate ATL_{sc} with memory-less strategies with ATL with explicit strategies, providing a decidable fragment.

1 Introduction

With Alternating-time Temporal Logic $\mathsf{ATL}^{(*)}$ ([2,14]) one can reason about the *ability* of a coalition to ensure something *whatever the other agents do.* It is the logic of sentences like "The monitoring units $u_1, \dots, u_l$ can ensure that the system stays in a failsafe state." In this paper, we consider the recent variant of ATL with strategy contexts [4,6]. A *strategy context* is the actual current strategy of some committed set of agents. The truth value of an ATL_{sc}-formula is evaluated in a concurrent game structure, at a state, and *wrt. a strategy context.* Informally, the formula $\langle\cdot A\cdot\rangle\psi$ states that A has a strategy to ensure the property ψ in the context of the current strategy commitment. Like in ATL, the formula ψ typically represents a temporal property, but unlike the ATL path quantifier, the modality $\langle\cdot A\cdot\rangle$ *commits* the members of A to their chosen strategy F_A. Henceforth, the commitment is used for the evaluation of ψ. That is, ψ is evaluated wrt. to a strategy context consisting in the initial strategy context updated with F_A. The operator $\cdot\rangle A\langle\cdot$ *releases* this commitment. Under the common assumptions of ATL, the ATL path quantifier is trivially captured by

$$\langle\langle A\rangle\rangle\psi \overset{\text{def}}{=} \cdot\rangle\Sigma\langle\cdot\langle\cdot A\cdot\rangle\psi,$$

where Σ is the set of all agents.

The notion of ability of a coalition in ATL_{sc} is their ability given the context of the strategies that the coalition is actually committed to. *Actual agency*, the property of some agentive entity in the act of doing something, is ubiquitous in our everyday life: "Unit u_1 is inspecting the register $0x12345678$." It is all the more important in a multi-agent framework where agents strategise given some input (observation, expectation, belief, etc.) about the strategies followed by the other players, and their abilities depend on it: "If units $u_1, \dots, u_{i-1}, u_{i+1}, \dots, u_l$ do not know which register u_i is inspecting, they cannot ensure that no system failure will occur." Actual agency is also central to game theory, where for instance, a Nash equilibrium occurs when every agent is playing his best response to the current strategy of the other agents. With the advent of the Internet and service-oriented computing, system designers in industry and in academia rely increasingly on the multi-agent paradigm. As we seek after the 'next generation' of logics for the specification of properties of societies of agents, and for the verification of their designs, it appears important to be able to talk and reason about actual agency of coalitions of agents, and their contextualised ability.

ATL_{sc} and ATL^*_{sc} can capture a variety of notions of *strategic actual agency* that lie beyond the mere ability of coalitions as captured by ATL. For instance, a type of STIT modality ([11,5]) can be defined as

$$[A\ \mathsf{sstit}]\psi \overset{\text{def}}{=} \cdot\rangle \Sigma \setminus A \langle\cdot \langle\cdot \emptyset \cdot\rangle \psi,$$

reading "A is seeing to it that ψ." (See the earlier report [18] for a detailed discussion about ATL and STIT modalities.)

In the pure tradition of knowledge representation it is also useful to be able to talk about strategies in a more explicit manner. Practically, they can serve, e.g., as explicit delegation instruction between agents. We will contrast the use of strategy contexts with *explicit strategies.* ATL_{sc} and ATLES ([19]) capture the notions of commitment to, release and recall of strategies, as well as irrevocable strategies ([1]). We introduce ATLES on concurrent game structures in Section 3 and relate ATL_{sc} with ATLES, determining a decidable fragment of ATL_{sc}.

Originating from theoretical computer science and verification, the focus of ATL_{sc} has been on model checking so far, and not satisfiability. In Section 4, we establish that the satisfiability problem for both ATL_{sc} and ATL^*_{sc} is undecidable in general, emphasising the significance of the fragment previously identified.

In the next section we define rigorously the syntax and semantics of ATL_{sc} and ATL^*_{sc} that we have informally presented in this introduction.

2 ATL with Strategy Contexts

We fix a countable set of *atomic propositions* $\mathbf{\Pi}$ and a finite set of *agents* (or *players*) $\mathbf{\Sigma}$. The following grammar was given for ATL^*_{sc} in [6].

Definition 1 (ATL^*_{sc} syntax). *The following grammar defines state formulas φ and path formulas ψ, where p ranges over $\mathbf{\Pi}$ and A over finite subsets of $\mathbf{\Sigma}$. The language of ATL^*_{sc} consists of the state formulas.*

$$\begin{aligned} \varphi &::= \ p \mid \neg\varphi \mid \varphi \vee \varphi \mid \cdot\rangle A \langle\cdot \varphi \mid \langle\cdot A \cdot\rangle \psi \\ \psi &::= \ \varphi \mid \neg\psi \mid \psi \vee \psi \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi \end{aligned}$$

The remaining Boolean operators $\wedge$, $\rightarrow$ and $\leftrightarrow$ as well as the logical constants $\top$ and $\bot$ can be defined as usual in terms of the operators given. The linear temporal logic operators 'sometime' and 'forever' can be defined as path formulas $\Diamond\varphi = (\top\,\mathcal{U}\,\varphi)$ and $\Box\varphi = \neg(\top\,\mathcal{U}\,\neg\varphi)$.

The language of ATL_{sc} consists only of some formulas from ATL^*_{sc}. The syntax of the path formulas ψ is restricted as follows (where φ refers to the state formulas in Def. 1):

$$\psi ::= \neg\psi \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi$$

We evaluate the formulas on Concurrent Game Structures (CGSs), which are defined as follows.

Definition 2 (Concurrent Game Structure). *Let $\Sigma = \{1, \dots, n\} \subset \mathbf{\Sigma}$, with $n \geq 1$, be a finite set of agents, and $\Pi \subset \mathbf{\Pi}$ be a finite set of atomic propositions. A* Concurrent Game Structure *(CGS) $\mathcal{C}$ for $\langle \Sigma, \Pi \rangle$ is a tuple $\mathcal{C} = \langle W, V, \Sigma, M, Mov, E \rangle$, where:*

- *W is a non-empty set of worlds (or game positions);*
- *$V : W \to 2^{\Pi}$ is a valuation function;*
- *M is a finite, non-empty set of moves;*
- *$Mov : W \times \Sigma \to 2^M \setminus \emptyset$ maps every world w and agent a to the non-empty set $Mov(w, a)$ of moves available to a at w; and*
- *$E : W \times M^{\Sigma} \to W$ is a transition function mapping a world w and a move profile $\boldsymbol{m} = \langle m_1, \dots, m_n \rangle$ (one move for each agent) to the world $E(w, \boldsymbol{m})$.*

Let $\mathcal{C}$ be a CGS. The component Mov determines which of the moves from M are available for an agent at a world w. Let $\mathsf{prof}(w)$ be the set of available move profiles at world w, i.e.,

$$\mathsf{prof}(w) = \{\langle m_1, \dots, m_n \rangle \mid m_i \in Mov(w, i)\}.$$

A move profile is used to determine a successor of a world using the transition function E. Let $\mathsf{succ}(w)$ be the set of possible successors of w, formally

$$\mathsf{succ}(w) = \{E(w, \boldsymbol{m}) \mid \boldsymbol{m} \in \mathsf{prof}(w)\}.$$

An infinite sequence $\lambda = x_0 x_1 x_2 \cdots \in W^{\omega}$ of worlds is called a *play* or *computation* if $x_{i+1} \in \mathsf{succ}(x_i)$ for all positions $i \geq 0$. Denote with $\lambda[i]$ the i-th component x_i in λ, and with $\lambda[0, i]$ the initial sequence $x_0 \cdots x_i$ of λ.

A *strategy for an agent $a \in \Sigma$* is a function f_a that maps a world w from W to a move profile $f_a(w) \in Mov(w, a)$ available to a at w. A *strategy for a coalition $A \subseteq \Sigma$* is a set F_A of strategies with $F_A = \{\sigma_a \mid a \in A\}$ containing one strategy for every agent in A. We refer to a strategy also as *strategy context.* We denote with $\mathsf{strat}(A)$ the set of strategies available to coalition A. The strategies considered here are *memoryless* as they are functions from worlds to move profiles and, thus, do not take previously visited states into account.

We define two operations on strategies: upgrade and release of strategies. Let F_A and F be strategies for sets of agents, where F_A contains strategies for the agents in A. The *upgrade* of F with the strategies in F_A is the result of *overwriting* F with strategies for the agents in $A \cap \mathsf{dom}(F)$ and *supplementing* F with strategies for agents for which F does not already provide a strategy (i.e., for agents in $A \setminus \mathsf{dom}(F)$). We will use $\circ$ as a strategy upgrade operator. Formally,

$$F_A \circ F = F_A \cup \{f_a \in F \mid a \notin A\}.$$

The *release* of the strategies for the agents in B from F is the *restriction* of F to strategies for agents that do not occur in B (i.e., for agents in $\Sigma \setminus B$). Formally, for $C = \Sigma \setminus B$,

$$F|_C = \{f_a \in F \mid a \in C\}.$$

The set $\mathsf{out}(w, F_A)$ of *outcomes* of a strategy F_A for the agents in A starting at a world w is the set of all plays $\lambda = x_0x_1x_2 \cdots \in W^\omega$ such that $x_0 = w$ and, for every $i \geq 0$, there is a move profile $\boldsymbol{m} = \langle m_1, \ldots, m_n\rangle \in \mathsf{prof}(x_i)$ such that:

(i) $m_a = f_a(x_i)$, for all $a \in A$; and
(ii) $x_{i+1} = E(x_i, \boldsymbol{m})$.

The truth values of ATL^*_{sc}-formulas over CGSs is given as follows, where state formulas are evaluated at worlds (or game positions) and path formulas over infinite paths in a CGS.

Definition 3 (ATL^*_{sc} Semantics). *Given a CGS $\mathcal{C} = \langle W, R, V, \Sigma, M, Mov, E\rangle$ for $\langle \Sigma, \Pi\rangle$ and a strategy context F, the consequence relation $\models$ is inductively defined as follows:*

- $\mathcal{C}, w \models_F p$ *iff* $p \in V(w)$, *for all atomic propositions* $p \in \Pi$;
- $\mathcal{C}, w \models_F \neg\varphi$ *iff* $\mathcal{C}, w \not\models_F \varphi$;
- $\mathcal{C}, w \models_F \varphi_1 \vee \varphi_2$ *iff* $\mathcal{C}, w \models_F \varphi_1$ *or* $\mathcal{C}, w \models_F \varphi_2$;
- $\mathcal{C}, w \models_F \cdot\rangle A\langle\cdot\varphi$ *iff* $\mathcal{C}, w \models_S \varphi$, *where* $S = F|_{\Sigma\setminus A}$;
- $\mathcal{C}, w \models_F \langle\cdot A\cdot\rangle\psi$ *iff there is* $F_A \in \mathsf{strat}(A)$ *such that for all plays* $\lambda \in \mathsf{out}(w, S)$, *it holds that* $\mathcal{C}, \lambda \models_S \psi$, *where* $S = F_A \circ F$;
- $\mathcal{C}, \lambda \models_F \varphi$ *iff* $\mathcal{C}, \lambda[0] \models_F \varphi$, *when* φ *is a state formula;*
- $\mathcal{C}, \lambda \models_F \neg\psi$ *iff* $\mathcal{C}, \lambda \not\models_F \psi$;
- $\mathcal{C}, \lambda \models_F \psi_1 \vee \psi_2$ *iff* $\mathcal{C}, \lambda \models_F \psi_1 \vee \psi_2$;
- $\mathcal{C}, \lambda \models_F \bigcirc\varphi$ *iff* $\mathcal{C}, \lambda[1] \models_F \varphi$;
- $\mathcal{C}, \lambda \models_F (\varphi_1 \,\mathcal{U}\, \varphi_2)$ *iff there is an* $i \geq 0$ *such that* $\mathcal{C}, \lambda[i] \models_F \varphi_2$ *and* $\mathcal{C}, \lambda[j] \models_F \varphi_1$ *for all* j *with* $0 \leq j < i$.

A formula φ is satisfiable *if $\mathcal{C}, w \models_F \varphi$ for some CGS $\mathcal{C}$, some world w in $\mathcal{C}$ and some strategy context F in $\mathcal{C}$; a formula is called* valid *if $\mathcal{C}, w \models_F \varphi$ for all $\mathcal{C}$, all w and all F.*

In this paper, we do not assume agents being capable of perfect recall. In fact, we use a semantics for ATL_{sc} and ATL^*_{sc} that is based on memoryless strategies. This means that agents use strategies that prescribe for every world which move to take. The history of previously visited worlds is not taken into account. In [4,6], these logics are denoted with $\mathsf{ATL}_{\mathrm{sc},0}$ and $\mathsf{ATL}^*_{\mathrm{sc},0}$.

3 Strategy Contexts and Explicit Strategies

In this section, we contrast the notion of strategy contexts with explicit strategies. Many notions relevant to strategies come into the picture and our principal aim is to discuss them informally. We first present ATLES, the extension of ATL with explicit strategies from [19] (Section 3.1). We introduce it over CGSs while its original presentation was in terms of alternating transition systems. We then translate a fragment of ATL_{sc} into ATLES (Section 3.2).

3.1 ATLES

The language of ATL is enriched with symbols for strategies and commitment functions that assign agents to strategies they are committed to play. Thus ATLES allows to reason explicitly about strategies. This is not possible with any of ATL and ATL_{sc} (and their respective LTL-extensions) as strategies are pure semantic constructs and they do not occur in the object language.

Formally, the signature of the language is extended by a set $\boldsymbol{\Upsilon}$ of *strategy terms*, where $\boldsymbol{\Upsilon} = \bigcup_{a \in \boldsymbol{\Sigma}} \boldsymbol{\Upsilon}_a$ and $\boldsymbol{\Upsilon}_a$ is a countably infinite set of *strategy terms* σ_a for agent a in $\boldsymbol{\Sigma}$. A *commitment function* is a partial function $\rho : \boldsymbol{\Sigma} \rightarrow \boldsymbol{\Upsilon}$ with a finite domain mapping an agent $a \in \boldsymbol{\Sigma}$ to a strategy term $\rho(a) \in \boldsymbol{\Upsilon}_a$ for a. Note that a commitment function ρ is a finite object and as such it is used to additionally parameterise path-quantifiers as $\langle\!\langle A \rangle\!\rangle_\rho$. The set $\mathsf{dom}(\rho)$ consists of the committed agents. If $\rho(a)$ is defined, then ρ contains a mapping of the form $a \mapsto \sigma_a$ which is called a *commitment* of agent a (or *a commits*) to play the strategy denoted by the strategy term σ_a. On the other hand, if $\rho(a)$ is undefined, then a does not commit to any strategy and, thus, a can quantify freely over the strategies available to a. The reading of an ATL-path quantifier $\langle\!\langle A \rangle\!\rangle$ with commitment function ρ is as follows:

> $\langle\!\langle A \rangle\!\rangle_\rho \varphi$ states that, given the commitment of any agent b in $\mathsf{dom}(\rho)$ to use the strategy denoted by $\rho(b)$, the agents in $A \setminus \mathsf{dom}(\rho)$ have a strategy to ensure the temporal property φ, no matter what the agents in $\Sigma \setminus (\mathsf{dom}(\rho) \cup A)$ do.

Notice that the committed agents in $\mathsf{dom}(\rho)$ do not take part in the quantification over strategies in $\langle\!\langle A \rangle\!\rangle_\rho$.

We remark that $\langle\!\langle A \rangle\!\rangle_\rho$ is not how the path quantifier really looks like when used in a formula. The symbol ρ is merely a meta-logical reference to an actual commitment function, which is a collection of mappings of the form $a \mapsto \sigma_a$, where σ_a is a strategy term for agent a. This should be considered when analysing the length of a formula.

The notion of commitment to strategies requires the same strategies to be played again at a later stage. This means, in formulas of the form $\langle\!\langle A \rangle\!\rangle_\rho \Psi$, the same commitment $a \mapsto \sigma_a$ from ρ occurs in a commitment function ξ of a nested path quantifier $\langle\!\langle B \rangle\!\rangle_\xi$ in Ψ. Both ρ and ξ prescribe the strategy term σ_a for agent a (or, in both cases, a commits to σ_a). We have that $\rho(a) = \xi(a)$. Release of commitment to σ_a is modelled as easily as committing to it in the first place. This is achieved by having a commitment function χ of a nested path quantifier not include the commitment $a \mapsto \sigma_a$, i.e., either $\chi(a) \neq \sigma_a$ or χ is undefined for a. In case release of commitment is not desired, the notion of irrevocable strategies is used. It can be modelled explicitly in ATLES by only allowing commitment functions ρ to extend conservatively the commitment functions ξ under whose range they occur, i.e., ρ and ξ agree for all agents in $\mathsf{dom}(\xi)$. Thus, IATL can be defined in ATLES while avoiding the update semantics employed in [1].

The language of ATLES is defined over the extended signature $\langle \boldsymbol{\Pi}, \boldsymbol{\Sigma}, \boldsymbol{\Upsilon} \rangle$.

Definition 4 (ATLES **Syntax).** *The following grammar defines state formulas φ and path formulas ψ, where p ranges over $\mathbf{\Pi}$, A ranges over finite subsets of $\mathbf{\Sigma}$ and ρ over commitment functions. The language of* ATLES *consists of state formulas.*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\!\langle A \rangle\!\rangle_\rho \psi$$
$$\psi ::= \bigcirc\varphi \mid \Box\varphi \mid \varphi\,\mathcal{U}\,\varphi$$

The language of ATLES could easily be extended to allow for negation of the temporal operators next-time and until. We refrain from extending the syntax in this paper as we use the established complexity result of the satisfiability problem for ATLES from [19] in order to use ATLES to determine a decidable fragment of ATL_{sc} whose satisfiability can be solved in ExpTime.

Strategy terms in $\mathbf{\Upsilon}$ are interpreted as strategies in a CGS via assignments. An *assignment* $\mathfrak{a}$ in $\mathcal{C}$ is a function mapping strategy terms σ_a in $\mathbf{\Upsilon}_a$ for any agent a in $\mathbf{\Sigma}$ to a strategy $\mathfrak{a}(\sigma_a)$ in $\mathsf{strat}(a)$ for a in $\mathcal{C}$. Note that the assignment $\mathfrak{a}$ in a CGS acts like an assignment in First-order Logic with the difference that in ATLES strategy terms are mapped to actual strategies in the CGS instead of domain elements as in FOL. In [19] an assignment is called denotation function, which comes as a component of an ATS.

To define the semantics of ATLES, we use the notions of a strategy and outcome as in Section 2. We lift the notion of assignment to commitment functions as follows. The application of an assignment $\mathfrak{a}$ to a commitment function ρ is the set $\mathfrak{a}(\rho)$ of strategies for the agents in $\mathsf{dom}(\rho)$. Formally,

$$\mathfrak{a}(\rho) = \{f_a \in \mathsf{strat}(a) \mid f_a = \mathfrak{a}(\rho(a)), a \in \mathsf{dom}(\rho)\}.$$

It is readily checked that $\mathfrak{a}(\rho)$ is indeed a set of strategies, one for each agent in $\mathsf{dom}(\rho)$. To see this, recall that ρ is functional, i.e., it yields exactly one strategy term $\rho(a)$ for every agent for which ρ is defined.

An assignment $\mathfrak{a}$ acts as an interpretation of the commitment function ρ (i.e. the strategy terms in ρ). We can view a strategy term $\sigma_a = \rho(a)$, for any a in $\mathsf{dom}(\rho)$, as a constant rather than a variable. As we will see below in the semantics of ATLES, the assignment $\mathfrak{a}$ does not change during the evaluation of a formula and, thus, the strategy $\mathfrak{a}(\sigma_a)$ is fixed. We can think of the strategy term σ_a as being existentially quantified in the sense that there exists a strategy for a that is referenced by σ_a and provided by $\mathfrak{a}$. ATLES does not provide references to universally quantified strategies.

Using the notion of assignments, we can now define how to interpret the formulas of ATLES over CGSs.

Definition 5 (ATLES **Semantics).** *Given a CGS $\mathcal{C} = \langle W, R, V, \Sigma, M, Mov, E\rangle$ for $\langle\Sigma, \Pi\rangle$ and an assignment $\mathfrak{a}$, the consequence relation $\models$ is inductively defined as follows, and the notions of validity and satisfiability are defined as usual:*

- *$\mathcal{C}, w \models^{\mathfrak{a}} p$ iff $w \in V(p)$, for all atomic propositions $p \in \Pi$;*
- *$\mathcal{C}, w \models^{\mathfrak{a}} \neg\varphi$ iff $\mathcal{C}, w \not\models^{\mathfrak{a}} \varphi$;*
- *$\mathcal{C}, w \models^{\mathfrak{a}} \varphi_1 \vee \varphi_2$ iff $\mathcal{C}, w \models^{\mathfrak{a}} \varphi_1$ or $\mathcal{C}, w \models^{\mathfrak{a}} \varphi_2$;*

- $\mathcal{C}, w \models^{\mathfrak{a}} \langle\!\langle A \rangle\!\rangle_\rho \psi$ *iff there is a strategy* F_A *in* $\mathsf{strat}(A)$ *such that for all plays* $\lambda \in \mathsf{out}(w, S)$*, it holds that* $\mathcal{C}, \lambda \models^{\mathfrak{a}} \psi$*, where* $S = \mathfrak{a}(\rho) \circ F_A$*;*
- $\mathcal{C}, \lambda \models^{\mathfrak{a}} \bigcirc\varphi$ *iff* $\mathcal{C}, \lambda[1] \models^{\mathfrak{a}} \varphi$*;*
- $\mathcal{C}, \lambda \models^{\mathfrak{a}} \Box\varphi$ *iff* $\mathcal{C}, \lambda[i] \models^{\mathfrak{a}} \varphi$ *for all positions* $i \geq 0$*;*
- $\mathcal{C}, \lambda \models^{\mathfrak{a}} (\varphi_1 \mathcal{U} \varphi_2)$ *iff there is an* $i \geq 0$ *such that* $\mathcal{C}, \lambda[i] \models^{\mathfrak{a}} \varphi_2$ *and* $\mathcal{C}, \lambda[j] \models^{\mathfrak{a}} \varphi_1$ *for all positions* j *with* $0 \leq j < i$.

The ATLES semantics of $\langle\!\langle A \rangle\!\rangle_\rho$ is similar to the semantics of $\langle \cdot A \cdot \rangle$ in ATL_{sc}, which facilitates comparison. We recall that the operator $\circ$ from Section 2 yields $\mathfrak{a}(\rho) \circ F_A = \mathfrak{a}(\rho) \cup \{f_a \in F_A \mid a \notin \mathsf{dom}(\rho)\}$. Intuitively, $\mathfrak{a}(\rho) \circ F_A$ states that commitments of agents are respected as prescribed in ρ, all other agents in A play their just selected strategies.

3.2 Comparing ATL_{sc} and ATLES

Obvious differences between ATL_{sc} and ATLES are that, while the former includes a separate release operator $\cdot\rangle A \langle\cdot$ and a strategy context in the semantics, the latter allows to specify commitments in the form of $a \mapsto \sigma_a$ in the syntax which are interpreted using assignments. However, commitments and assignments in ATLES can play the roles of strategy release and strategy contexts in ATL_{sc}. A crucial difference between the logics is the semantics of the path quantifiers $\langle \cdot A \cdot \rangle$ and $\langle\!\langle A \rangle\!\rangle_\rho$. For $\langle \cdot A \cdot \rangle$, the strategies F_A selected by A upgrade or overwrite the strategy context F_{context} (cf. Def. 3), whereas, for $\langle\!\langle A \rangle\!\rangle_\rho$, the strategies $\mathfrak{a}(\rho)$ specified by the commitment ρ are supplemented by F_A (cf. Def. 5). The set of plays considered for further evaluation depends on the upgraded context $F_A \circ F_{\mathsf{context}}$ or the supplemented commitments $\mathfrak{a}(\rho) \circ F_A$. Both are not necessarily equivalent. The following proposition states under which conditions $\langle \cdot A \cdot \rangle$ and $\langle\!\langle A \rangle\!\rangle_\rho$ determine the same set $\mathsf{out}(x, S)$ of plays, where S is defined as $S = F_A \circ F_{\mathsf{context}}$ in the former case, and $S = \mathfrak{a}(\rho) \circ F_A$ in the latter.

Proposition 1. *It holds that* $F_A \circ F_{\mathsf{context}} = \mathfrak{a}(\rho) \circ F_A$ *if one of the following three conditions is satisfied:*

(i) $F_{\mathsf{context}} = \mathfrak{a}(\rho) = \emptyset$*;*
(ii) $F_A = \emptyset$ *and* $F_{\mathsf{context}} = \mathfrak{a}(\rho)$*; or*
(iii) $F_A = F_{\mathsf{context}} = \mathfrak{a}(\rho)$.

The proposition can be shown by using the fact that the strategy upgrade operator $\circ$ forms an idempotent semigroup on the set strat of strategies, and that $\circ$ is not commutative.

Proposition 1 makes clear that a strategy context F_{context} in ATL_{sc} corresponds to the strategy commitment $\mathfrak{a}(\rho)$ in ATLES with the difference that F_{context} is a purely semantic object, whereas $\mathfrak{a}(\rho)$ consists of a syntactic component ρ and a semantic component $\mathfrak{a}$. This means we can explicitly describe strategy contexts in the language of ATLES, whereas in ATL_{sc} we have to make use of $\langle \cdot A \cdot \rangle$ and $\cdot\rangle A \langle\cdot$ that describe that strategies for A are either pushed into the context or released from it. Notice how using strategy commitments in the syntax is more flexible than the strategy context model as every path quantifier in ATLES can be parameterised with a different commitment function, which describes explicitly which agent is using what strategy. In particular, this does not require a dedicated release operator.

The notion of *irrevocable strategies* is captured in ATL_{sc} by carefully avoiding quantification over strategies of committed agents. In ATLES, irrevocability can be made explicit in the syntax.

Once a strategy in the strategy context is overwritten with a new strategy or released, it cannot be recovered in ATL_{sc}, because any reference to it is lost. This could be described with the notion of *forgetting forever.* Not so in ATLES, where 'forgetting forever' can be modelled explicitly in the language, but it is no restriction of the logic as in ATL_{sc}. In fact, an agent in ATLES may *resume* a commitment after releasing it, which also captures a notion of agents having a *strategy memory.*

A strength of ATL_{sc} is to push *any* strategy that is available to an agent into the context. This is achieved with formulas of the form $\neg\langle\cdot A\cdot\rangle\psi$, where the agents in A quantify universally over their strategies F_A. In the semantics, before we continue with the evaluation of the path formula ψ, the strategies F_A are used to upgrade the strategy context (cf. Def. 3). This is another crucial difference to ATLES, which is restricted to existential quantification over commitments. To make more precise the relationship between ATL_{sc} and ATLES, we present an equivalence preserving mapping from a fragment of ATL_{sc} into ATLES. The fragment under consideration is represented by the set of ATL_{sc}-formulas where (i) a negated path subformula can only have the form $\neg(\top\,\mathcal{U}\,\varphi)$, and (ii) every $\langle\cdot A\cdot\rangle$ (for any A) is under the scope of an even number of negations. Let us denote $L(e)$ this language. Let us also denote $L(o)$ the language satisfying (i) but such that every $\langle\cdot A\cdot\rangle$ (for any A) is under the scope of an odd number of negations. We define the translation $tr(\cdot,\cdot)$ as follows:

$$\begin{aligned}
tr(p,\xi) &\overset{\text{def}}{=} p;\\
tr(\neg\varphi_o,\xi) &\overset{\text{def}}{=} \neg tr(\varphi_o,\xi);\\
tr(\varphi_1\vee\varphi_2,\xi) &\overset{\text{def}}{=} tr(\varphi_1,\xi)\vee tr(\varphi_2,\xi);\\
tr({\cdot\rangle}A{\langle\cdot}\varphi,\xi) &\overset{\text{def}}{=} tr(\varphi,\chi),\ \text{ where } \chi=\xi|_{\Sigma\setminus A};\\
tr(\langle\cdot A\cdot\rangle\bigcirc\varphi,\xi) &\overset{\text{def}}{=} \langle\!\langle A\rangle\!\rangle_\rho\bigcirc tr(\varphi,\rho);\\
tr(\langle\cdot A\cdot\rangle\Box\varphi,\xi) &\overset{\text{def}}{=} \langle\!\langle A\rangle\!\rangle_\rho\Box\, tr(\varphi,\rho);\\
tr(\langle\cdot A\cdot\rangle(\varphi_1\,\mathcal{U}\,\varphi_2),\xi) &\overset{\text{def}}{=} \langle\!\langle A\rangle\!\rangle_\rho(tr(\varphi_1,\rho)\,\mathcal{U}\,tr(\varphi_2,\rho)),
\end{aligned}$$

where φ_o is in $L(o)$, φ, φ_1 and φ_2 are in $L(e)$, and where the commitment function ρ overwrites/updates ξ at A with fresh strategy terms. Formally,

$$\rho=\xi|_{\mathsf{dom}(\xi)\setminus A}\cup\{a\mapsto\sigma_a \mid a\in A,\sigma_a \text{ is fresh}\}.$$

The following proposition states that $tr(\cdot,\cdot)$ is indeed equivalence preserving. The proof works by induction on the structure of ATL_{sc}-formulas that are translated.

Proposition 2. *Let φ be a formula in $L(e)$, $\mathcal{C}$ a CGS, x a world in $\mathcal{C}$ and F a strategy in $\mathcal{C}$. The following are equivalent:*

(a) $\mathcal{C}, x \models_F \varphi$*;*
(b) $\mathcal{C}, x \models^{\mathfrak{a}} tr(\varphi, \rho_F)$*, for some* $\langle \rho_F, F\rangle$*-compatible assignment* $\mathfrak{a}$*,*

where $\rho_F = \{a \mapsto \sigma_a \mid f_a \in F, \sigma_a \textit{ is fresh}\}$ *and an assignment* $\mathfrak{a}$ *is* $\langle \rho_F, F\rangle$-compatible *if* $\mathfrak{a}(\rho_F(a)) = f_a$*, for every* $a \in \mathsf{dom}(\rho_F)$ *and* $f_a \in F$*.*

The satisfiability checking problem for $L(e)$ can be solved in ExpTime by Proposition 2 and the fact that ATLES is in ExpTime [19]. This is in contrast with the complexity of full ATL_{sc}, which we establish in the following section.

4 Complexity

This section is devoted to investigating the computational complexity of ATL_{sc} and ATL^*_{sc}.

Generally, high expressiveness tends to come with the price of high computational complexity of reasoning problems. While the model checking problem was already considered in [6,4] (and shown to be between 2ExpTime-hard and non-elementary for ATL_{sc}, while it is 2ExpTime-complete for ATL^*, cf. [2]), we focus here on the satisfiability problem. The lower complexity bounds carry over to ATL_{sc} and ATL^*_{sc} from their respective fragments ATL and ATL^*. It turns out, however, that extending ATL with strategy contexts comes with a much higher price. In the following we show that ATL_{sc} is undecidable. In fact, we show this for the release-free fragment of ATL_{sc}. We use a reduction of the satisfiability problem for the product logic $\mathsf{S5}^n$, which is known to be undecidable. As we have hinted upon in the introduction, ATL_{sc} can capture some type of STIT actual group agency. Thus the undecidability of ATL_{sc} may not come as a surprise considering the undecidability of Chellas' STIT logic of group agency ([10]).

4.1 Product Logic S5

The language of $\mathsf{S5}^n$ is the basic propositional n-modal language given by the following grammar, where p ranges over $\mathbf{\Pi}$, and $i \in \{1, \ldots, n\}$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Diamond_i \varphi.$$

The semantic structures for $\mathsf{S5}^n$ are as follows. A *universal product S5*^{n}*-frame* is a tuple $\mathfrak{F} = (W_1 \times \cdots \times W_n, R_1, \ldots, R_n)$, where for every $i \in \{1, \ldots, n\}$, W_i is a non-empty set of worlds and R_i is the universal relation on W_i. As the relations R_i are determined by the sets W_i, we also denote such frames by $(W_1 \times \cdots \times W_n)$. An S5^n-model is a pair $\mathfrak{M} = (\mathfrak{F}, V)$, where $\mathfrak{F} = (W_1 \times \cdots \times W_n)$ is a universal product S5^n-frame and V a valuation in $\mathfrak{F}$ mapping every propositional variable p to a subset $V(p)$ of $W_1 \times \cdots \times W_n$. The consequence relation $\models_{\mathrm{S5}^n}$ is defined inductively between S5^n-models $\mathfrak{M}$, worlds $\boldsymbol{x} = \langle x_1, \ldots, x_n\rangle$ in $\mathfrak{M}$ and formulas of S5^n as follows:

– $(\mathfrak{M}, \boldsymbol{x}) \models p$ iff $\boldsymbol{x} \in V(p)$;
– $(\mathfrak{M}, \boldsymbol{x}) \models \neg\varphi$ iff $(\mathfrak{M}, \boldsymbol{x}) \not\models_{\mathrm{S5}^n} \varphi$;

– $(\mathfrak{M}, \boldsymbol{x}) \models \varphi \vee \psi$ iff $(\mathfrak{M}, \boldsymbol{x}) \models_{S5^n} \varphi$ or $(\mathfrak{M}, \boldsymbol{x}) \models_{S5^n} \psi$;
– $(\mathfrak{M}, \boldsymbol{x}) \models \Diamond_i \varphi$ iff there is a $y_i \in W_i$ such that $(\mathfrak{M}, \boldsymbol{y}) \models \varphi$, where $\boldsymbol{y} = \langle x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n \rangle$.

We make use of the following results.

Theorem 1. *The satisfiability problem for* $\mathsf{S5}^n$ *over finite models is*

(i) *NExpTime-complete for* $n = 2$*; and*
(ii) *undecidable for all* $n \geq 3$*.*

As S5^2 has the finite model property ([17]), (i) follows from Marx's result on the NExpTime-hardness of S5^2 ([16]). Undecidability of $\mathsf{S5}^n$, for $n \geq 3$, over arbitrary models was shown by Maddux ([15]) in an algebraic setting, via a reduction of the undecidable word problem of semigroups. As the word problem of all finite semigroups is also undecidable ([8]), Maddux's original proof actually shows the undecidability of S5^n reasoning restricted to finite models (even though S5^n lacks the finite model property for $n \geq 3$, cf. [13]). Another way of showing the undecidability of finite model reasoning with S5^n, for $n = 3$, is using Trakhtenbrot's theorem ([3, Section 2.1.2]). He showed how to encode the $\omega \times \omega$ grid and halting Turing machines in finite models, using the first-order language having binary predicates and 3 variables only. This language can be translated to S5^3 while keeping the models finite, using the Halmos-Johnson technique ([9,12], see also [7, Section 8.1]).[1]

4.2 Satisfiability of ATL_{sc}

Theorem 2. *The satisfiability problem for* ATL_{sc} *is*

(i) *NP-hard for formulas with* $n = 1$ *agent;*
(ii) *NExpTime-hard for formulas with* $n = 2$ *agents; and*
(iii) *undecidable for formuals with* $n \geq 3$ *agents.*

We show the lower complexity bounds in Theorem 2 by a reduction of the satisfiability problem for $\mathsf{S5}^n$ to the problem for ATL_{sc}. We leave the matching upper bounds for (i) and (ii) as open problems. Define inductively a translation $tr(\cdot)$ mapping S5^n-formulas to formulas of ATL_{sc} as follows:

$$\begin{aligned} tr(p) &\stackrel{\text{def}}{=} \langle\cdot\emptyset\cdot\rangle \bigcirc p; \\ tr(\neg\varphi) &\stackrel{\text{def}}{=} \neg tr(\varphi); \\ tr(\varphi \vee \psi) &\stackrel{\text{def}}{=} tr(\varphi) \vee tr(\psi); \\ tr(\Diamond_i \varphi) &\stackrel{\text{def}}{=} \langle\cdot i \cdot\rangle (\bot \,\mathcal{U}\, tr(\varphi)). \end{aligned}$$

Notice that the translation does not make use of the strategy release operator $\cdot\rangle A \langle\cdot$ of ATL_{sc}. Thus Theorem 2 holds already for the $\cdot\rangle A \langle\cdot$-free fragment of ATL_{sc}.

[1] We are grateful to Agi Kurucz for referencing and summarising these details for us.

Lemma 1. *Let φ be an $\mathsf{S5}^n$-formula and let Σ_φ be the set of agents that occur in φ. The following are equivalent:*

(i) φ is satisfiable wrt. $\mathsf{S5}^n$ in a finite model;
(ii) $\langle\!\cdot\Sigma_\varphi\cdot\!\rangle \bot\,\mathcal{U}\, tr(\varphi)$ is satisfiable wrt. ATL_{sc}.

Proof. "$(i) \Rightarrow (ii)$": Given a finite $\mathsf{S5}^n$-model $\mathfrak{M} = (\mathfrak{F}, V)$ with $\mathfrak{F} = (W_1 \times \cdots \times W_n)$, we construct a CGS $\mathcal{C}_\mathfrak{M} = \langle W_\mathfrak{M}, V_\mathfrak{M}, \Sigma_\mathfrak{M}, M_\mathfrak{M}, Mov_\mathfrak{M}, E_\mathfrak{M}\rangle$ as follows. Set:

- $W_\mathfrak{M} = W_1 \times \cdots \times W_n$;
- $V_\mathfrak{M}(\boldsymbol{w}) = \{p \mid \boldsymbol{w} \in V(p)\}$, for all $\boldsymbol{w} \in W_\mathfrak{M}$;
- $\Sigma_\mathfrak{M} = \{1, \ldots, n\}$;
- $Mov_\mathfrak{M}(\boldsymbol{w}, i) = \{\{\langle x_1, \ldots, x_n\rangle \mid x_j \in W_j \text{ for all } j \neq i\} \mid x_i \in W_i\}$, for all $\boldsymbol{w} \in W_\mathfrak{M}$ and all $i \in \Sigma_\mathfrak{M}$;
- $M_\mathfrak{M} = \bigcup_{i=1,\ldots,n} Mov_\mathfrak{M}(\boldsymbol{w}, i)$, for some arbitrary $\boldsymbol{w} \in W_\mathfrak{M}$; and
- $E_\mathfrak{M}(\boldsymbol{w}, \boldsymbol{m}) \in \bigcap_{i=1,\ldots,n} m_i$, for all $\boldsymbol{w} \in W_\mathfrak{M}$ and all $\boldsymbol{m} = \langle m_1, \ldots, m_n\rangle \in \mathsf{prof}(\boldsymbol{w})$,

where $\mathsf{prof}(\boldsymbol{w}) = \{\langle m_1, \ldots, m_n\rangle \mid m_i \in Mov_\mathfrak{M}(\boldsymbol{w}, i)\}$. It is readily checked that $\mathcal{C}_\mathfrak{M}$ is indeed a CGS. To see this, note that for all $\boldsymbol{m} = \langle m_1, \ldots, m_n\rangle \in \mathsf{prof}(w)$, each m_i is a subset of $W_\mathfrak{M}$, and verify that the intersection $\bigcap_{i=1,\ldots,n} m_i$ is a singleton set.

Given a world $\boldsymbol{x} = \langle x_1, \ldots, x_n\rangle$ in $\mathfrak{M}$, set the strategy $F_{\boldsymbol{x}} = \{f_1^{\boldsymbol{x}}, \ldots, f_n^{\boldsymbol{x}}\}$ for the agents in $\Sigma_\mathfrak{M}$ as follows: For all $i = 1, \ldots, n$ and all $\boldsymbol{w} \in W_\mathfrak{M}$,

$$f_i^{\boldsymbol{x}}(\boldsymbol{w}) = \{\langle y_1, \ldots, y_n\rangle \mid y_i = x_i, y_j \in W_j \text{ for all } j \neq i\}.$$

$F_{\boldsymbol{x}}$ is indeed a strategy for $\Sigma_\mathfrak{M}$ as $f_i^{\boldsymbol{x}}(\boldsymbol{w}) \in Mov(\boldsymbol{w}, i)$, for all $i = 1, \ldots, n$ and $\boldsymbol{w} \in W_\mathfrak{M}$. Note that $F_{\boldsymbol{x}}$ specifies the same complete move profile $\langle f_1^{\boldsymbol{x}}(\boldsymbol{w}), \ldots, f_n^{\boldsymbol{x}}(\boldsymbol{w})\rangle$ at every state $\boldsymbol{w}$ in $\mathcal{C}_\mathfrak{M}$.

To show this direction of the lemma, it is sufficient to show that, for all $\mathsf{S5}^n$-formulas φ, all $\mathsf{S5}^n$-models and worlds $\boldsymbol{x}$ in $\mathfrak{M}$ and states $\boldsymbol{w}$ in $\mathcal{C}_\mathfrak{M}$:

$$\mathfrak{M}, \boldsymbol{x} \models_{\mathsf{S5}^n} \varphi \text{ iff } \mathcal{C}_\mathfrak{M}, \boldsymbol{w} \models_{F_{\boldsymbol{x}}} tr(\varphi). \tag{1}$$

$F_{\boldsymbol{x}}$ specifies the same state $E_\mathfrak{M}(\boldsymbol{w}, F_{\boldsymbol{x}}) = \boldsymbol{x}$ as successor of any state $\boldsymbol{w}$ in $\mathcal{C}_\mathfrak{M}$. It follows that the set $\mathsf{out}(\boldsymbol{w}, F_{\boldsymbol{x}})$ consists of exactly one play λ such that $\lambda[i] = \boldsymbol{x}$, for all positions $i \geq 0$. Together with the right-hand side of (1), this implies that $\mathcal{C}_\mathfrak{M}, \boldsymbol{w} \models_S \langle\!\cdot\Sigma_\varphi\cdot\!\rangle \Box tr(\varphi)$ and $\mathcal{C}_\mathfrak{M}, \boldsymbol{w} \models_S \langle\!\cdot\Sigma_\varphi\cdot\!\rangle \bot\,\mathcal{U}\, tr(\varphi)$, for any strategy S for $\Sigma_\mathfrak{M}$. Hence, the left-to-right direction of the lemma follows.

To show (1), we proceed by induction on the structure of φ. In the induction base, φ is a proposition p. The following equivalences hold: $\mathfrak{M}, \boldsymbol{x} \models_{\mathsf{S5}^n} p$ iff $\boldsymbol{x} \in V(p)$ iff $p \in V_\mathfrak{M}(\boldsymbol{x})$ iff, for every state $\boldsymbol{w}$ in $\mathcal{C}_\mathfrak{M}$, $\mathcal{C}_\mathfrak{M}, \boldsymbol{w} \models_{F_{\boldsymbol{x}}} \langle\!\cdot\emptyset\cdot\!\rangle \bigcirc p$. For the induction step, assume that we have already shown the induction hypothesis for φ. Consider the following case of the induction step (we omit the Boolean cases):

- $\varphi = \Diamond_i \psi$. Then: $\mathfrak{M}, \boldsymbol{x} \models_{\mathsf{S5}^n} \Diamond_i \psi$ iff there is a $y_i \in W_i$ such that $\mathfrak{M}, \boldsymbol{x'} \models_{\mathsf{S5}^n} \psi$, where $\boldsymbol{x'} = \langle x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n\rangle$. By the induction hypothesis, this is equivalent to $\mathcal{C}_{\mathfrak{M}}, \boldsymbol{w} \models_{F_{\boldsymbol{x}'}} tr(\psi)$ (for all $\boldsymbol{w}$). The strategy $f_i^{\boldsymbol{x}'}(\boldsymbol{w})$ is a move in $Mov_{\mathfrak{M}}(\boldsymbol{w}, a)$ available to agent i at any state $\boldsymbol{w}$ in $W_{\mathfrak{M}}$. Since $F_{\boldsymbol{x}}$ and $F_{\boldsymbol{x}'}$ differ at most in their i-th component, we have that $\mathcal{C}_{\mathfrak{M}}, \boldsymbol{w} \models_{F_{\boldsymbol{x}}} \langle\!\cdot i \cdot\!\rangle \bot \,\mathcal{U}\, tr(\psi)$, which is equivalent to $\mathcal{C}_{\mathfrak{M}}, \boldsymbol{w} \models_{F_{\boldsymbol{x}}} tr(\Diamond_i \psi)$.

This finishes the induction and, thus, this direction of the proof.

"$(ii) \Rightarrow (i)$": Given a CGS $\mathcal{C} = \langle W, V, \Sigma, M, Mov, E\rangle$ for $\langle \Sigma, \Pi\rangle$ with $\Sigma = \{1, \ldots, n\}$ and a world x in $\mathcal{C}$, construct an $\mathsf{S5}^n$-model $\mathfrak{M}_{(\mathcal{C},x)} = (\mathfrak{F}_{(\mathcal{C},x)}, V_{(\mathcal{C},x)})$ with $\mathfrak{F}_{(\mathcal{C},x)} = (W_1^{(\mathcal{C},x)} \times \cdots \times W_n^{(\mathcal{C},x)})$ as follows. Set:

- $W_i^{(\mathcal{C},x)} = Mov(x, i)$ for all $i \in \Sigma$; and
- $V_{(\mathcal{C},x)}(p) = \{\boldsymbol{m} \in W_1^{(\mathcal{C},x)} \times \cdots \times W_n^{(\mathcal{C},x)} \mid p \in V(E(x, \boldsymbol{m}))\}$, for all $p \in \Pi$.

It is readily checked that $\mathfrak{M}_{(\mathcal{C},x)}$ is indeed a finite $\mathsf{S5}^n$-model. While a move profile determines a unique successor $E(x, \boldsymbol{m})$ at a state x, two move profiles $\boldsymbol{m}_1 \neq \boldsymbol{m}_2$ may be mapped to the same successor, i.e. $E(x, \boldsymbol{m}_1) = E(x, \boldsymbol{m}_2)$. However, in the product model $\mathfrak{M}_{(\mathcal{C},x)}$ the move profiles $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ are different worlds. Let $F_\Sigma = \{f_1, \ldots, f_n\}$ be a strategy in $\mathcal{C}$ for the agents in Σ. A world $\boldsymbol{m}$ in $\mathfrak{M}_{(\mathcal{C},x)}$ is called an $F_{\Sigma,x}$*-world* if $E(x, \boldsymbol{m}) = \lambda[1]$ with $\{\lambda\} = \mathsf{out}(x, F_{\Sigma,x})$.

To show this direction of the lemma, it is sufficient to show that, for all S5^n-formulas φ, for all CGSs $\mathcal{C}$, all worlds x in $\mathcal{C}$ and all strategies F for Σ, and all F-worlds $\boldsymbol{w}_F$ in $\mathfrak{M}_{(\mathcal{C},x)}$:

$$\mathcal{C}, x \models_F tr(\varphi) \text{ iff } \mathfrak{M}_{(\mathcal{C},x)}, \boldsymbol{w}_F \models_{\mathsf{S5}^n} \varphi. \tag{2}$$

Then, the right-to-left direction of the lemma follows from (2) together with the fact that $\mathcal{C}, x \models_S \langle\!\cdot \Sigma_\varphi \cdot\!\rangle tr(\varphi)$ implies $\mathcal{C}, x \models_F tr(\varphi)$, for any strategy S.

To show (2), we proceed by induction on the structure of φ. In the induction base, φ is a proposition p. The following equivalences hold: $\mathcal{C}, x \models_F tr(p)$ iff $\mathcal{C}, x \models_F \langle\!\cdot \emptyset \cdot\!\rangle \bigcirc p$ iff $\mathcal{C}, y \models_F p$, where $y = \mathsf{out}(x, F)$ iff $p \in V(y)$ iff $\boldsymbol{w}_F \in V_{(\mathcal{C},x)}(p)$ iff $\mathfrak{M}_{(\mathcal{C},x)}, \boldsymbol{w}_F \models_{\mathsf{S5}^n} p$. For the induction step, assume that we have already shown the induction hypothesis for φ. Again, we skip the Boolean cases and proceed with the interesting case:

- $\varphi = \Diamond_i \psi$. Then: $\mathcal{C}, x \models_F tr(\Diamond_i \psi)$ iff $\mathcal{C}, x \models_F \langle\!\cdot i \cdot\!\rangle \bot \,\mathcal{U}\, tr(\psi)$ iff there is a strategy f_i such that it holds that $\mathcal{C}, x \models_S tr(\psi)$, with $S = \{f_i\} \cup \{f_b \in F \mid b \neq i\}$. By the induction hypothesis, we obtain $\mathfrak{M}_{(\mathcal{C},x)}, \boldsymbol{w}_S \models_{\mathsf{S5}^n} \psi$. Since F and S are identical with the possible exception of the strategy f_i for agent i, the worlds $\boldsymbol{w}_S$ and $\boldsymbol{w}_F$ differ at most in their i-th component. We have that $\mathfrak{M}_{(\mathcal{C},x)}, \boldsymbol{w}_F \models_{\mathsf{S5}^n} \Diamond_i \psi$. The other direction of this case can be shown similarly.

□

Corollary 1. *The satisfiability problem of any variant of* ATL *with strategy contexts in [4] is undecidable.*

Acknowledgements. We thank Agi Kurucz for her input on the undecidability of finite model reasoning of the product logic $\mathsf{S5}^n$. We are indebted to the reviewers of LAMAS 2012 and the workshop "Modeling Strategic Reasoning" at the Lorentz Center in Leiden, The Netherlands, 20–24 February 2012. The first author is supported by a Marie Curie COFUND fellowship, and the second author by a Juan de la Cierva fellowship of Spain and the project Agreement Technologies (Grant CONSOLIDER CSD2007-0022, INGENIO 2010), and the MICINN projects TIN2006-15455 and TIN2009-14562-CO5.

References

1. Ågotnes, T., Goranko, V., Jamroga, W.: Alternating-time temporal logics with irrevocable strategies. In: Proceedings of TARK 2007, pp. 15–24. ACM (2007)
2. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. The Journal of the ACM 49(5), 672–713 (2002)
3. Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. In: Perspectives Mathematical Logic. Springer (1997)
4. Brihaye, T., Da Costa, A., Laroussinie, F., Markey, N.: ATL with Strategy Contexts and Bounded Memory. In: Artemov, S., Nerode, A. (eds.) LFCS 2009. LNCS, vol. 5407, pp. 92–106. Springer, Heidelberg (2008)
5. Broersen, J., Herzig, A., Troquard, N.: A STIT-Extension of ATL. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) JELIA 2006. LNCS (LNAI), vol. 4160, pp. 69–81. Springer, Heidelberg (2006)
6. Da Costa, A., Laroussinie, F., Markey, N.: ATL with strategy contexts: Expressiveness and model checking. In: Proceedings of FSTTCS 2010. LIPIcs, vol. 8, pp. 120–132. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik (2010)
7. Gabbay, D., Kurucz, A., Wolter, F., Zakharyaschev, M.: Many-dimensional modal logics: theory and applications. Studies in Logic, vol. 148. Elsevier Science (2003)
8. Gurevich, Y., Lewis, H.R.: The word problem for cancellation semigroups with zero. The Journal of Symbolic Logic 49(1), 184–191 (1984)
9. Halmos, P.: Algebraic logic, IV. Transactions of the AMS 86, 1–27 (1957)
10. Herzig, A., Schwarzentruber, F.: Properties of logics of individual and group agency. In: Proceedings of AiML 2008, pp. 133–149. College Publications (2008)
11. Horty, J.F.: Agency and Deontic Logic. Oxford University Press, Oxford (2001)
12. Johnson, J.: Nonfinitizability of classes of representable polyadic algebras. The Journal of Symbolic Logic 34(3), 344–352 (1969)
13. Kurucz, A.: S5 x S5 x S5 lacks the finite model property. In: Proceedings of AiML 2002, pp. 321–327. World Scientific (2002)
14. Laroussinie, F., Markey, N., Oreiby, G.: On the expressiveness and complexity of ATL. Logical Methods in Computer Science 4(2) (2008)
15. Maddux, R.: The equational theory of CA_3 is undecidable. The Journal of Symbolic Logic 45(2), 311–316 (1980)
16. Marx, M.: Complexity of products of modal logics. Journal of Logic and Computation 9(2), 197–214 (1999)
17. Mortimer, M.: On languages with two variables. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 21, 135–140 (1975)
18. Troquard, N., Walther, D.: ATL with contexts: agency and explicit strategies. In: Proceedings of LAMAS@AAMAS 2012 (2012)
19. Walther, D., van der Hoek, W., Wooldridge, M.: Alternating-time temporal logic with explicit strategies. In: Proceedings of TARK 2007, pp. 269–278. ACM (2007)

Jumping to Conclusions
A Logico-Probabilistic Foundation for Defeasible Rule-Based Arguments

Bart Verheij

Artificial Intelligence, University of Groningen

Abstract. A theory of defeasible arguments is proposed that combines logical and probabilistic properties. This logico-probabilistic argumentation theory builds on two foundational theories of nonmonotonic reasoning and uncertainty: the study of nonmonotonic consequence relations (and the associated minimal model semantics) and probability theory. A key result is that, in the theory, qualitatively defined argument validity can be derived from a quantitative interpretation. The theory provides a synthetic perspective of arguments 'jumping to conclusions', rules with exceptions, and probabilities.

1 Introduction

Jumping to conclusions is a necessary and oft-used skill. We hear a voice on the phone, and conclude it's our father's. We smell foul coffee, and conclude it's from that dreaded machine down the hall. We find a note on the kitchen table, and conclude that our son has gone out. But sometimes we jump too far. It's not our father, but his brother. It's not coffee from *that* machine, but from a similar one on the next floor. And we find our son in his room at home, playing his favorite computer game, as his message was yesterday's.

In this paper, a mathematical theory of jumping to conclusions is developed. The theory's starting point is that jumping to conclusions is allowed ('valid', using a heavily laden term) when the conclusions do not lead us too far from the premises. Or, to be a bit more precise, when *the case made* on the basis of the premises, is close to those premises. Here 'the case made' is defined as the conjunction of premises and conclusions. In other words, we can jump to certain conclusions, if adding them to the premises is not a jump too far.

For instance, when a witness says she saw the suspect at the crime scene (w), we 'jump to' the conclusion that the suspect was indeed there (s) by making the case $w \wedge s$. A valid jump will be written $\varphi \mid\!\sim \psi$, an invalid jump as $\varphi \mid\!\not\sim \psi$. That the corresponding case made is sufficiently close to the premises is written as $\varphi \sim \varphi \wedge \psi$, that it is too far a jump as $\varphi \succ \varphi \wedge \psi$.

Our theory formalizes *ampliative reasoning*, as it has been called by Peirce, that is: reasoning that goes beyond the premises. Toulmin used the term *substantial reasoning* for the same concept, and considered reasoning only interesting when it adds information to what is already in the premises.

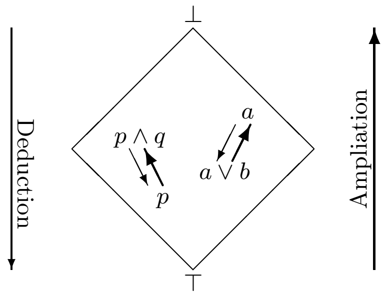


Fig. 1. Deduction and ampliation in a Boolean lattice

In Section 2, ampliative inference is studied in relation to nonmonotonic inference. In Section 3, a connection is made to the mathematical (pre)order relations, as a first step towards the quantitative interpretation, given in Section 4. It is shown that well-behaved ampliative inference can be derived from an 'argument value' function on the language with properties close to those of a standard probability function. In Section 5, a connection to argumentation theory is made by considering arguments with local structure in terms of premises, rules, and exceptions. It is shown that the global validity of an ampliative argument can be determined by the application of non-excluded rules.

2 Ampliative Arguments

The propositions that occur in arguments are expressed in a classical language L with BNF specification $\varphi ::= \top \mid \perp \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \leftrightarrow \psi$, and the associated classical, deductive, monotonic consequence relation, denoted $\vdash$. As in the study of nonmonotonic inference, we write $\varphi \mid\!\sim \psi$ to denote the validity of an ampliative argument from premises φ to conclusions ψ (plural to allow us to speak of separate premises/conclusions that are in an argument joined by conjunction). The proposition $\varphi \wedge \psi$ is the *case made* by the argument.

In the Boolean lattice associated with our language L, ampliative and deductive arguments have opposite directions (Figure 1, with $\top$ at the bottom and $\perp$ on top). In Figure 1, we can jump from p to $p \wedge q$ and from $a \vee b$ to a; written as $p \mid\!\sim p \wedge q$ and $a \vee b \mid\!\sim a$.

The following qualitative properties, well-known in the general study of nonmonotonic inference relations [2,11,12], are our starting point. For all propositions φ, φ', ψ, ψ', $\chi \in L$:

(LE)	If $\varphi \mid\!\sim \psi$, $\vdash \varphi \leftrightarrow \varphi'$ and $\vdash \psi \leftrightarrow \psi'$, then $\varphi' \mid\!\sim \psi'$.
(Ant)	If $\varphi \mid\!\sim \psi$, then $\varphi \mid\!\sim \varphi \wedge \psi$.
(PR)	If $\varphi \mid\!\sim \varphi \wedge \psi$, then $\varphi \mid\!\sim \psi$.
(R)	$\varphi \mid\!\sim \varphi$.
(RW)	If $\varphi \mid\!\sim \psi \wedge \chi$, then $\varphi \mid\!\sim \psi$.
(CCM)	If $\varphi \mid\!\sim \psi \wedge \chi$, then $\varphi \wedge \psi \mid\!\sim \chi$.
(CCT)	If $\varphi \mid\!\sim \psi$ and $\varphi \wedge \psi \mid\!\sim \chi$, then $\varphi \mid\!\sim \psi \wedge \chi$.

(LE), for Logical Equivalence, expresses that in a valid ampliative argument the premises and the conclusions can be replaced by a classical equivalent (in the sense of $\vdash$). (Ant), for Antededence, expresses that when we can jump from certain premises to a conclusion, we can also jump to the case made by the argument (recall: the conjunction of conclusion and premises). Since (Ant) holds for ampliative arguments, every argument $\varphi \mathrel{|\!\sim} \psi$ has an associated 'ampliation', i.e., an argument of which the conclusion deductively implies the premises, namely $\varphi \mathrel{|\!\sim} \varphi \wedge \psi$. (PR), for Premise Reduction, says that we can also jump to a conclusion that classically follows from the case made by a valid argument. As the converse of (Ant), it is technically useful below. (R), for Reflexivity, expresses the validity of the limiting case of jumping from premises to themselves. (RW), for Right Weakening, expresses that when the premises justify a composite conclusion also the intermediate conclusions are justified. It strengthens (PR) (given (LE)). (CCM), for Conjunctive Cautious Monotony, expresses that we can still jump to the case made by a valid argument when an intermediate conclusion is added to the argument's premises. (CCT), for Conjunctive Cumulative Transitivity, is a variation of the Cumulative Transitivity property (CT, also known as Cut) extensively studied in the literature (which has $\varphi \mathrel{|\!\sim} \chi$ instead of $\varphi \mathrel{|\!\sim} \psi \wedge \chi$ as a consequent). The variation may seem minor, but is essential in the absence of the (And) property (If $\varphi \mathrel{|\!\sim} \psi$ and $\varphi \mathrel{|\!\sim} \chi$, then $\varphi \mathrel{|\!\sim} \psi \wedge \chi$). Assuming (Ant), (CCT) expresses the validity of chaining jumps from φ via $\varphi \wedge \psi$ to $\varphi \wedge \psi \wedge \chi$.

The relation $\sim$ associated with $\mathrel{|\!\sim}$ singles out those arguments that have the case made by the argument as conclusion, i.e., have a conclusion that logically implies the premises.

Definition 1. *For $\mathrel{|\!\sim} \subseteq L \times L$, we define:*

$\varphi \sim \psi := \psi \vdash \varphi$ *and* $\varphi \mathrel{|\!\sim} \psi$.

We now show that the properties of $\mathrel{|\!\sim}$ have close counterparts in terms of $\sim$. Beware: notwithstanding the suggestive notation, $\sim$ need not be symmetric.

(LEAmpl)	If $\varphi \sim \psi$, $\vdash \varphi \leftrightarrow \varphi'$ and $\vdash \psi \leftrightarrow \psi'$, then $\varphi' \sim \psi'$.
(Ampl)	If $\varphi \sim \psi$, then $\psi \vdash \varphi$.
(Eq)	If $\vdash \varphi \leftrightarrow \psi$, then $\varphi \sim \psi$.
(Int)	If $\chi \vdash \psi$, $\psi \vdash \varphi$ and $\varphi \sim \chi$, then $\varphi \sim \psi$ and $\psi \sim \chi$.
(Tr)	If $\varphi \sim \psi$ and $\psi \sim \chi$, then $\varphi \sim \chi$.

(LEAmpl) expresses Logical Equivalence, this time for $\sim$. By (Ampl), for Ampliation, $\sim$ is an ampliation relation: when $\varphi \sim \psi$, ψ goes logically beyond φ. Also, by (Ampl), $\sim$ is not normally symmetric (which would come at the price of reducing $\sim$ to classical equivalence). (Eq), for Equivalence, says that a proposition's ampliations include all propositions logically equivalent to the proposition. (Int), for Interpolation, says that an ampliation can be split at a proposition that is logically in between. (Tr), for Transitivity, says that the ampliation relation is transitive.

The listed properties for $\mathrel{|\!\sim}$ and $\sim$ have close formal connections:

Proposition 1. *Let $\mathrel{|\!\sim}$ be an inference relation obeying (LE), (Ant) and (PR), and $\sim$ the associated ampliation relation. Then the following hold:*

1. *$\sim$ obeys (LEAmpl) and (Ampl).*
2. *$\varphi \mathrel{|\!\sim} \psi$ if and only if $\varphi \sim \varphi \wedge \psi$.*
3. *$\mathrel{|\!\sim}$ obeys (R) if and only if $\sim$ obeys (Eq).*
4. *$\mathrel{|\!\sim}$ obeys (RW) and (CCM) if and only if $\sim$ obeys (Int).*
5. *$\mathrel{|\!\sim}$ obeys (CCT) if and only if $\sim$ obeys (Tr).*

When the conditions (LE), (Ant) and (PR) of the proposition obtain, so that 1–5 follow, we speak of an *ampliative inference* relation $\mathrel{|\!\sim}$ and a corresponding *ampliation* relation $\sim$. When all properties listed obtain, we speak of *qualitative ampliative inference* and *qualitative ampliation*. The inference relation can be recovered from the ampliation relation (part 2 of the proposition).

An example of qualitative ampliative inference is the following, for which $p \mathrel{|\!\sim} q$, but $p \wedge e \not\mathrel{|\!\sim} q$, so e is an exception blocking the jump from p to q:

$\varphi \mathrel{|\!\sim} \psi$ if and only if $(p \wedge q \vdash \varphi \wedge \psi$ and $\varphi \vdash p)$ or $(p \wedge e \vdash \varphi \wedge \psi$ and $\varphi \vdash p \wedge e)$ or $(\phi \vdash \psi)$.

A second example shows that we can sometimes jump to incompatible conclusions, e.g., in the case of two reasonable, but inconsistent decisions (as in a choice situation). Here we can jump from p to either q, or to $\neg q$, but not to their conjunction:

$\varphi \mathrel{|\!\sim} \psi$ if and only if $(p \wedge q \vdash \varphi \wedge \psi$ and $\varphi \vdash p)$ or $(p \wedge \neg q \vdash \varphi \wedge \psi$ and $\varphi \vdash p)$ or $(\phi \vdash \psi)$.

This example of non-qualitative ampliative inference illustrates the role of (CCT) and (Tr):

$\varphi \mathrel{|\!\sim} \psi$ if and only if $(p \wedge q \vdash \varphi \wedge \psi$ and $\varphi \vdash p)$ or $(p \wedge q \wedge r \vdash \varphi \wedge \psi$ and $\varphi \vdash p \wedge q)$ or $(\varphi \vdash \psi)$.

The relation $\mathrel{|\!\sim}$ obeys all properties listed above, but not (CCT)/(Tr) since it is not possible to chain the arguments $p \mathrel{|\!\sim} p \wedge q$ and $p \wedge q \mathrel{|\!\sim} p \wedge q \wedge r$ since $p \not\mathrel{|\!\sim} p \wedge q \wedge r$. Cf. the following figure.

$$\begin{tikzcd}
\boxed{p} \arrow[r] \arrow[rr, bend right, "\times"] & p \wedge q \arrow[r] & p \wedge q \wedge r
\end{tikzcd} \tag{1}$$

3 Ampliation and the Ordering of Propositions

In Section 4, we establish a quantitative interpretation of well-behaved ampliative inference in terms of a numeric 'argument value' function on the language. By this function, the propositions of the language can be ordered, in a way that is similar to a probability function that assigns decreasing probabilities to more specific propositions. As a first step, we consider certain (pre)order relations associated with ampliation. For instance, the ampliation relation itself is a preorder, since Reflexivity ($\varphi \sim \varphi$) holds by (LEAmpl) and (Eq), and Transitivity is part of the definition. Since Antisymmetry obtains for L's logical equivalence classes, $\sim$ is even a partial order on logical equivalence classes. But Symmetry (If $\varphi \sim \psi$, then $\psi \sim \varphi$) fails in general, so $\sim$ is not an equivalence relation.

The following definitions are needed.

Definition 2. *For an ampliative inference relation $\mathrel{|\!\sim}$ and a corresponding ampliation relation $\sim$, we define:*

$\varphi \succ \psi := \psi \vdash \varphi$ *and* $\varphi \nsim \psi$
$\varphi \succsim \psi := \varphi \sim \psi$ *or* $\varphi \succ \psi$
$\varphi \prec \psi := \psi \succ \varphi$
$\varphi \precsim \psi := \psi \succsim \varphi$

As a result, $\varphi \succsim \psi$ is equivalent to $\psi \vdash \varphi$. Also, given $\psi \vdash \varphi$, $\varphi \mathrel{|\!\sim} \psi$ is equivalent to $\varphi \sim \psi$, and $\varphi \not\mathrel{|\!\sim} \psi$ is equivalent to $\varphi \succ \psi$. Furthermore, for all φ and ψ, exactly one of $\varphi \sim \psi$, $\varphi \succ \psi$, and $\psi \nvdash \varphi$ holds.

The following result says that, if we stay within an ampliative chain, $\sim$ and $\succ$ are well-defined 'up to $\sim$', although care is needed by the failure of Symmetry.

Proposition 2. *Let $\mathrel{|\!\sim}$ be a qualitative ampliative inference relation. Assume $\chi \vdash \psi \vdash \varphi$.*

1. *When $\varphi \sim \psi$, the following hold:*
   (a) $\psi \sim \chi$ if and only if $\varphi \sim \chi$.
   (b) $\psi \succ \chi$ if and only if $\varphi \succ \chi$.
2. *When $\psi \sim \chi$, the following hold:*
   (a) $\varphi \sim \psi$ if and only if $\varphi \sim \chi$.
   (b) $\varphi \succ \psi$ if and only if $\varphi \succ \chi$.

If $\sim$ and $\succ$ are to be represented by numeric values, their properties should be sufficiently well-behaved *over the whole language*, and not just within ampliative chains. This requires a further property that expresses a kind of conservativeness across ampliation chains. Consider a case in which $\varphi \mathrel{|\!\sim} \psi$ and $\psi \mathrel{|\!\sim} \chi$ (cf. the following picture). The arrows indicate ampliations (in the upward direction of the arrow), the dotted lines deductions (in the downward direction).

$$\begin{tikzcd}
& \varphi \wedge \psi \arrow[dr, dash, dotted, "\precsim"'] & & \psi \wedge \chi \arrow[dr, dash, dotted, "\precsim"'] & \\
\varphi \arrow[ur, "\sim"'] & & \psi \arrow[ur, "\sim"'] & & \chi
\end{tikzcd} \tag{2}$$

The two ampliations occur in (possibly) different chains, one containing φ and $\varphi \wedge \psi$, the other ψ and $\psi \wedge \chi$. When the relations are numerically derived, one would expect that they can be chained. For instance, $\varphi \sim \varphi \wedge \psi$ and $\varphi \wedge \psi \precsim \psi$ would give φ "$\precsim$" ψ, and similarly ψ "$\precsim$" χ, hence φ "$\precsim$" χ.

Now consider what happens if also $\chi \mathrel{|\!\sim} \varphi$ (as in the figure below).

$$\begin{tikzcd}
& \varphi\wedge\psi \arrow[dr, dash, "\precsim"] & & \psi\wedge\chi \arrow[dr, dash, "\precsim"] & & \chi\wedge\varphi \arrow[dr, dash, "\precsim"] & & \\
\varphi \arrow[ur, "\sim"'] & & \psi \arrow[ur, "\sim"'] & & \chi \arrow[ur, "\sim"'] & & \varphi & \cdots
\end{tikzcd} \tag{3}$$

In this situation, we conclude φ "$\precsim$" $\chi\wedge\varphi$. Since also $\chi\wedge\varphi$ "$\precsim$" φ (by $\chi\wedge\varphi \precsim \varphi$, i.e., $\chi\wedge\varphi \vdash \varphi$), we conclude φ "$\sim$" $\chi\wedge\varphi$. But φ and $\chi\wedge\varphi$ are in the same chain, so we could have $\varphi \not\sim \chi\wedge\varphi$, in disagreement with φ "$\sim$" $\chi\wedge\varphi$. What we need is that a sequence of $\sim$ and $\precsim$ statements as in the figure is *conservative* on ampliation chains. All this can be made precise using the following definition.

Definition 3. *For an ampliative inference relation $\mathrel{|\!\sim}$ and a corresponding ampliation relation $\sim$, we define:*

$\precsim^*$ *is the transitive closure of* $\mathrel{|\!\sim}$
$\varphi \sim^* \psi := \varphi \precsim^* \psi$ *and* $\psi \precsim^* \varphi$
$\varphi \prec^* \psi := \varphi \precsim^* \psi$ *and not* $\psi \precsim^* \varphi$
$\varphi \succsim^* \psi := \psi \precsim^* \varphi$
$\varphi \succ^* \psi := \psi \prec^* \varphi$

$\precsim^*$ extends $\precsim$ and $\sim^*$ extends $\sim$, but $\prec^*$ does not always extend $\prec$. (Cf. the counterexample to Conservativeness below.) By (R), $\precsim^*$ is reflexive, hence $\sim^*$ is also reflexive. As $\precsim^*$ is transitive, it is a preorder, and $\sim^*$ an equivalence relation. $\precsim^*$ is well-defined on $\sim^*$-equivalence classes:

Proposition 3. *Let $\mathrel{|\!\sim}$ be a qualitative ampliative inference relation. Then the following holds:*

If $\varphi \precsim^* \psi$, $\varphi \sim^* \varphi'$ and $\psi \sim^* \psi'$, then $\varphi' \precsim^* \psi'$.

Conservativeness is expressed as follows. The $\mathrel{|\!\sim}$ version is known as Loop.

(L) If $\varphi_0 \mathrel{|\!\sim} \varphi_1$, $\varphi_1 \mathrel{|\!\sim} \varphi_2$, ... and $\varphi_n \mathrel{|\!\sim} \varphi_0$, then $\varphi_0 \mathrel{|\!\sim} \varphi_n$.
(C) If $\varphi \precsim^* \psi$ and $\psi \vdash \varphi$, then $\varphi \sim \psi$.

Assuming qualitative ampliative inference, the $\mathrel{|\!\sim}$ and $\sim$ versions are equivalent:

Proposition 4. *Assume an ampliative inference relation $\mathrel{|\!\sim}$ and a corresponding ampliation relation $\sim$. Then $\mathrel{|\!\sim}$ obeys (L) if and only if $\sim$ obeys (C).*

Here is an example of a consequence relation for qualitative ampliative arguments that does not obey (L)/(C):

$\varphi \mathrel{|\!\sim} \psi$ if and only if ($p\wedge q \vdash \varphi\wedge\psi$ and $\varphi \vdash p$) or ($q\wedge r \vdash \varphi\wedge\psi$ and $\varphi \vdash q$) or ($p\wedge r \vdash \varphi\wedge\psi$ and $\varphi \vdash r$) or ($\varphi \vdash \psi$).

For this consequence relation, we have $p \mathrel{|\!\sim} q \mathrel{|\!\sim} r \mathrel{|\!\sim} p$, while $p \mathrel{|\!\not\sim} r$.

4 Quantifiable Qualitative Ampliative Arguments

The next step towards a quantified interpretation of qualitative ampliative arguments requires the notion of a numeric order of magnitude of a proposition.

Definition 4. *For a relation $\mid\!\sim\subseteq L \times L$, we define the* order *of φ, notation $O(\varphi)$, as the maximal length n of a sequence $\varphi_0, \ldots, \varphi_n$ with $\varphi = \varphi_0$ and, for all $i \in \{0, \ldots, n-1\}$, $\varphi_i \succ^* \varphi_{i+1}$ (if such a finite maximal length exists). When every proposition has a finite order, we say that $\mid\!\sim$* has finite orders. *When also there is a maximum order, we say that $\mid\!\sim$* has bounded finite orders.

When the property (L)/(C) holds, there cannot be a $\prec^*$-loop. So, for a language with a finite number of elementary propositions, (L)/(C) implies finite orders.

Restricting to bounded finite orders, qualitative ampliative inference can be defined in terms of the order function.

Theorem 1. *Let $\mid\!\sim$ express qualitative ampliative inference with bounded finite orders. Then there is an integer-valued function $O : L \to \mathbb{N}$ such that*

$\varphi \sim \psi$ *if and only if* $O(\varphi) = O(\psi)$ *and* $\psi \vdash \varphi$
$\varphi \succ \psi$ *if and only if* $O(\varphi) > O(\psi)$ *and* $\psi \vdash \varphi$

for which the following properties hold:

1. $O(\bot) \leq O(\varphi) \leq O(\top)$.
2. $O(\varphi) = 0$ *if and only if* $\varphi \mid\!\sim \bot$.
3. $O(\varphi) \geq max(O(\varphi \wedge \psi), O(\varphi \wedge \neg\psi))$.
4. *If* $\psi \vdash \varphi$*, then* $O(\varphi) \geq O(\psi)$.
5. $\varphi \mid\!\sim \psi$ *if and only if* $O(\varphi) = O(\varphi \wedge \psi)$.
6. $\varphi \not\mid\!\sim \psi$ *if and only if* $O(\varphi) > O(\varphi \wedge \psi)$.

In the proof, $O(\varphi)$ is taken to be the order of φ. The properties in the theorem are already quite close to the properties of probability theory. But the order of a proposition behaves like an order of magnitude, reflected by the max-relation in part 3, in contrast with probability theory's addition (which also has equality instead of $\geq$).

We next show that (assuming a further finitizing restriction) the max-relation can be replaced by addition. The order function is replaced by an 'argument value' function, in such a way that the order function behaves like a kind of rounded logarithm of the value function. We need definitions of maximally specific conclusions and of minimally specific exceptions.

Definition 5. *A* maximal conclusion *(or* extension*) of a proposition φ is a proposition ψ with $\varphi \mid\!\sim \psi$ and $\psi \vdash \varphi$ that is maximally specific in the sense of $\vdash$, i.e., for all χ, if $\psi \not\vdash \chi$ and $\chi \vdash \psi$, then $\varphi \not\mid\!\sim \chi$. When every conclusion of a proposition can be amplified to a maximal conclusion, we say that $\mid\!\sim$ has* finitely expressible maximal conclusions.

A minimal non-conclusion *of a proposition φ is a proposition ψ with $\varphi \not\mid\!\sim \psi$ and $\psi \vdash \varphi$ that is minimally specific in the sense of $\vdash$, i.e., for all χ, if $\psi \vdash \chi$*

and $\chi \nvdash \psi$, *then* $\varphi \mathrel{|\!\sim} \chi$. *When every non-conclusion of a proposition (*$\vdash$*-implied by the non-conclusion) is an amplification of a minimal non-conclusion, we say that* $\mathrel{|\!\sim}$ *has* finitely expressible minimal non-conclusions.

A minimal exception *of a proposition* φ *is a minimal non-conclusion* ψ *of* φ *such that there does not exist a maximal conclusion* χ *of* φ *with* $\psi \vdash \chi$.

To simplify technicalities (involving infinite disjunctions and conjunctions corresponding to infinite unions and intersections of sets), we restrict ourselves to inference relations with finitely expressible maximal conclusions and minimal non-conclusions.

The following theorem contains the inductive definition of an integer-valued function $v : L \to \mathbb{N}$ that characterizes qualitative ampliative inference. The induction is based on the property that minimal non-conclusions of a proposition have an order that is strictly lower than the proposition.

Theorem 2. *Let* $\mathrel{|\!\sim}$ *express qualitative ampliative inference, have bounded finite orders, have finitely expressible maximal conclusions and minimal non-conclusions, have a bounded finite number of maximal conclusions per proposition, and a bounded finite number of minimal exceptions per proposition. Let* C *denote the minimal upper bound of the number of maximal conclusions. Let* O *be the order function on propositions and* $c(\varphi)$ *the number of maximal conclusions of a proposition* φ. *We inductively define the* argument value *function* $v : L \to \mathbb{N}$*:*

$$v(\varphi) := \begin{cases} 0 & \text{if } O(\varphi) = 0; \\ 1 & \text{if } O(\varphi) = 1; \\ c(\varphi).v(O(\varphi)) + \sum\{v(\psi) \mid \psi \text{ minimal exception of } \varphi\} & \text{if } O(\varphi) > 1, \end{cases}$$

where $v(n) = (C^2 + 1)^{n-1}$ *for* $n > 0$. *Then:*

1. $v(\bot) \leq v(\varphi) \leq v(\top)$.
2. $v(\varphi) = 0$ *if and only if* $\varphi \mathrel{|\!\sim} \bot$.
3. $v(\varphi) \geq v(\varphi \wedge \psi) + v(\varphi \wedge \neg\psi)$.
4. *If* $\psi \vdash \varphi$, *then* $v(\varphi) \geq v(\psi)$.
5. $\varphi \mathrel{|\!\sim} \psi$ *if and only if and* $\varphi \mathrel{|\!\sim} \bot$ *or* $\frac{v(\varphi \wedge \psi)}{v(\varphi)} > \frac{1}{C+1}$.
6. $\varphi \mathrel{|\!\not\sim} \psi$ *if and only if* $\varphi \mathrel{|\!\not\sim} \bot$ *and* $\frac{v(\varphi \wedge \psi)}{v(\varphi)} \leq \frac{1}{C+1}$.

Note the special role of $\bot$, related to 0 having a logarithm of minus infinity.

As promised, the integer value function v of the theorem behaves almost like a probability function (bearing the standard connection between propositions and sets in mind). There is still one telling difference: whereas the sum of the probability of disjoint sets is equal to the probability of their union (in accordance with Kolmogorov's standard axioms), the sum of the argument values of mutually inconsistent propositions may not be *equal* to their disjunction, but can also be lower. There is a natural interpretation for this technical difference. Whereas probability theory counts cases of which all properties are available, in our setting of ampliative argumentation, the argument values count cases in which there can be unknown properties. Logically speaking, probability counts worlds (complete interpretations), whereas ampliative argumentation counts states expressed by propositions (partial interpretations). The role of partiality is to be expected, as ampliation is a way of adding information to the partial information expressed by the premises (cf. the examples at the start of the paper). The amplified information will again be partial.

In light of the close analogy between probabilities and the argument values used in the theorem, we will write $v(\psi|\varphi) := v(\varphi \wedge \psi)/v(\varphi)$. The argument value version of Bayes' theorem is an immediate consequence: $v(\psi|\varphi) = v(\varphi|\psi)v(\psi)/v(\varphi)$.

Theorem 2 shows that the 'conditional argument value' $v(\psi|\varphi)$ can be interpreted as the strength of the argument. When the strength is above the threshold $1/(C+1)$, the argument is valid, when the strength is below (or equal to) the threshold it is invalid.

5 Structured Arguments

Until now, by our focus on the inference relation $\mid\!\sim$, we have used the classical model of arguments as unstructured premise-conclusion pairs. As such, we have established a theory of the global validity of arguments. However, in contemporary formal theory of defeasible arguments (see [16] for an overview), arguments have additional structure since they are constructed using premises, rules, exceptions, defeaters, etc. A central problem addressed in this kind of work is how the local structure of an argument, and of the arguments attacking the argument, determines the global validity of the argument. Today, Dung's seminal abstract perspective [5] plays a key role in determining such global validity (or *argumentative warrant*). Dung proposed different kinds of 'semantics' — preferred, complete, grounded and stable —, each determining a different kind of argumentative warrant. Several other semantics have been proposed, e.g., the stage and semi-stable semantics, both in [17] (though the semi-stable with another name), and the ideal and cf2 semantics; cf. the overview [1].

When we restrict ourselves to qualitative ampliative inference, as in this paper, determining the global validity of structured arguments becomes relatively simple: an argument is globally valid if the case it makes is constructed using rules that are not excluded by an exception. This is possible because we do not assume that all sets of rules, exceptions, defeaters, etc. can occur; the input from which arguments are constructed is constrained by the properties of a global theory (expressed in the inference relation). As a result, arguments are only constructed from 'well-behaved' sets rules and exceptions.

Syntactically, structured arguments are sequences of premises (propositions in L), rules of the form $\varphi \Rightarrow \psi$, and exceptions of the form $\neg(\varphi \Rightarrow \psi)$. An exception $\neg(\varphi' \Rightarrow \psi)$ *excludes* a rule $\varphi \Rightarrow \psi$ when $\varphi' \vdash \varphi$. When $\varphi \mid\!\sim \psi$, $\varphi \Rightarrow \psi$ is an $\mid\!\sim$-rule; when $\varphi \not\mid\!\sim \psi$, $\neg(\varphi \Rightarrow \psi)$ is an $\mid\!\sim$-exception. We will assume a language

L with a finite number of elementary propositions and a qualitative ampliative inference relation $\mid\!\sim$.

Definition 6. *Let $\mid\!\sim$ express qualitative ampliative inference. We inductively define valid arguments α from premises $P(\alpha) \in L$ making the case $C(\alpha) \in L$ as follows.*

1. *The empty argument $[\ ]$ is a valid argument from $\top$ making the case $\top$. It has no applicable rules and no applicable exceptions.*
2. *When an argument $\alpha = [\gamma_0, \dots, \gamma_n]$ is valid, it can be extended to a valid argument α' by adding a premise φ provided that, for each applicable rule $\psi \Rightarrow \chi$ occurring in α such that $C(\alpha) \wedge \varphi \not\mid\!\sim \chi$, an $\mid\!\sim$-exception overruling the rule is also added (after the new premise). Each thus excluded rule is not applicable in α'. Also each rule that comes after an excluded rule is not applicable in α'. α' is an argument from $P(\alpha) \wedge \varphi$ making the case $P(\alpha) \wedge \varphi \wedge \psi_0 \wedge \dots \wedge \psi_k$, where $\psi_0, \dots, \psi_k$ are the consequents of the applicable rules of α'.*
3. *When an argument $\alpha = [\gamma_0, \dots, \gamma_n]$ is valid, it can be extended to a valid argument α' by adding a $\mid\!\sim$-rule $\varphi \Rightarrow \psi$ provided that $C(\alpha) \vdash \varphi$ and $C(\alpha) \mid\!\sim \psi$. That rule is an additional applicable rule of α'. The extended argument's case is $C(\alpha) \wedge \psi$.*

Theorem 3. *Let $\mid\!\sim$ express qualitative ampliative inference. Then $\varphi \mid\!\sim \psi$ if and only if there is a valid argument, structured as in Definition 6, from φ making the case $\varphi \wedge \psi$.*

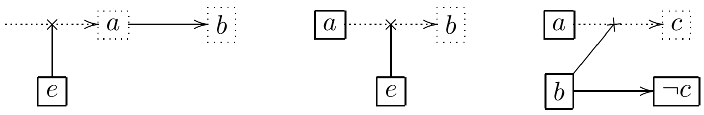
By this theorem, we can use the same term 'argument' for an argument in the sense of a global judgment of warrant and for an argument in the sense of a valid structured argument, as defined in Definition 6.

Assume for instance that the argument $[p, p \Rightarrow q, q \Rightarrow r, e, \neg(e \wedge q \Rightarrow r)]$ is valid, where all rules and exceptions are taken from a qualitative ampliative inference relation $\mid\!\sim$. This argument's construction consists of three steps. Initially there is only the premise p, then the rule $p \Rightarrow q$ is applied, then a second rule $q \Rightarrow r$, but that rule is immediately excluded by the exception e. The argument is an argument from $p \wedge e$ making the case $p \wedge q \wedge e$.

As said, the construction of valid arguments is more straightforward than in other proposals because the properties of qualitative ampliative inference restrict which rules and exceptions can form an argument. Consider for instance this argument, a slight variant of a puzzle studied by Pollock [14]: $[p, p \Rightarrow q, q \Rightarrow r, \neg(p \wedge r \Rightarrow q)]$. After the application of the second rule, the first one is excluded, making the application of the second rule impossible; contradiction. In the present proposal, this example is not a valid argument of a qualitative ampliative inference relation $\mid\!\sim$. For if the two rules can both be validly applied, we would have that $p \mid\!\sim q \wedge r$, hence by (CCM) and (RW) (and (LE)) $p \wedge r \mid\!\sim q$, so $\neg(p \wedge r \Rightarrow q)$ is not an $\mid\!\sim$-exception.

What about the expressiveness of the present approach? Isn't it too restrictive? Let's consider the three kinds of argument attack that are most prominent in state-of-the-art argumentation formalizations (e.g., [15]): *assumption-based attack*, aimed at the defeasible assumptions of an argument, *undercutting attack*, aimed at the connection between a reason and its conclusion, and *rebutting attack*, in which a reason for the opposite of the argument's conclusion is given.

Consider the following three arguments, one of each type:

 (4)

The argument $[\Rightarrow a, a \Rightarrow b, e, \neg(e \Rightarrow a)]$, depicted on the left, in which the defeasible assumption a is attacked by e, is an example of assumption-based attack. The argument from a to b is based on the defeasible assumption a (formalized by the rule $\Rightarrow a$ with empty antecedent). The assumption is attacked by the exception e. When the argument is valid (with respect to a qualitative inference relation $\mid\!\sim$), the rules and exceptions in the argument correspond to $\mid\!\sim a, a \mid\!\sim b$ and $e \not\mid\!\sim a$. The gradual construction of the argument starts with the assumption. At this stage of construction, the argument's validity corresponds to $\mid\!\sim a$ (cf. Theorem 3). Then the rule $a \Rightarrow b$ is applied. The argument's validity now corresponds to $\mid\!\sim a \wedge b$. Finally, the premise e is added, that leads to the addition of an exception $\neg(e \Rightarrow a)$ excluding the assumption $\Rightarrow a$. The argument's validity now corresponds to $e \not\mid\!\sim a$ and $e \mid\!\sim e$.

Similarly for the middle, undercutting argument $[a, a \Rightarrow b, e, \neg(e \wedge a \Rightarrow b)]$, in which the reason a for b is undercut by e. Its rule and exception correspond to $a \mid\!\sim b$ and $a \wedge e \not\mid\!\sim b$. Its gradual construction corresponds to the sequence $a \mid\!\sim a, a \mid\!\sim a \wedge b, e \wedge a \not\mid\!\sim b, e \wedge a \mid\!\sim e \wedge a$.

The argument on the right, $[a, a \Rightarrow c, b, \neg(a \wedge b \Rightarrow c), b \Rightarrow \neg c]$, an example of a reason b for $\neg c$ that rebuts the reason a for c, has rules corresponding to $a \mid\!\sim c$ and $b \mid\!\sim \neg c$ and an exception corresponding to $a \wedge b \not\mid\!\sim c$. Its gradual construction corresponds to $a \mid\!\sim a, a \mid\!\sim a \wedge c, a \wedge b \not\mid\!\sim c, a \wedge b \mid\!\sim \neg c$.

6 Summary and Concluding Remarks

In this paper, the foundations of argumentation theory have been reconsidered using the formal properties of nonmonotonic consequence as a starting point [2,11,12]. Well-known properties of a nonmonotonic consequence relation $\mid\!\sim$ were reinterpreted in terms of a so-called ampliation relation $\sim$. By this reinterpretation, ampliative inference could be connected to properties studied in the mathematical theory of (pre)orders. This allowed the development of a quantitative interpretation of qualitative ampliative inference, in two versions.

The first quantitative interpretation was in terms of a numeric function measuring an 'order of magnitude' of a proposition (if you like: an 'order of normality'), with larger orders being more general (with the order of $\top$ as maximum) and lower orders more specific (with the order of $\bot$ as minimum). The properties of this order-of-magnitude function remind of possibility logic [4] by its use of a maximum property (as opposed to the additivity of probability), with a key difference being the use of an inequality. The order-of-magnitude function helps to explain why preferential logic (which can be obtained from our qualitative ampliative inference by adding the (And) and (Or) rules) can be defined both in terms of limits (Geffner & Pearl [6]) and in terms of minimal models (Kraus, Lehmann, Magidor [11]): a premise and its conclusions have the same order of magnitude, with a difference that vanishes 'in the limit'.

From the order function, a second quantitative interpretation of qualitative ampliative inference has been derived in terms of an 'argument value' function with properties that closely resemble the Kolmogorov probability axioms [7]. This second 'logico-probabilistic' interpretation of ampliative arguments uses numeric argument values that behave like conditional probabilities and that measure argument strength. When the strength of an argument is above a threshold, the argument is valid. The Kolmogorov additivity axiom is replaced by an inequality, that has the natural interpretation that the cases that are the basis of the numeric distribution (e.g., by counting observations or by otherwise weighting them) contain partial information, and not information about the 'whole world'. This partiality of information is a cornerstone of ampliative inference, that can by our formal proposal be interpreted as *jumping to conclusions* when staying sufficiently close in value to the premises.

One interpretation of the case made by an argument, defined here as the conjunction of premises and conclusions, is as an explanation supported by the argument's premises. As a result, the approach in this paper is related to theories of abductive inference to the best explanation [10]. This is no coincidence since one inspiration for the present approach was the intuition that argument-based and explanation-based approaches could be formally integrated (cf. Bex's hybrid argumentative-narrative theory of legal evidential reasoning [3]).

Our approach can be regarded as a bridge between logic and probability. While [8] consider the (And) rule as a watershed between qualitative and quantitative interpretations of nonmonotonic reasoning, we have proposed a quantitative interpretation of qualitative ampliative arguments that is independent of (And). Here, the (CCT) property, that expresses a kind of defeasible rule application and is probabilistically not valid, plays a key role in distinguishing quantitative and qualitative argumentation.

Bayesian networks [9,13] also combine logic and probability. They are however often causally interpreted and need additional tools for the modeling of an agent's decision making, e.g., utilities. By the combination of the decision-oriented notions of defeasible argument, rules with exceptions and argument strength, the here proposed theory provides a fresh perspective on intelligent agents jumping to conclusions in order to interpret their world and act in it.

The distinguishing role of defeasible rule application and the associated property (CCT) shows on formal grounds that the rules and exceptions underlying qualitative ampliative inference cannot be derived from statistical correlations alone. Defeasible rules need to be tested by applying them, thereby generating hypotheses. Sometimes rules with their associated exceptions will lead to hypotheses that are not falsified too often. When this happens, such rules can be regarded as 'knowledge', in the sense that they describe accurate patterns.

Acknowledgements. The author would like to thank Allard Tamminga and Rineke Verbrugge for their valuable comments.

References

1. Baroni, P., Caminada, M., Giacomin, M.: Review: an introduction to argumentation semantics. The Knowledge Engineering Review 26(4), 365–410 (2011)
2. van Benthem, J.: Foundations of conditional logic. Journal of Philosophical Logic, 303–349 (1984)
3. Bex, F., Van Koppen, P., Prakken, H., Verheij, B.: A hybrid formal theory of arguments, stories and criminal evidence. Artificial Intelligence and Law, 1–30 (2010)
4. Dubois, D., Prade, H.: Possibility theory, probability theory and multiple-valued logics: A clarification. Annals of Mathematics and Artificial Intelligence 32(1), 35–66 (2001)
5. Dung, P.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77, 321–357 (1995)
6. Geffner, H., Pearl, J.: Conditional entailment: Bridging two approaches to default reasoning. Artificial Intelligence 53(2-3), 209–244 (1992)
7. Hájek, A.: Interpretations of probability. In: Zalta, E. (ed.) The Stanford Encyclopedia of Philosophy, Stanford University (2011)
8. Hawthorne, J., Makinson, D.: The quantitative/qualitative watershed for rules of uncertain inference. Studia Logica 86(2), 247–297 (2007)
9. Jensen, F., Nielsen, T.: Bayesian networks and decision graphs. Springer, Berlin (2007)
10. Josephson, J., Josephson, S.: Abductive Inference: Computation, Philosophy, Technology. Cambridge University Press, Cambridge (1996)
11. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44, 167–207 (1990)
12. Makinson, D.: General patterns in nonmonotonic reasoning. In: Gabbay, D., Hogger, C., Robinson, J. (eds.) Handbook of Logic in Artificial Intelligence and Logic Programming. Nonmonotonic Reasoning and Uncertain Reasoning, vol. 3, pp. 35–110. Clarendon Press, Oxford (1994)
13. Pearl, J.: Causality. Cambridge University Press, Cambridge (2000)
14. Pollock, J.: Cognitive Carpentry: A Blueprint for How to Build a Person. The MIT Press, Cambridge (1995)
15. Prakken, H.: An abstract framework for argumentation with structured arguments. Argument and Computation 1(2), 93–124 (2010)
16. Rahwan, I., Simari, G. (eds.): Argumentation in Artificial Intelligence. Springer, Dordrecht (2009)
17. Verheij, B.: Two approaches to dialectical argumentation: admissible sets and argumentation stages. In: Meyer, J.J., van der Gaag, L. (eds.) Proceedings of NAIC 1996, pp. 357–368. Universiteit Utrecht, Utrecht (1996)

Beyond Maxi-Consistent Argumentation Operators[⋆]

Srdjan Vesic and Leendert van der Torre

Computer Science and Communication
University of Luxembourg
{srdjan.vesic,leon.vandertorre}@uni.lu

Abstract. The question whether Dung's abstract argumentation theory can be instantiated with classical propositional logic has drawn a considerable amount of attention among scientists in recent years. It was shown by Cayrol in 1995 that if *direct undercut* is used, then stable extensions of an argumentation system correspond exactly to maximal (for set inclusion) consistent subsets of the knowledge base from which the argumentation system was constructed. Until now, no other correspondences were found between the extensions of an argumentation framework and its knowledge base (except if preferences are also given at the input of the system). This paper's contribution is twofold. First, we identify four intuitive conditions describing a class of attack relations which return extensions corresponding exactly to the maximal (for set inclusion) consistent subsets of the knowledge base. Second, we show that if we relax those conditions, it is possible to instantiate Dung's abstract argumentation theory with classical propositional logic and obtain a meaningful result which does not correspond to the maximal consistent subsets of the knowledge base used for constructing arguments. Indeed, we define a whole class of instantiations that return different results. Furthermore, we show that these instantiations are sound in the sense that they satisfy the postulates from argumentation literature (e.g. consistency, closure). In order to illustrate our results, we present one particular instantiation from this class, which is based on cardinalities of minimal inconsistent sets a formula belongs to.

1 Introduction

The question how to reason in presence of inconsistency is on of the keywords of logic and artificial intelligence. A notable example are paraconsistent logics [11] where one is able to draw some (but not all) conclusions from an inconsistent set of formulae. As another example take belief revision, belief merging or voting [8]. Generally speaking, an inference relation is a way to go from a (possibly inconsistent) knowledge base to a set of subsets of that knowledge base. For example, given a knowledge base $\{\varphi, \neg\varphi \wedge \psi\}$, an inference relation could return two sets: $\{\varphi\}$ and $\{\neg\varphi \wedge \psi\}$. One of the simplest inference relations is a function returning the set of all maximal (for set inclusion) consistent subsets of a knowledge base. It has been shown [6] that the result obtained

⋆ SV was funded by the National Research Fund, Luxembourg. His work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. This Programme is supported by the Marie Curie Co-funding of Regional, National and International Programmes (COFUND) of the European Commission.

by this inference relation can be also obtained by an instantiation of Dung's abstract argumentation theory [7]. Namely, when *direct undercut* is used as attack relation on the set of all the arguments built from a knowledge base, then stable extensions of the resulting argumentation framework correspond exactly to the set of maximal consistent subsets of the knowledge base. This is the first result which shows that Dung's abstract argumentation theory can be instantiated in a way to capture an inference relation.

An important question is whether Dung's theory can be used as a general framework for nonmonotonic logic, and if so, which class of inference relations can be studied as instances of Dung's theory. Indeed, only a very small fragment of logics has been represented in such a way. This may also not be very surprising, given the richness of the logic literature and the strong constraints imposed by Dung's theory. This raises two important questions for the community. First, which class of logics can be captured by Dung's theory? Second, how to generalize Dung's theory? In this paper we address the first question, whereas the second question is a part of our long term research agenda.

The starting point of our work is to note that since the first result [6] showing how to capture an inference relation in Dung's theory, not much work has been done in this direction. Indeed, no "reasonable" logic-based instantiations of Dung's abstract theory were found that capture another inference relation. By "reasonable", we mean that they satisfy at least some basic postulates proposed for instantiated argumentation frameworks [5] like consistency, closure, and so on.

The challenges of this paper are: First, is it possible to define conditions that characterize the circumstances when a semantics returns maximal consistent subsets under subset relation? Second, how to define a class of attack relations in terms of the knowledge base such that the stable extensions of the obtained argumentation framework do not correspond to exactly to the maximal for set inclusion consistent subsets of the knowledge base? Third, how to ensure that those instantiations of Dung's theory still return a reasonable result?

The layout of this paper is as follows: After introducing the notions of argumentation framework and formally defining its logic-based instantiations (Section 2), we identify four conditions describing a class of attack relations returning extensions corresponding to maximal consistent subsets of a knowledge base (Section 3). Then, we show that if two conditions are dropped, it is possible to instantiate Dung's abstract argumentation theory in a meaningful way and obtain a substantially different result (Section 4). The last section concludes and reviews questions left for future work. The proofs are omitted due to the space restrictions.

2 Dung's Abstract Argumentation Theory and Its Instantiation with Classical Propositional Logic

In this section, we present the most common way of instantiating Dung's abstract argumentation theory [7] with classical propositional logic. $\mathcal{L}$ denotes the set of well-formed formulae, $\vdash$ stands for classical entailment, and $\equiv$ for logical equivalence. We denote by Σ a finite set of classical propositional formulae from which arguments are constructed. We use the notation $\texttt{MC}(\Sigma)$ for the set of all maximal (for set inclusion) consistent subsets of Σ, and $\texttt{MinConf}(\Sigma)$ for the set of minimal (for set inclusion) inconsistent subsets of Σ. A formula φ is called a free formula of a knowledge base Σ if and only if φ does not belong to any minimal (for set inclusion) inconsistent subset of Σ. A logical argument is defined as a pair $(support, conclusion)$.

Definition 1 (Argument). *Let $\mathcal{L}$ be a classical propositional language with $\vdash$ its associated logical consequence, let $\Sigma \subseteq \mathcal{L}$ and $\alpha \in \mathcal{L}$. An argument is a pair (Φ, α) such that $\Phi \subseteq \Sigma$ is a minimal (for set inclusion) consistent set of formulae such that $\Phi \vdash \alpha$.*

Example 1. Let $\Sigma = \{\varphi, \varphi \to \psi, \omega\}$. $(\{\varphi, \varphi \to \psi\}, \psi)$, $(\{\varphi \to \psi\}, \neg\varphi \vee \psi)$ and $(\{\varphi, \psi\}, \varphi \leftrightarrow \psi)$ are some of the arguments that can be constructed from Σ.

For an argument $a = (\Phi, \alpha)$, we write $\texttt{Supp}(a) = \Phi$ to denote its support and $\texttt{Conc}(a) = \alpha$ to denote its conclusion. For a set of arguments $\mathcal{E}$, we denote by $\texttt{Concs}(\mathcal{E})$ the set of conclusions of all the arguments from $\mathcal{E}$. In other words, $\texttt{Concs}(\mathcal{E}) = \{\texttt{Conc}(a) \mid a \in \mathcal{E}\}$. For a given set of formulae $S \subseteq \mathcal{L}$, we denote by $\texttt{Arg}(S)$ the set of arguments constructed from S. Formally, $Arg(S) = \{a \mid a$ is an argument and $\texttt{Supp}(a) \subseteq S\}$. Let $\texttt{Arg}(\mathcal{L})$ denote the set of all arguments that could be made from propositional logic formulae. For a given set of arguments $\mathcal{E}$, we denote $\texttt{Base}(\mathcal{E}) = \bigcup_{a \in \mathcal{E}} \texttt{Supp}(a)$. Now we provide a definition of argumentation framework.

Definition 2 (Argumentation framework). *An argumentation framework is a pair $(\mathcal{A}, \mathcal{R})$ where $\mathcal{A} \subseteq \texttt{Arg}(\mathcal{L})$ is a set of arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ a binary relation. For each pair $(a, b) \in \mathcal{R}$, we say that a attacks b. We also sometimes use notation $a\mathcal{R}b$ instead of $(a, b) \in \mathcal{R}$.*

In the rest of the paper, we suppose that all the arguments from Σ are constructed, i.e. that $\mathcal{A} = \texttt{Arg}(\Sigma)$. We now introduce the notions of conflict-freeness and defence used to define different semantics.

Definition 3 (Conflict-free, defence). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an argumentation framework, $\mathcal{E} \subseteq \mathcal{A}$ and $a \in \mathcal{A}$.*

- *$\mathcal{E}$ is* conflict-free *if and only if there exist no two arguments $a, b \in \mathcal{E}$ s.t. $(a, b) \in \mathcal{R}$*
- *$\mathcal{E}$ defends a if and only if for every $b \in \mathcal{A}$ we have that if $b\mathcal{R}a$ then there exists $c \in \mathcal{E}$ such that $c\mathcal{R}b$.*

Let us now define the most commonly used acceptability semantics.

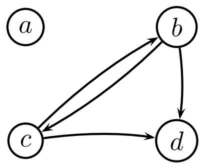
Definition 4 (Acceptability semantics). *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an argumentation framework and $\mathcal{B} \subseteq \mathcal{A}$. We say that a set $\mathcal{B}$ is* admissible *if and only if it is conflict-free and defends all its elements.*

- *$\mathcal{B}$ is a* complete *extension if and only if $\mathcal{B}$ defends all its arguments and contains all the arguments it defends.*
- *$\mathcal{B}$ is a* preferred *extension if and only if it is a maximal (with respect to set inclusion) admissible set.*
- *$\mathcal{B}$ is a* stable *extension if and only if $\mathcal{B}$ is conflict-free and for all $a \in \mathcal{A} \setminus \mathcal{B}$, there exists $b \in \mathcal{B}$ such that $b\ \mathcal{R}\ a$.*
- *$\mathcal{B}$ is a* semi-stable *extension if and only if $\mathcal{B}$ is a complete extension and the union of the set $\mathcal{B}$ and the set of all arguments attacked by $\mathcal{B}$ is maximal (for set inclusion).*

- *$\mathcal{B}$ is a* grounded extension *if and only if $\mathcal{B}$ is a minimal (for set inclusion) complete extension.*
- *$\mathcal{B}$ is an* ideal *extension if and only if $\mathcal{B}$ is a maximal (for set inclusion) admissible set contained in every preferred extension.*

For an argumentation framework $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ we denote by $\mathtt{Ext}_x(\mathcal{F})$; or, by a slight abuse of notation, by $\mathtt{Ext}_x(\mathcal{A}, \mathcal{R})$ the set of its extensions with respect to semantics x. We use abbreviations c, p, s, ss, g and i for respectively complete, preferred, stable, semi-stable, grounded and ideal semantics. For example, $\mathtt{Ext}_p(\mathcal{F})$ denotes the set of preferred extensions argumentation framework $\mathcal{F}$.

Example 2. Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an argumentation framework with $\mathcal{A} = \{a, b, c, d\}$ and $\mathcal{R} = \{(b, c), (c, b), (b, d), (c, d)\}$. The graph is visualised below.



There are three complete extensions: $\{a, b\}$, $\{a, c\}$ and $\{a\}$, and two preferred / stable / semi-stable extensions: $\{a, b\}$ and $\{a, c\}$. The grounded extension of this framework coincides with its ideal extension, which is the set $\{a\}$.

We now introduce the most common ways in which attack relations are defined in terms of (i.e. as functions of) the knowledge base in argumentation literature [9].

Definition 5. *For a set of formulae $\Phi = \{\varphi_1, \ldots, \varphi_k\}$, let $\bigwedge \Phi$ denote $\varphi_1 \wedge \ldots \wedge \varphi_k$. Let $a, b \in \mathtt{Arg}(\mathcal{L})$. We define the following attack relations:*

- defeat*: $a\mathcal{R}_d b$ if and only if* $\mathtt{Conc}(a) \vdash \neg \bigwedge \mathtt{Supp}(b)$
- direct defeat*: $a\mathcal{R}_{dd} b$ if and only if there exists $\varphi \in \mathtt{Supp}(b)$ s.t.* $\mathtt{Conc}(a) \vdash \neg\varphi$
- undercut*: $a\mathcal{R}_u b$ if and only if there exists $\Phi \subseteq \mathtt{Supp}(b)$ such that* $\mathtt{Conc}(a) \equiv \neg \bigwedge \Phi$
- direct undercut*: $a\mathcal{R}_{du} b$ if and only if there exists $\varphi \in \mathtt{Supp}(b)$ s.t.* $\mathtt{Conc}(a) \equiv \neg\varphi$
- canonical undercut*: $a\mathcal{R}_{cu} b$ if and only if* $\mathtt{Conc}(a) \equiv \neg \bigwedge \mathtt{Supp}(b)$
- rebut*: $a\mathcal{R}_r b$ if and only if* $\mathtt{Conc}(a) \equiv \neg\mathtt{Conc}(b)$
- defeating rebut*: $a\mathcal{R}_{dr} b$ if and only if* $\mathtt{Conc}(a) \vdash \neg\mathtt{Conc}(b)$

Note that all the attack relations from the previous definition are defined on $\mathtt{Arg}(\mathcal{L}) \times \mathtt{Arg}(\mathcal{L})$. For a given Σ, one can just use the restriction of the relation from $\mathtt{Arg}(\mathcal{L}) \times \mathtt{Arg}(\mathcal{L})$ to $\mathtt{Arg}(\Sigma) \times \mathtt{Arg}(\Sigma)$. This is not the case with all the attack relations we use in this paper. Namely, for some attack relations we use, there exist arguments $a, b \in \mathtt{Arg}(\mathcal{L})$, such that whether a attacks b or not depends also on the knowledge base Σ. Formally, the more general case is when an attack relation is defined by specifying its behaviour on every $\mathtt{Arg}(\Sigma) \times \mathtt{Arg}(\Sigma)$ for every finite $\Sigma \subseteq \mathcal{L}$. In the rest of the paper, when we use the term “attack relation”, we refer to the more general case. Formally, one should write $(a, b, \Sigma) \in \mathcal{R}$. However, since it is always clear to which Σ we refer to, there is no danger of confusion and in order to simplify the notation we write $(a, b) \in \mathcal{R}$ or $a\mathcal{R}b$ throughout the paper.

3 Some Hypotheses Leading to Maximal Consistent Subsets of the Knowledge Base

In this section, we identify four simple and intuitive conditions such that every instantiation which satisfies all of them returns the extensions corresponding exactly to maximal consistent subsets of the knowledge base arguments are constructed from. Namely, similar to the principles that can be satisfied by an acceptability semantics [4], there exist principles that an attack relation can satisfy [1,5]. An important requirement is that an attack relation should return consistent extensions (abbreviated $\mathtt{CE}$).

Definition 6 (CE). *Let $\mathcal{R}$ be an attack relation. We say that $\mathcal{R}$ returns consistent extensions under semantics x if and only if for every $\Sigma \subseteq \mathcal{L}$, for every $\mathcal{F} = (\mathtt{Arg}(\Sigma), \mathcal{R})$, for every extension $\mathcal{E}$ of $\mathcal{F}$ under semantics x, it holds that $\mathtt{Base}(\mathcal{E})$ is a consistent set.*

Example 3. $\mathcal{R}_{du}$ and $\mathcal{R}_{dd}$ satisfy $\mathtt{CE}$ under stable, semi-stable, preferred, and complete semantics, whereas $\mathcal{R}_u, \mathcal{R}_{cu}, \mathcal{R}_r, \mathcal{R}_{dr}$ do not satisfy $\mathtt{CE}$ under neither of those semantics [9].

Another requirement in logic-based argumentation is that an argument should not attack another one if the union of their supports is consistent. This property of an attack relation is called conflict-dependence [1] for what we use the abbreviation $\mathtt{CD}$.

Definition 7 (CD). *Let $\mathcal{R}$ be an attack relation. We say that $\mathcal{R}$ is conflict-dependent if and only if for every $\Sigma \subseteq \mathcal{L}$, for every $a, b \in \mathtt{Arg}(\Sigma)$, if $(a, b) \in \mathcal{R}$ then $\mathtt{Supp}(a) \cup \mathtt{Supp}(b) \vdash \perp$.*

Example 4. Attack relations $\mathcal{R}_{dd}, \mathcal{R}_u, \mathcal{R}_{du}, \mathcal{R}_{cu}, \mathcal{R}_r, \mathcal{R}_{dr}$ are conflict-dependent.

The next requirement specifies that the arguments having the same support are attacked by the same arguments. We call this $\mathtt{AS}$ (“assumption attack”).

Definition 8 (AS). *Let $\mathcal{R}$ be an attack relation. We say that $\mathcal{R}$ satisfies $\mathtt{AS}$ if and only if for every $\Sigma \subseteq \mathcal{L}$, for every $a, b, c \in \mathtt{Arg}(\Sigma)$, if $\mathtt{Supp}(b) = \mathtt{Supp}(c)$ then $a\mathcal{R}b$ if and only if $a\mathcal{R}c$.*

Note that $\mathtt{AS}$ is already present in argumentation literature [3,9].

Example 5. Attack relations $\mathcal{R}_{dd}, \mathcal{R}_u, \mathcal{R}_{du}$ and $\mathcal{R}_{cu}$ satisfy $\mathtt{AS}$, whereas $\mathcal{R}_r$ and $\mathcal{R}_{dr}$ do not.

The last property we consider in this paper specifies that when one constructs a set $\mathtt{Arg}(S)$ containing all the arguments made from a maximal consistent set S, then every argument outside of $\mathtt{Arg}(S)$ is attacked by at least one argument from $\mathtt{Arg}(S)$. We call the resulting condition $\mathtt{MS}$, which is an abbreviation telling that the intuition behind it is that any maximal consistent set should be stable.

Definition 9 (MS). *Let $\mathcal{R}$ be an attack relation. We say that $\mathcal{R}$ satisfies* $\texttt{MS}$ *if and only if for every $\Sigma \subseteq \mathcal{L}$, for every $S \in \texttt{MC}(\Sigma)$, for every $a' \in \texttt{Arg}(\Sigma) \setminus \texttt{Arg}(S)$, there exists $a \in \texttt{Arg}(S)$ such that $(a, a') \in \mathcal{R}$.*

To the best of our knowledge, this property was not formally stated until now.

Example 6. Attack relations $\mathcal{R}_{dd}, \mathcal{R}_u, \mathcal{R}_{du}, \mathcal{R}_{cu}$ satisfy $\texttt{MS}$.

Those conditions seem as properties one would like an attack relation to satisfy (at least in some contexts). The goal of this section is to answer the question: is it possible to define an instantiation of Dung's theory that captures reasoning substantially different from the approach which returns maximal consistent subsets and at the same satisfies $\texttt{CE}$, $\texttt{CD}$, $\texttt{AS}$ and $\texttt{MS}$? The answer is no, as shown by Proposition 3.

We start by defining a notion of independence of a set of formulae, which is used to describe the extensions of attack relations satisfying $\texttt{AS}$. The idea is that no formula in a set can be derived from other formulae of that set.

Definition 10 (Independent set of formulae). *A set $S \subseteq \mathcal{L}$ is independent if and only if there exists no formula $\varphi \in S$ s.t. $S \setminus \{\varphi\} \vdash \varphi$.*

Our first goal is to show that for the class of attack relations satisfying $\texttt{AS}$, conclusion of an argument has no impact on its acceptability. In other words, the membership to an extension is uniquely determined by argument's support. To prove this result, we need the following lemma.

Lemma 1. *Let $\mathcal{R}$ be an attack relation satisfying* $\texttt{AS}$*, let $\Sigma \subseteq \mathcal{L}$ be a knowledge base, $\mathcal{F} = (\texttt{Arg}(\Sigma), \mathcal{R})$ and let $\mathcal{E} \subseteq \texttt{Arg}(\mathcal{A})$ an admissible set. Let $a, b \in \texttt{Arg}(\Sigma)$ be two arguments such that* $\texttt{Supp}(a) = \texttt{Supp}(b)$*, $a \in \mathcal{E}$ and $b \notin \mathcal{E}$. Then, $\mathcal{E} \cup \{b\}$ is also an admissible set.*

We can now show that if two arguments have the same support, and an attack relation satisfying $\texttt{AS}$ is used, those two arguments are exactly in the same extensions.

Proposition 1. *Let $\mathcal{R}$ be an attack relation satisfying* $\texttt{AS}$*, let $\Sigma \subseteq \mathcal{L}$ be a knowledge base, $\mathcal{F} = (\texttt{Arg}(\Sigma), \mathcal{R})$ and $\mathcal{E} \in \texttt{Ext}_x(\mathcal{F})$ with $x \in \{s, ss, p, g, i\}$. Let $a, b \in \texttt{Arg}(\Sigma)$ and* $\texttt{Supp}(a) = \texttt{Supp}(b)$*. Then, $a \in \mathcal{E}$ if and only if $b \in \mathcal{E}$.*

We can now show that for attack relations satisfying $\texttt{AS}$, every extension can be characterised by a collection of sets of formulae.

Proposition 2. *Let $\mathcal{R}$ be an attack relation satisfying* $\texttt{AS}$*, let $\Sigma \subseteq \mathcal{L}$ be a knowledge base, $\mathcal{F} = (\texttt{Arg}(\Sigma), \mathcal{R})$ and $\mathcal{E} \in \texttt{Ext}_x(\mathcal{F})$ with $x \in \{s, ss, p, g, i\}$. Then: there exists a unique collection of sets $S_1, \ldots, S_n \subseteq \Sigma$ such that:*

1. *every S_i is consistent*
2. *every S_i is independent*
3. $\mathcal{E} = \{a \in \texttt{Arg}(\mathcal{L}) \mid$ *there exists S_i such that* $\texttt{Supp}(a) = S_i\}$.

The significance of the previous result lays in the fact that it is a step forward towards understanding the expressivity of attack relations satisfying $\texttt{AS}$. Namely, is shows that every extension can be fully characterised by a unique collection of consistent and independent sets. Roughly speaking, every attack relation satisfying $\texttt{AS}$ provides us with no more or less information than a function which separates Σ in a finite number of collections of consistent and independent sets.

We can now prove that if an attack relation satisfies $\texttt{CE}_s$, $\texttt{CD}$, $\texttt{AS}$ and $\texttt{MS}$, then its extensions are exactly the sets of arguments constructed from maximal consistent subsets of the knowledge base. In other words, for any maximal consistent subset S of Σ, the set of all arguments constructed from S is an extension, and for any extension, there exists a maximal consistent set $S \subseteq \Sigma$ such that $\mathcal{E} = \texttt{Arg}(S)$.

Proposition 3. *Let $\mathcal{R}$ be an attack relation satisfying $\texttt{CE}_s$, $\texttt{CD}$, $\texttt{AS}$ and $\texttt{MS}$. Then, for every $\Sigma \subseteq \mathcal{L}$, extensions of $(\texttt{Arg}(\Sigma), \mathcal{R})$ under stable semantics are exactly $\{\texttt{Arg}(S) \mid S \in \texttt{MC}(\Sigma)\}$.*

The previous result shows that the attack relations satisfying $\texttt{CE}_s$, $\texttt{CD}$, $\texttt{AS}$ and $\texttt{MS}$ simply mimic the result obtained by selecting the maximal consistent subsets of the knowledge base. This proposition is proved under stable semantics, but we believe that similar results can be obtained for other acceptability semantics, which will be a part of our future work.

4 A New Class of Instantiations: Beyond Maximal Consistent Sets

In this section, we show that if conditions $\texttt{AS}$ and $\texttt{MS}$ are dropped, it is possible to define a new instantiation of Dung's abstract argumentation theory which captures a result different from maximal consistent subsets of a knowledge base by and at the same time: i) uses only the information from the knowledge base (i.e. no external data about the preferences, values...), ii) and satisfies postulates from the argumentation literature (e.g. consistency, closure).

In general, it is possible to go from a knowledge base to a set of extensions in two steps. First, we define a measure, attaching to each element of a knowledge base a value; second, we define a procedure using that measure to calculate extensions. First, one can define different measures on the set of formulae of a propositional knowledge base. Second, once we have a measure, there are still many ways to go from the knowledge base and the measure to the sets of extensions. We can for example try to define an attack relation such that an extension contains the elements having a minimal sum of values. In this paper, we use the approach inspired by the work of Amgoud and Vesic [2]. The idea is to construct an attack relation which makes extensions contain as much elements having low values as possible, until a maximal consistent subset of a knowledge base is reached.

4.1 Shapley Inconsistency Value of a Formula

The main idea behind the class of instantiations we propose is that the arguments made from “less inconsistent” formulae have “more chance” to be in extensions. This means that we need a tool for indicating how inconsistent a set or a formulae is. In this paper, we use Shapley Inconsistency Values, introduced by Hunter and Konieczny [10], to obtain that measure. This concept for measuring inconsistency is inspired by a Shapley Value, which was originally developed by Shapley [12] for defining merits of each individual of a coalition in a cooperative game theory.

The idea behind the class of instantiations we propose is that a user is free to choose a basic inconsistency measure, under the condition that it satisfies the four properties we state in the following definition. The corresponding Shapley Inconsistency Value can then be calculated automatically. Thus, different basic inconsistency measures give different Shapley Inconsistency Values.

Note that we present only the most important concepts linked to the definition of a Shapley Inconsistency Value, for more details the reader is referred to the paper in which they were introduced [10].

Definition 11 (Basic inconsistency measure [10]). *A basic inconsistency measure I is a function that for every finite set of formulae returns a real number and satisfies the following properties for all finite sets $\Sigma, \Sigma' \subseteq \mathcal{L}$ and all formulae $\varphi, \psi \in \mathcal{L}$:*

- *$I(\Sigma) = 0$ if and only if Σ is a consistent set* *(Consistency)*
- *$I(\Sigma \cup \Sigma') \geq I(\Sigma)$* *(Monotony)*
- *If φ is a free formula of $\Sigma \cup \varphi$, then $I(\Sigma \cup \varphi) = I(\Sigma)$* *(Free Formula Independ.)*
- *If $\varphi \vdash \psi$ and $\varphi \nvdash \perp$, then $I(\Sigma \cup \{\varphi\}) \geq I(\Sigma \cup \{\psi\})$* *(Dominance)*

A basic inconsistency measure gives a number indicating how conflicting a knowledge base is. Let us give an example of a basic inconsistency measure.

Definition 12 (MI inconsistency measure [10]). *The* `MI` *inconsistency measure is defined as the number of minimal inconsistent subsets of Σ, i.e.*

$$I_{\mathtt{MI}}(\Sigma) = |\ \mathtt{MinConf}(\Sigma)\ |$$

Example 7. Let $\Sigma = \{\varphi, \neg\varphi, \varphi \to \psi, \neg\psi, \omega\}$. Then, $\mathtt{MinConf}(\Sigma) = \{C_1, C_2\}$, with $C_1 = \{\varphi, \neg\varphi\}$ and $C_2 = \{\varphi, \varphi \to \psi, \neg\psi\}$. Thus, $\mathtt{MI}(\Sigma) = 2$.

The `MI` inconsistency measure is a basic inconsistency measure.

Originally, Shapley's idea was to measure the merit of an individual in a coalition. Here, the idea is to use it to measure the "blame" of a formula for the inconsistency of a knowledge base. To do that, the identical mathematical expression from Shapley [12] is used, but with different interpretation.

Definition 13 (Shapley Inconsistency Value [10]). *Let $\Sigma \subseteq \mathcal{L}$ and let I be a basic inconsistency measure. We define the corresponding Shapley Inconsistency Value (SIV), noted S^I, as follows. For every $\varphi \in \Sigma$:*

$$S^I_\varphi(\Sigma) = \sum_{S \subseteq \Sigma} \frac{(|S|-1)!(|\Sigma|-|S|)!}{|\Sigma|!}(I(S) - I(S \setminus \{\varphi\})).$$

Beside the fact that this measure gives very sensible results, it has also been shown that the previous formula is the only one which satisfies a set of intuitive axioms for measuring inconsistency [10]. This SIV gives a value for each formula of the base Σ. Thus, the previous definition allows us to define to what extent a formula is concerned with the inconsistencies. Note that for a formula φ, SIV depends essentially on the sum of differences of inconsistencies of all subsets of Σ together and without φ. Those values are then just multiplied with coefficients which depend only on the cardinalities of the corresponding sets. So, the main intuition can be resumed in: "How much does inconsistency decrease when φ is removed?"

It has been shown [10] that the SIV corresponding to basic inconsistency measure `MI` is:

$$S_{\varphi}^{I_{MI}}(\Sigma) = \sum_{C \in \texttt{MinConf}(\Sigma) \text{ such that } \varphi \in C} \frac{1}{|C|}.$$

In other words, the inconsistency blame of a formula φ is obtained by summing up the values $\frac{1}{|C|}$ for all minimal conflicts C such that $\varphi \in C$.

Example 8 (Example 7 Cont.). SIV values of the formulae from Σ are as follows: $S_{\varphi}^{I_{MI}}(\Sigma) = \frac{5}{6}$, $S_{\neg\varphi}^{I_{MI}}(\Sigma) = \frac{1}{2}$, $S_{\varphi\to\psi}^{I_{MI}}(\Sigma) = \frac{1}{3}$, $S_{\neg\psi}^{I_{MI}}(\Sigma) = \frac{1}{3}$, and $S_{\omega}^{I_{MI}}(\Sigma) = 0$.

On the one hand, this measure takes into account the fact that a formula being in more minimal inconsistent sets is more inconsistent (which can be justified by saying that to obtain consistency, one has to remove at least one formula from every minimal conflict, thus by removing a formula which is in more minimal conflicts, one obtains consistency "faster"). On the other hand, this measure takes into account the intuition that, for example, a formula is in a minimal inconsistent set having 1000 formulae makes it "less inconsistent" than if it were in a minimal inconsistent sets having 2 formulae.

However, `MI` is just one possible basic inconsistency value, which we presented in order to illustrate the idea. In the rest of the paper, we suppose that an arbitrary basic inconsistency measure and the corresponding SIV are used.

4.2 Defining Instantiations

In this section, we use the method for measuring inconsistency of a formula to define an instantiation of Dung's abstract argumentation theory. Suppose that we are given a basic inconsistency measure. We can obtain the corresponding SIV, and use it to compare the formulae of the knowledge base. We first define how to construct a stratified version of a knowledge base, where the least inconsistent formulae (according to a given measure) are put in Σ_0 and the most inconsistent ones in Σ_n.

Definition 14 (Inconsistency ordered version of a knowledge base). *Let I be a basic inconsistency measure, and S^I the corresponding SIV. Let $\Sigma \subseteq \mathcal{L}$ be a knowledge base. The inconsistency ordered version of Σ (with respect to I) is a n-tuple $(\Sigma_0, \ldots, \Sigma_n)$ such that*

- $\Sigma_0 \cup \ldots \cup \Sigma_n = \Sigma$,
- *for every* $i, j \in \{0, \ldots, n\}$, *if* $i \neq j$ *then* $\Sigma_i \cap \Sigma_j = \emptyset$,
- *for any two formulae* $\varphi, \psi \in \Sigma$ *such that* $\varphi \in \Sigma_i$ *and* $\psi \in \Sigma_j$, *we have*

$$S^I_\varphi(\Sigma) \geq S^I_\psi(\Sigma) \quad \textit{if and only if} \quad i \geq j.$$

Example 9 (Example 8 Cont.). The inconsistency ordered version of Σ with respect to MI is: $\Sigma_0 = \{\omega\}$, $\Sigma_1 = \{\varphi \to \psi, \neg\psi\}$, $\Sigma_2 = \{\neg\varphi\}$, $\Sigma_3 = \{\varphi\}$.

This order induces a preference on Σ, which can be used to define a preference relation on $\mathtt{Arg}(\Sigma)$. Let us first define a level of a formula and of an argument.

Definition 15 (Level of formulae and arguments). *Let I be a basic inconsistency measure, S^I the corresponding SIV, let $\Sigma \subseteq \mathcal{L}$ be a knowledge base and $(\Sigma_0, \ldots, \Sigma_n)$ its inconsistency ordered version with respect to I. For a formula $\varphi \in \Sigma$,*

$$\mathtt{level}(\varphi) = i \textit{ if and only if } \varphi \in \Sigma_i.$$

For an argument $a \in \mathtt{Arg}(\Sigma)$,

$$\mathtt{level}(a) = max_{\varphi \in \mathtt{Supp}(a)} \mathtt{level}(\varphi).$$

We can now define an attack relation taking into account the level of formulae.

Definition 16 (Direct undercut on the ordered knowledge base). *Direct undercut on the ordered knowledge base $(\Sigma_0, \ldots, \Sigma_n)$ is a relation $\mathcal{R}_{duo}$ defined as: $a\mathcal{R}_{duo}b$ if and only if $(a\mathcal{R}_{du}b$ and $\mathtt{level}(a) \leq \mathtt{level}(b))$ or $(b\mathcal{R}_{du}a$ and $\mathtt{level}(a) < \mathtt{level}(b))$.*

As an illustration we consider again our running example.

Example 10 (Example 9 Cont.). Let $a = (\{\neg\psi, \varphi \to \psi\}, \neg\varphi)$, $b = (\{\varphi\}, \varphi)$, and $c = (\{\neg\varphi\}, \neg\varphi)$. Then, $a\mathcal{R}_{du}b$, $\mathtt{level}(a) = 1$ and $\mathtt{level}(b) = 3$. Thus, $a\mathcal{R}_{duo}b$. However, even if $b\mathcal{R}_{du}c$, we do not have that $b\mathcal{R}_{duo}c$, since $\mathtt{level}(b) = 3$ and $\mathtt{level}(c) = 2$.

Attack relation $\mathcal{R}_{duo}$ satisfies CD.

Proposition 4. *For any basic inconsistency measure I and the corresponding SIV S^I, $\mathcal{R}_{duo}$ is* CD.

We can also show that it returns consistent extensions which are closed for $\vdash$ and for sub-arguments[1].

Proposition 5. *Let I be a basic inconsistency measure and S^I the corresponding Shapley inconsistency measure. Let $\Sigma \subseteq \mathcal{L}$ be a knowledge base and $(\Sigma_0, \ldots, \Sigma_n)$ its inconsistency ordered version. Let $\mathcal{E}$ be a stable extension of $(\mathtt{Arg}(\Sigma), \mathcal{R}_{duo})$. Then:*

- $\mathtt{Base}(\mathcal{E})$ *and* $\mathtt{Concs}(\mathcal{E})$ *are consistent sets*
- $\mathtt{Concs}(\mathcal{E})$ *is closed for* $\vdash$, *i.e. for every* $\varphi \in \mathcal{L}$, *if* $\mathtt{Concs}(\mathcal{E}) \vdash \varphi$ *then* $\varphi \in \mathtt{Concs}(\mathcal{E})$,

[1] We suppose the definition of sub-argument by Gorogiannis and Hunter [9].

– *$\mathcal{E}$ is closed for sub-arguments, i.e. if $a \in \mathcal{E}$ and b is an argument such that* $\mathtt{Supp}(b) \subseteq \mathtt{Supp}(a)$*, then $b \in \mathcal{E}$.*

Note that by following the approach we describe in this section, one obtains a refinement of the approach returning extensions corresponding to the maximal consistent subsets of the knowledge base. Namely, if a basic inconsistency measure is used to order the knowledge base, and $\mathcal{R}_{duo}$ is then applied to calculate the extensions under stable semantics, every extension corresponds to exactly one maximal consistent subset of Σ, but there are some maximal consistent subsets of Σ which do not correspond to any extensions. Proposition 6 shows that for every extension, there exists a maximal consistent subset of Σ corresponding to that extension. Example 11 illustrates the fact that there can exist maximal consistent sets which do note correspond to any stable extensions.

Proposition 6. *Let I be a basic inconsistency measure and S^I the corresponding SIV. Let $\Sigma \subseteq \mathcal{L}$ be a knowledge base and $(\Sigma_0, \ldots, \Sigma_n)$ its inconsistency ordered version. Then:*

$$\mathtt{Ext}_s((\mathtt{Arg}(\Sigma), \mathcal{R}_{duo})) \subseteq \{\mathtt{Arg}(S) \mid S \in \mathtt{MC}(\Sigma)\}$$

Example 11 (Example 10 Cont.). The set $S = \{\varphi, \varphi \to \psi, \omega\}$ is a maximal consistent subset of Σ. Let $d = (\{\neg\psi\}, \neg\psi)$. It is clear that $d \notin \mathtt{Arg}(S)$. However, no argument from $\mathtt{Arg}(S)$ attacks d with respect to $\mathcal{R}_{duo}$. There exists only one argument $e = (\{\varphi, \varphi \to \psi\}, \psi)$, such that $e \in \mathtt{Arg}(S)$ and $e\mathcal{R}_{du}d$, but $\mathtt{level}(e) > \mathtt{level}(d)$, thus e is more inconsistent than d and, according to the definition of $\mathcal{R}_{duo}$, does not attack d.

We see from Propositions 4 and 5 that $\mathcal{R}_{duo}$ satisfies CD and CE$_s$. We now show that this attack relation falsifies AS and MS. To show that $\mathcal{R}_{duo}$ falsifies AS, consider the following example.

Example 12. Let $\Sigma = \{\neg\varphi, \neg(\varphi \wedge \psi), \varphi \wedge \psi\}$, and let us use the MI inconsistency measure and the corresponding Shapley Inconsistency Value $S^{I_{MI}}$. Then, $\Sigma_0 = \{\neg\varphi, \neg(\varphi \wedge \psi)\}$ and $\Sigma_1 = \{\varphi \wedge \psi\}$. Let $a = (\{\neg\varphi\}, \neg\varphi)$, $b = (\{\varphi \wedge \psi\}, \varphi)$, and $c = (\{\varphi \wedge \psi\}, \psi)$. Then, $\mathtt{Supp}(b) = \mathtt{Supp}(c)$, but at the same time $a\mathcal{R}_{duo}b$ and $\neg(a\mathcal{R}_{duo}c)$. Thus, $\mathcal{R}_{duo}$ does not satisfy AS.

By examining Example 10 one can observe that no argument attacks argument $c = (\{\neg\varphi\}, \neg\varphi)$ in this example. Thus, $\mathcal{R}_{duo}$ does not satisfy MS.

5 Summary

This paper advances the state of the art in instantiating Dung's abstract argumentation theory in several ways. First, we identify four simple conditions describing a wide class of attack relations based on attacking premises of an argument which return extensions corresponding to exactly maximal consistent subsets of the propositional knowledge base. Second, we show that when two of the conditions are dropped, it is possible to instantiate Dung's abstract argumentation theory with classical propositional logic and to obtain a result substantially different from the extensions which correspond to maximal consistent subsets of the knowledge base, without having external information such as preferences or values. We use Shapley Inconsistency Values [10] to measure inconsistency of a particular formula in the knowledge base and use that value to define attack relations which select extensions made of *less inconsistent* formulae. Third, we show that this whole class of instantiations satisfies the usual rationality postulates: its extensions have consistent bases, they are closed for sub-arguments, etc.

We identified a new class of inference relations that can be captured in Dung's theory, which is a first step towards a better understanding of possibilities and constraints imposed by this abstract theory. Our next goal is to characterise the class of all inference relations that can be represented in such a way.

To capture different results from simply returning the extensions corresponding to maximal consistent sets, we use an original attack relation, which has several features deserving some comments. First, this attack relation is dependent on the knowledge base Σ. In other words, whether an argument attacks another one cannot be determined without knowing what knowledge base they come from. This raises some conceptually and technically interesting questions which will be part of our future work. Second, the procedure we use rank-orders arguments on the basis of some kind of preference on the formulae in their supports. Our attack relation in some way "simulates" what is done in preference-based argumentation [2], and protects less inconsistent arguments from more inconsistent ones. An important difference is that in the present paper, we do not suppose any preferences at the input of our system. If the proposed class of instantiations selects some maximal consistent sets and not all of them, it comes from the fact that they have different degrees of inconsistency.

Obviously, the result of our work depends on the acceptability semantics used for evaluating arguments. Our main results were shown under stable semantics. We plan to examine whether similar results can be obtained under other semantics, and more generally, to determine the role played by a semantics when capturing different results as instantiations of Dung's abstract theory. Our goal is to study a large class of semantics satisfying some minimal requirements [4] (e.g. conflict-freeness, syntax independence).

This paper shows that the class of attack relations satisfying CE, CD, AS and MS is rather narrow, in the sense that they always return a result identical to that obtained from maximal consistent sets of the knowledge base. Thus, if one wants to subsume richer approaches, at least one of those four conditions has to be dropped. For example, Section 4 of the current paper uses attack relations satisfying CD and CE and violating AS and MS. First, note that we present the first attack relation which violates AS and returns sound results. Considering violating MS, it does not seem surprising, since this condition basically says that every maximal consistent set should yield a stable extension. Violating conflict-dependency and keeping some good properties of the system looks like a difficult task, although we do not claim that is impossible. However, it would be hard to justify attack relations returning extensions with inconsistent bases. The only possible explanation for that could be that argumentation is seen just as the first step of some longer process, and it resolves *some* (but not necessarily all) conflicts. Then, another mechanism is used to reason with the set of obtained extensions.

References

1. Amgoud, L., Besnard, P.: Bridging the Gap between Abstract Argumentation Systems and Logic. In: Godo, L., Pugliese, A. (eds.) SUM 2009. LNCS, vol. 5785, pp. 12–27. Springer, Heidelberg (2009)
2. Amgoud, L., Vesic, S.: A new approach for preference-based argumentation frameworks. Annals of Mathematics and Artificial Intelligence (2011)
3. Amgoud, L., Vesic, S.: On the Equivalence of Logic-Based Argumentation Systems. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 123–136. Springer, Heidelberg (2011)
4. Baroni, P., Giacomin, M.: On principle-based evaluation of extension-based argumentation semantics. Artificial Intelligence Journal 171, 675–700 (2007)
5. Caminada, M., Amgoud, L.: On the evaluation of argumentation formalisms. Artificial Intelligence Journal 171(5-6), 286–310 (2007)
6. Cayrol, C.: On the relation between argumentation and non-monotonic coherence-based entailment. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 1995), pp. 1443–1448 (1995)
7. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence Journal 77, 321–357 (1995)
8. Gabbay, D., Pigozzi, G., Rodrigues, O.: Common foundations for belief revision, belief merging and voting. In: Formal Models of Belief Change in Rational Agents (Dagstuhl Seminar Proceedings) (2007)
9. Gorogiannis, N., Hunter, A.: Instantiating abstract argumentation with classical logic arguments: Postulates and properties. Artificial Intelligence Journal 175, 1479–1497 (2011)
10. Hunter, A., Konieczny, S.: On the measure of conflicts: Shapley inconsistency values. Artificial Intelligence Journal 174(14), 1007–1026 (2010)
11. Priest, G.: Paraconsistent logic. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. 6, pp. 287–393. Kluwer Academic Publishers, Dordrecht (2002)
12. Shapley, L.: A value for n-person games. In: Contributions to the Theory of Games II, pp. 307–317. Princeton University Press (1953)

Reasoning about Agent Programs Using ATL-Like Logics

Nitin Yadav and Sebastian Sardina[⋆]

RMIT University, Melbourne, Australia

Abstract. We propose a variant of Alternating-time Temporal Logic (ATL) grounded in the agents' operational know-how, as defined by their libraries of abstract plans. Inspired by ATLES, a variant itself of ATL, it is possible in our logic to explicitly refer to "rational" strategies for agents developed under the Belief-Desire-Intention agent programming paradigm. This allows us to express and verify properties of BDI systems using ATL-type logical frameworks.

Keywords: Agent Programming, Reactive plans, ATL, Model Checking.

1 Introduction

The formal verification of agent-oriented programs requires logic frameworks capable of representing and reasoning about agents' abilities and capabilities, and the goals they can feasibly achieve. In particular, we are interested here in programs written in the family of Belief-Desire-Intention (BDI) agent programming systems [5, 6, 18], a popular paradigm for building multi-agent systems. Traditional BDI logics based on CTL (e.g., [17]) are generally too weak for representing ability; their success has primarily been in defining "rationality postulates," i.e., constraints on rational behaviour. Further, such logics do not encode agents' capabilities (as represented by their plan libraries) and thereby leave a sizable gap between agent programs and their formal verification.

Recent work (e.g., [1, 2, 9]) has better bridged the gap between formal logic and practical programming by providing an axiomatisation of a class of models that is designed to closely model a programming framework. However, this is done by restricting the logic's models to those that satisfy the transition relations of agents' plans, as defined by the semantics of the programming language itself. In such a framework, it is not possible to reason about the agent's know-how and what the agent could achieve *if it had* specific capabilities. It is also not possible to reason about coalition of agents.

Our aim thus is to define a framework, together with model checking techniques, that will allow us to speculate about a group of agents' capabilities and what they can achieve with such capabilities under the BDI paradigm, which enables abstract plans written by programmers to be combined and used in real-time under the principles of

This requires the ability to represent capabilities directly in our logic. To that end, we adapt ATLES, a version of ATL (Alternating-time Temporal Logic) [3] with Explicit Strategies [20], to our purpose. ATL is a logic for reasoning about the ability of

[⋆] We would like to thank Lawrence Cavedon for earlier discussions on the topic of this paper. We acknowledge the support of the Australian Research Council under grant DP120100332.

agent coalitions in *multi-player game structures*. This is achieved by reasoning about strategies (and their success) employed by teams of agents: $\langle\langle A\rangle\rangle\varphi$ expresses that the coalition team of agents A has a joint strategy for guaranteeing that the temporal property φ holds. Walther et al. [20], standard ATL does not allow agents' strategies to be explicitly represented in the syntax of the logic. They thus rectified this shortcoming by defining ATLES, which extends ATL by allowing strategy terms in the language: $\langle\langle A\rangle\rangle_{\rho}\varphi$ holds if coalition A has a joint strategy for ensuring φ, when some agents are committed to *specific* strategies as specified by so-called commitment function ρ.

In this paper, we go further and develop a framework—called BDI-ATLES—in which the strategy terms are tied directly to the plans available to agents under the notion of practical reasoning embodied by the BDI paradigm [6, 18]: *the only strategies that can be employed by a BDI agent are those that ensue by the (rational) execution of its predefined plans, given its goals and beliefs*. The key construct $\langle\langle A\rangle\rangle_{\omega,\varrho}\varphi$ in the new framework states that coalition A has a joint strategy for ensuring φ, *under the assumptions that some agents in the system are BDI-style agents* with capabilities and goals as specified by assignments ω and ϱ, respectively. For instance, in the Gold Mining domain from the International Agent Contest,[1] one may want to verify if two miner agents programmed in a BDI language can successfully collect gold pieces when equipped with navigation and communication capabilities and want to win the game, while the opponent agents can perform any physically legal action. More interesting, a formula like $\langle\langle A\rangle\rangle_{\emptyset,\emptyset}\varphi \supset \langle\langle A\rangle\rangle_{\omega,\varrho}\varphi$ can be used to check whether coalition A has enough know-how and motivations to carry out a task φ that is indeed physically feasible for the coalition.

We observe that the notion of "rationality" used in this work is that found in the literature on BDI and agent programming, rather than that common in game-theory (generally captured via *solution concepts*). As such, rationality shall refer from now on to reasonable constraints on how the various mental modalities—e.g., beliefs, intention, goals—may interact. In particular, we focus on the constraint that agents select actions from their know-how in order to achieve their goals in the context of their beliefs.

Finally, we stress that this work aims to contribute to the agent-oriented programing community more than to the (ATL) verification one. Indeed, our aim is to motivate the former to adopt well-established techniques in game-theory for the effective verification of their "reactive" style agent programs.

2 Preliminaries

2.1 ATL/ATLES Logics of Coalitions

Alternating-time Temporal Logic (ATL) [3] is a logic for reasoning about the ability of agent coalitions in *multi-agent game structures*. ATL formulae are built by combining propositional formulas, the usual temporal operators—namely, $\bigcirc$ ("in the next state"), $\Box$ ("always"), $\Diamond$ ("eventually"), and $\mathcal{U}$ ("strict until")—and a *coalition path quantifier* $\langle\langle A\rangle\rangle$ taking a set of agents A as parameter. As in CTL, which ATL extends, temporal operators and path quantifiers are required to alternate. Intuitively, an ATL formula $\langle\langle A\rangle\rangle\phi$, where A is a set of agents, holds in an ATL structure if by suitably choosing

[1] `http://www.multiagentcontest.org/`

their moves, the agents in A *can force ϕ true*, no matter how other agents happen to move. The semantics of ATL is defined in so-called *concurrent game structures* where, at each point, all agents simultaneously choose their moves from a finite set, and the next state deterministically depends on such choices. More concretely, an ATL structure is a tuple $\mathcal{M} = \langle \mathcal{A}, Q, \mathcal{P}, Act, d, \mathcal{V}, \sigma \rangle$, where $\mathcal{A} = \{1, \ldots, k\}$ is a finite set of agents, Q is the finite set of states, $\mathcal{P}$ is the finite set of propositions, Act is the set of all domain actions, $d : \mathcal{A} \times Q \mapsto 2^{Act}$ indicates all available actions for an agent in a state, $\mathcal{V} : Q \mapsto 2^{\mathcal{P}}$ is the valuation function stating what is true in each state, and $\sigma : Q \times Act^{|\mathcal{A}|} \mapsto Q$ is the transition function mapping a state q and a joint-move $a \in \mathcal{D}(q)$—where $\mathcal{D}(q) = \times_{i=1}^{|\mathcal{A}|} d(i, q)$ is the set of legal joint-moves in q —to the resulting next state q'.

A *path* $\lambda = q_0 q_1 \cdots$ in a structure $\mathcal{M}$ is a, possibly infinite, sequence of states such that for each $i \geq 0$, there exists a joint-move $\boldsymbol{a_i} \in \mathcal{D}(q_i)$ for which $\sigma(q_i, \boldsymbol{a_i}) = q_{i+1}$. We use $\lambda[i] = q_i$ to denote the i-th state of λ, Λ to denote the set of all paths in $\mathcal{M}$, and $\Lambda(q)$ to denote those starting in q. Also, $|\lambda|$ denotes the length of λ as the number of state transitions in λ: $|\lambda| = \ell$ if $\lambda = q_0 q_1 \ldots q_\ell$, and $|\lambda| = \infty$ if λ is infinite. When $0 \leq i \leq j \leq |\lambda|$, then $\lambda[i, j] = q_i q_{i+1} \ldots q_j$ is the finite subpath between the i-th and j-th steps in λ. Finally, a *computation path* in $\mathcal{M}$ is an infinite path in Λ.

To provide semantics to formulas $\langle\langle \cdot \rangle\rangle \varphi$, ATL relies on the notion of agent strategies. Technically, an ATL *strategy* for an agent agt is a function $f_{agt} : Q^+ \mapsto Act$, where $f_{agt}(\lambda q) \in d(agt, q)$ for all $\lambda q \in Q^+$, stating a particular action choice of agent agt at path λq. A *collective strategy* for group of agents $A \subseteq \mathcal{A}$ is a set of strategies $F_A = \{f_{agt} \mid agt \in \mathcal{A}\}$ providing one specific strategy for each agent $agt \in A$. For a collective strategy F_A and an initial state q, it is not difficult to define the set $out(q, F_A)$ of all *possible outcomes* of F_A starting at state q as the set of all computation paths that may ensue when the agents in A behave as prescribed by F_A, and the remaining agents follow any arbitrary strategy [3, 20]. The semantics for the coalition modality is then defined as follows (here ϕ is a *path formula*, that is, it is preceded by $\bigcirc$, $\Box$, or $\mathcal{U}$, and $\mathcal{M}, \lambda \models \phi$ is defined in the usual way [3]):

$$\mathcal{M}, q \models \langle\langle A \rangle\rangle \phi \textit{ iff } \text{there is a collective strategy } F_A \text{ such that for all computations } \lambda \in out(q, F_A), \text{ we have } \mathcal{M}, \lambda \models \phi.$$

The coalition modality only allows for implicit (existential) quantification over strategies. In some contexts, though, it is important to refer to strategies explicitly in the language, e.g., can a player win the game if the opponent plays a specified strategy? To address this limitation, Walther et al. [20] proposed ATLES, an extension of ATL where the coalition modality is extended to $\langle\langle A \rangle\rangle_\rho$, where ρ is a *commitment function*, that is, a partial function mapping agents to so-called *strategy terms*. Formula $\langle\langle A \rangle\rangle_\rho \phi$ thus means that *"while the agents in the domain of ρ act according to their commitments, the coalition A can cooperate to ensure ϕ as an outcome."*

The motivation for our work stems from the fact that ATLES is agnostic on the source of the strategic terms: all meaningful strategies have already been identified. In the context of multi-agent systems, it may not be an easy task to identify those strategies compatible with the agents' behaviors, as those systems are generally built using programming frameworks [5] that are very different from ATL(ES).

2.2 BDI Programming

The BDI agent-oriented programming paradigm is a popular and successful approach for building agent systems, with roots in philosophical work on rational action [6] and a plethora of programming languages and systems available, such as JACK, JASON, JADEX, 2APL [5], and GOAL [11], among others.

A typical BDI agent continually tries to achieve its goals (or desires) by selecting an adequate plan from its *plan library* given its current beliefs, and placing it into the *intention base* for execution. The agent's plan library Π encodes the standard operational knowledge of the domain by means of a set of *plan-rules* (or "recipes") of the form $\phi[\alpha]\psi$: *plan α is a reasonable plan to adopt for achieving ψ when (context) condition ϕ is believed true*. For example, walking towards location x from y is a reasonable strategy, if there is a short distance between x and y (and the agent wants to be eventually at location x). Conditions ϕ and ψ are (propositional) formulas talking about the current and goal states, respectively. Though different BDI languages offer different constructs for crafting plans, most allow for sequences of domain actions that are meant to be directly executed in the world (e.g., lifting an aircraft's flaps), and the posting of (intermediate) *sub-goals* $!\varphi$ (e.g., obtain landing permission) to be resolved. The intention base, in turn, contains the current, partially executed, plans that the agent has already *committed to* for achieving certain goals. Current intentions being executed provide a screen of admissibility for attention focus [6].

Though we do not present it here for lack of space, most BDI-style programming languages come with a clear single-step semantics basically realizing [18]'s execution model in which *(rational) behavior arises due to the execution of plans from the agent's plan library so as to achieve certain goals relative to the agent's beliefs*.

3 BDI-ATLES: ATL for BDI Agents

Here we develop an ATL(ES)-like logic that bridges the gap between verification frameworks and BDI agent-oriented programming languages. The overarching idea is for BDI programmers to be able to encode BDI applications in ATL in a principled manner.

Recall that ATL(ES) uses strategies to denote the agent's choices among possible actions. For a BDI agent these strategies are *implicit* in her know-how. In particular, we envision BDI agents defined with a set of *goals* and so-called *capabilities* [7, 16]. Generally speaking, a capability is a set/module of procedural knowledge (i.e., plans) for some functional requirement. An agent may have, for instance, the Navigate capability encoding all plans for navigating an environment. Equipped with a set of capabilities, a BDI agent executes actions as per plans available so as to achieve her goals, e.g., exploring the environment. In this context, the BDI developer is then interested in what agents can achieve at the level of goals and capabilities. Inspired by ATLES, we develop a logic that caters for this requirement without departing much from the ATL framework.

In this work, we shall consider plans consisting of single actions, that is, given BDI plan for the form $\phi[\alpha]\psi$, the body of the plan α consists of one primitive action. Such plans are akin to those in the GOAL agent programming language [11], as well as universal-plans [19], and reactive control modules [4]. Let $\boldsymbol{\Pi}^{\mathcal{P}}_{Act}$ be the (infinite) set of all possible plan-rules given a set of actions *Act* and a set of domain propositions $\mathcal{P}$.

3.1 BDI-ATLES Syntax

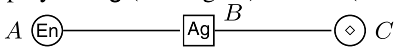
The language of BDI-ATLES is defined over a finite set of atomic propositions $\mathcal{P}$, a finite set of agents $\mathcal{A}$, and a finite set of capability terms $\mathcal{C}$ available in the BDI application of concern. Intuitively, each capability term $c \in \mathcal{C}$ (e.g., Navigate) stands for a plan library Π^c (e.g., Π^{Navigate}). As usual, a *coalition* is a set $A \subseteq \mathcal{A}$ of agents. A *capability assignment* ω consists of a set of pairs of agents with their capabilities of the form $\langle agt : C_{agt} \rangle$, where $agt \in \mathcal{A}$ and $C_{agt} \subseteq \mathcal{C}$. A *goal assignment* ϱ, in turn, defines the goal base (i.e., set of propositional formulas) for some agents, and is a set of tuples of the form $\langle agt : G_{agt} \rangle$, where $agt \in \mathcal{A}$ and G_{agt} is a set of boolean formulas over $\mathcal{P}$. We use $\mathcal{A}_\omega$ to denote the set of agents for which their capabilities are defined by assignment ω, that is, $\mathcal{A}_\omega = \{agt \mid \langle agt : C_{agt} \rangle \in \omega\}$. Set $\mathcal{A}_\varrho$ is defined analogously.

The set of BDI-ATLES formulas is then exactly like that of ATL(ES), except that coalition formulas are now of the form $\langle\!\langle A \rangle\!\rangle_{\omega,\varrho}\varphi$, where φ is a path formula (i.e., it is preceded by $\bigcirc$, $\Box$, or $\mathcal{U}$), A is a coalition, and ω and ϱ range over capability and goal assignments, respectively, such that $\mathcal{A}_\omega = \mathcal{A}_\varrho$. Its intended meaning is as follows:

> $\langle\!\langle A \rangle\!\rangle_{\omega,\varrho}\varphi$ expresses that coalition of agents A can jointly force temporal condition φ to hold when BDI agents in $\mathcal{A}_\omega$ (or $\mathcal{A}_\varrho$, since $\mathcal{A}_\varrho = \mathcal{A}_\omega$) are equipped with capabilities as per assignment ω and (initial) goals are per assignment ϱ.

Notice that we require, in each coalition (sub)formula, that the agents for which capabilities and goals are assigned to be the same. This enforces the constraint that BDI-style agents have *both* plans and goals. Hence, a formula of the form $\langle\!\langle A \rangle\!\rangle_{\emptyset,\{\langle a_1:\{\gamma\}\rangle\}}\varphi$ would not be valid, as agent a_1 has one goal (namely, to bring about γ), but its set of plans is not defined—we cannot specify what its rational behavior may be. This contrasts with formula $\langle\!\langle A \rangle\!\rangle_{\{\langle a_1:\emptyset\rangle\},\{\langle a_1:\{\gamma\}\rangle\}}\varphi$, a valid formula in which agent a_1 is assumed to have no plans (i.e., agent has empty know-how) and one goal.

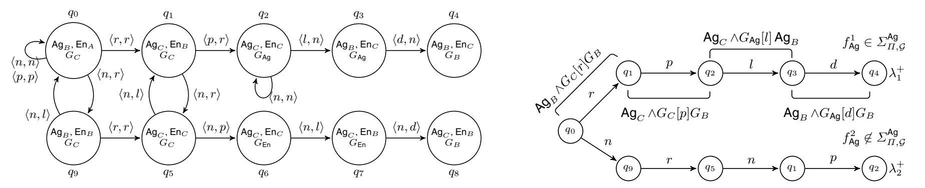
Example 1. Consider the following simplified instance of the gold mining domain with three locations A, B and C, a gold piece $\diamond$ at location C, the depot located at B (rectangle location), and two players Ag (BDI agent) and En (enemy):



Players can move LEFT/RIGHT, PICK/DROP gold, or remain still by executing special action NOOP. Proposition X_Y, where $X \in \{\mathsf{Ag}, \mathsf{En}\}$ and $Y \in \{A, B, C\}$, encodes that player X is at location Y; whereas propositions G_A, G_B, G_C, G_{Ag}, and G_{En} denote that the gold is at location A/B/C or being held by agent Ag/En, respectively. The depot is assumed to be always at B and hence is not represented explicitly.

The winning condition for player Ag is $\psi_{\mathit{WIN}} = G_B \wedge \mathsf{Ag}_B$: the player wins when collocated with gold at the depot.

Among the many capabilities available encoding the know-how information of the domain, we consider the following three. The Collect capability includes plans to pick gold, such as $\mathsf{Ag}_C \wedge G_C[\text{PICK}]G_B$: if gold needs to be at B and agent is at C, where there is indeed gold, then execute the PICK action. Similarly, capability Deposit contains plans like $G_{\mathsf{Ag}} \wedge \mathsf{Ag}_B[\text{DROP}]G_B$, for example, to allow dropping of gold at the desired location. Lastly, capability Navigate has plans for moving around, such as $\mathsf{Ag}_C[\text{LEFT}]\,\mathsf{Ag}_B$ to move left from location C to (desired destination) B. $\Box$


q_0 q_1 q_2 q_3 q_4
$\mathsf{Ag}_B, \mathsf{En}_A$ G_C
$\mathsf{Ag}_C, \mathsf{En}_B$ G_C
$\mathsf{Ag}_C, \mathsf{En}_C$ G_{Ag}
$\mathsf{Ag}_B, \mathsf{En}_C$ G_{Ag}
$\mathsf{Ag}_B, \mathsf{En}_C$ G_B
$\langle r,r\rangle$ $\langle p,r\rangle$ $\langle l,n\rangle$ $\langle d,n\rangle$
$\langle n,n\rangle$ $\langle p,p\rangle$ $\langle n,r\rangle$ $\langle n,l\rangle$ $\langle n,r\rangle$ $\langle n,n\rangle$ $\langle n,l\rangle$
$\mathsf{Ag}_B, \mathsf{En}_B$ G_C
$\mathsf{Ag}_C, \mathsf{En}_C$ G_C
$\mathsf{Ag}_C, \mathsf{En}_C$ G_{En}
$\mathsf{Ag}_C, \mathsf{En}_B$ G_{En}
$\mathsf{Ag}_C, \mathsf{En}_B$ G_B
$\langle r,r\rangle$ $\langle n,p\rangle$ $\langle n,l\rangle$ $\langle n,d\rangle$
q_9 q_5 q_6 q_7 q_8
$\mathsf{Ag}_C \wedge G_{\mathsf{Ag}}[l]\, \mathsf{Ag}_B$
$f^1_{\mathsf{Ag}} \in \Sigma^{\mathsf{Ag}}_{\Pi,\mathcal{G}}$
$\mathsf{Ag}_B \wedge G_C[r] G_B$
q_1 p q_2 l q_3 d q_4 λ_1^+
r
$\mathsf{Ag}_C \wedge G_C[p] G_B$
$\mathsf{Ag}_B \wedge G_{\mathsf{Ag}}[d] G_B$
q_0 n
$f^2_{\mathsf{Ag}} \notin \Sigma^{\mathsf{Ag}}_{\Pi,\mathcal{G}}$
q_9 r q_5 n q_1 p q_2 λ_2^+

(a) A section of the BDI-ATLES alternating model (b) Traces λ_1^+ and λ_2^+ resultant from strategies f^1_{Ag} and f^2_{Ag}, respectively

Fig. 1. A fragment of a Gold domain model and a picture showing rational traces and strategies. Actions LEFT, RIGHT, PICK, DROP, and NOOP are abbreviated with their first letter.

The remaining of the section involves providing the right interpretation to such formulas, under the assumption that agents act rationally as per the BDI paradigm.

3.2 BDI-ATLES Semantics

A BDI-ATLES concurrent game structure is a tuple $\mathcal{M} = \langle \mathcal{A}, Q, \mathcal{P}, Act, d, \mathcal{V}, \sigma, \Theta \rangle$, with:

- $\mathcal{A}$, Q, $\mathcal{P}$, Act, d, $\mathcal{V}$ and σ are as in ATL(ES).
- There is a distinguished dummy action NOOP $\in Act$ such that NOOP $\in d_{agt}(q)$ and $\sigma(q, \langle \text{NOOP}, \dots, \text{NOOP} \rangle) = q$, for all $agt \in \mathcal{A}$ and $q \in Q$, that is, NOOP is always available to all agents and the system remains still when all agents perform it.
- Capability function $\Theta : \mathcal{C} \mapsto \mathcal{F}(\boldsymbol{\Pi}^{\mathcal{P}}_{Act})$ maps capability terms to their (finite) set of plans. (Here, $\mathcal{F}(X)$ denotes the set of all finite subsets X.)

Example 2. Figure 1(a) shows a partial model for the gold game. The game starts at state q_0, with players Ag and En located at B and A, resp., and gold present at C. From there, player Ag has a winning strategy: reach the gold earlier and deposit it in the depot. This can be seen in path $q_0q_1q_2q_3q_4$. However, this is possible only when the agent Ag is indeed equipped with all three capabilities. If, on the other hand, the agent lacks capability Collect, for instance, then player En may actually manage to win the game, as evident from the path $q_0q_1q_5q_6q_7q_8$. □

BDI-ATLES models are similar to ATLES ones, except that capability, rather than strategy term, interpretations are used. In a nutshell, the challenge thus is to characterize what the underlying "low-level" ATL strategies for agents with certain capabilities and goals are. We call such strategies *rational strategies*, in that they are compatible with the standard BDI rational execution model [18]: *they represent the agent acting as per her available plans in order to achieve her goals in the context of her beliefs.*

So, given an agent $agt \in \mathcal{A}$, a plan-library Π, and a goal base $\mathcal{G}$, we define $\Sigma^{agt}_{\Pi,\mathcal{G}}$ to be the set of standard ATL strategies for agent agt in $\mathcal{M}$ that are rational strategies when the agent is equipped with plan-library Π and has $\mathcal{G}$ as (initial) goals, that is, those ATL strategies in which the agent always chooses an action that is directed by one of its available plans in order to achieve one of its goals in the context of its current beliefs. The core idea behind defining set $\Sigma^{agt}_{\Pi,\mathcal{G}}$ is to identify those "rational traces" in the structure that are compatible with the BDI deliberation process in which the agent acts as per her goals and beliefs. Traces just generalize paths to account for the actions performed at each step, and are hence of the form $\lambda^+ = q_0\, a_1\, q_1 \cdots a_\ell\, q_\ell$ such that $q_0 q_q \cdots q_\ell$ is a (finite) path. Rational strategies, then, are those that only yield rational traces. Technically, we define *rational traces* in three steps. First, we define a *goal-marking* function $g(\lambda^+, i)$ denoting the "active" goal base of the agent at the i-th stage of trace λ^+. Basically, a goal-marking function keeps track of the goals that the agent has already achieved at each stage in a trace. Second, we define $\mathit{Exec}(\phi[\alpha]\psi, g, \lambda^+)$ as the set of indexes (i.e., stages) in trace λ^+ where the plan $\phi[\alpha]\psi$ may have been executed by the agent: the plan's precondition ϕ was true, ψ was an active goal of the agent (as directed by goal-marking function g), and α was indeed performed. Finally, we say a trace λ^+ is deemed "rational" if at every moment in the run the agent executed one of its plans. That is, for every index i, it is the case that $i \in \mathit{Exec}_{agt}(\phi[\alpha]\psi, g, \lambda^+)$, for some plan $\phi[\alpha]\psi$ in her know-how library. Finally, we use $\Sigma^{agt}_{\Pi,\mathcal{G}}$ to denote the set of all ATL strategies whose executions always yield rational traces. The laborious, and arguably boring, technical details of all the above steps and notions can be found in [21].

Example 3. Figure 1(b) depicts two possible traces λ_1^+ and λ_2^+ (for agent Ag) compatible with strategies f^1_{Ag} and f^2_{Ag}, resp. Trace λ_1^+ is due to the agent executing actions as per its applicable plans, as evident from the plan labeling. For example, at state q_1, the agent is in a gold location and hence executes the pick action as per plan $\mathsf{Ag}_C \wedge G_C[\textsc{pick}]G_B$. Consequently the strategy f^1_{Ag} is rational, as it yields rational trace λ_1^+. Trace λ_2^+ on the other hand does not obey the BDI rationality constraints (e.g., the agent remains still in location B, despite an applicable plan being available). □

Assuming that set $\Sigma^{agt}_{\Pi,\mathcal{G}}$ of rational strategies has been suitably defined, we are ready to detail the semantics for formulas of the form $\langle\langle A\rangle\rangle_{\omega,\varrho}\varphi$. Following ATLES we first extend the notion of a joint strategy for a coalition to that of joint strategy *under a given capability and goal assignment*. So, given a capability (goal) assignment ω (ϱ) and an agent $agt \in \mathcal{A}_\omega$ ($agt \in \mathcal{A}_\varrho$), we denote agt's capabilities (goals) under ω (ϱ) by $\omega[agt]$ ($\varrho[agt]$). Intuitively, an *$\langle\omega, \varrho\rangle$-strategy for coalition A* is a joint strategy for A such that *(i)* agents in $A \cap \mathcal{A}_\omega$ only follow "rational" (plan-goal compatible) strategies as per their ω-capabilities and ϱ-goals; and *(b)* agents in $A \backslash \mathcal{A}_\omega$ follow arbitrary strategies. Formally, an $\underline{\langle\omega, \varrho\rangle\textit{-strategy for coalition } A}$ (with $\mathcal{A}_\omega = \mathcal{A}_\varrho$) is a collective strategy F_A for agents A such that for all $f_{agt} \in F_A$ with $agt \in A \cap \mathcal{A}_\omega$, it is the case that $f_{agt} \in \Sigma^{agt}_{\Pi,\mathcal{G}}$, where $\Pi = \cup_{c \in \omega[agt]}\Theta(c)$ and $\mathcal{G} = \varrho[agt]$. Note no requirements are asked on the strategies for the remaining agents $A \backslash \mathcal{A}_\omega$, besides of course being legal (ATL) strategies. Also, whereas ATLES ρ-strategies are collective strategies including *all* agents in the domain of commitment function ρ, our $\langle\omega, \varrho\rangle$-strategies are collective strategies for the coalition of concern only. This is because commitment functions induce deterministic agent behaviors, whereas capabilities and goals assignments induce non-deterministic ones. We will elaborate on this issue below.

Using the notions of $\langle\omega, \varrho\rangle$-strategies and that of possible outcomes for a given collective strategy from ATL (refer to function $\mathit{out}(\cdot,\cdot)$ from Preliminaries), we are now able to state the meaning of BDI-ATLES (coalition) formulas:[2]

[2] As with ATL(ES), φ ought to be a path formula and is interpreted in the usual manner. We omit the other ATL-like cases for brevity; see [20].

$\mathcal{M}, q \models \langle\langle A \rangle\rangle_{\omega,\varrho}\varphi$ *iff* there is a $\langle \omega, \varrho \rangle$-strategy F_A such that for all $\langle \omega, \varrho \rangle$-strategies $F_{\mathcal{A}_\omega \setminus A}$ for $\mathcal{A}_\omega \setminus A$, it is the case that $\mathcal{M}, \lambda \models \varphi$, for all paths $\lambda \in \mathit{out}(q, F_A \cup F_{\mathcal{A}_\omega \setminus A})$.

Intuitively, F_A stands for the collective strategy of agents A guaranteeing the satisfaction of formula φ. Because F_A is a $\langle \omega, \varrho \rangle$-strategy, some agents in A—those whose capabilities and goals are defined by ω and ϱ, resp.—are to follow strategies that correspond to rational executions of its capabilities. At the same time, because other agents outside the coalition could have also been assigned capabilities and goals, the chosen collective strategy F_A needs to work no matter how such agents (namely, agents $\mathcal{A}_\omega \setminus A$) behave, as long as they do it rationally given their plans and goals. That is, F_A has to work with *any* rational collective strategy $F_{\mathcal{A}_\omega \setminus A}$. Finally, the behavior of all remaining agents—namely those in $\mathcal{A} \setminus (A \cup \mathcal{A}_\omega)$—are taken into account when considering all possible outcomes, after all strategies for agents in $A \cup \mathcal{A}_\omega$ have been settled.

While similar to ATLES coalition formulas $\langle\langle A \rangle\rangle_\rho\varphi$, BDI-ATLES coalition formulas $\langle\langle A \rangle\rangle_{\omega,\varrho}\varphi$ differ in one important aspect that makes its semantics more involved. Specifically, whereas commitment functions ρ prescribe *deterministic* behaviors for agents, capabilities and goals assignments yield multiple potential behaviors for the agents of interest. This nondeterministic behavior stems from the fact that BDI agents can choose what goals to work on at each point and what available plans to use for achieving such goals. Technically, this is reflected in the strategies for each agent in $(\mathcal{A}_\omega \setminus A)$—those agents with assigned capabilities and goals but not part of the coalition—cannot be (existentially) considered together with those of agents in A or (universally) accounted for via the possible outcomes function $\mathit{out}(\cdot, \cdot)$, as such function puts no rationality constraints on the remaining (non-committed) agents. Thus, whereas agents in $A \cap \mathcal{A}_\omega$ are allowed to select one possible rational behavior, all rational behaviors for agents in $(A_\omega \setminus A)$ need to be taken into consideration.

We close this section by noting an important, and expected, monotonicity property of BDI-ATLES w.r.t. changes in the goals and plans of agents.

Proposition 1. $\models \langle\langle A \rangle\rangle_{\omega,\varrho}\varphi \supset \langle\langle A' \rangle\rangle_{\omega',\varrho'}\varphi$ *holds, provided that:*

- $A \subseteq A'$*, that is, the coalition is not reduced;*
- $\omega[agt] \subseteq \omega'[agt]$ *and* $\varrho[agt] \subseteq \varrho'[agt]$*, for all* $agt \in \mathcal{A}_\omega \cap A$*, that is, the goals and capabilities of those BDI agents in the coalition are not reduced; and*
- $\mathcal{A}_\omega \setminus A \subseteq \mathcal{A}_{\omega'} \setminus A'$*, that is, the set of non BDI agents outside the coalition is not reduced (but could be new BDI agents outside the coalition);*
- $\omega'[agt] \subseteq \omega[agt]$ *and* $\varrho'[agt] \subseteq \varrho[agt]$*, for all* $agt \in \mathcal{A}_\omega \setminus A$*, that is, the goals and capabilities of those BDI agents outside the coalition are not augmented.*

Informally, augmenting the goals/plans of agents in a coalition does not reduce the ability of agents. This is because a collective $\langle \omega, \varrho \rangle$-strategy for coalition A to bring about a formula would still work if more goals and plans are given to the agents in the coalition (second condition). Observe, on the other hand, that augmenting the goals or plans of those agents outside the coalition may yield new behavior that can indeed interfere with the coalition's original abilities (last condition). This even includes turning BDI agents into non BDI agents (third condition). Of course, as in ATL, enlarging the coalition does not reduce ability (first condition).

```
**foreach** $\varphi'$ *in Sub($\varphi$) w.r.t.* $\mathcal{M} = \langle \mathcal{A}, Q, \mathcal{P}, Act, d, \mathcal{V}, \sigma, \Theta \rangle$ **do**
    **case** $\varphi' = p : [\varphi']_{\mathcal{M}} = \mathcal{V}(p)$;
    **case** $\varphi' = \neg\theta : [\varphi']_{\mathcal{M}} = ([\text{TRUE}]_{\mathcal{M}} \setminus [\theta]_{\mathcal{M}})$;
    **case** $\varphi' = \theta_1 \vee \theta_2 : [\varphi']_{\mathcal{M}} = [\theta_1]_{\mathcal{M}} \cup [\theta_2]_{\mathcal{M}}$;
    **case** $\varphi' = \langle\langle A \rangle\rangle_{\omega,\varrho} \bigcirc \theta :\ [\varphi']_{\mathcal{M}} = ws(Pre(A, \omega, \Theta, [\theta]_{\mathcal{M}_\varrho}) \cap [\![\varrho]\!])$ ;
    **case** $\varphi' = \langle\langle A \rangle\rangle_{\omega,\varrho} \Box \theta :\ \rho = [\text{TRUE}]_{\mathcal{M}_\varrho}; \tau = [\theta]_{\mathcal{M}_\varrho}$ ;
      **while** $\rho \not\subseteq \tau$ **do** $\rho = \tau;\ \tau = Pre(A, \omega, \Theta, \rho) \cap [\theta]_{\mathcal{M}_\varrho}$ **od;**
      $[\varphi']_{\mathcal{M}} = ws(\rho \cap [\![\varrho]\!])$ ;
    **case** $\varphi' = \langle\langle A \rangle\rangle_{\omega,\varrho} \theta_1 \mathcal{U} \theta_2 :\ \rho = [\text{FALSE}]_{\mathcal{M}_\varrho}; \tau = [\theta_2]_{\mathcal{M}_\varrho}$ ;
      **while** $\tau \not\subseteq \rho$ **do** $\rho = \rho \cup \tau;\ \tau = Pre(A, \omega, \Theta, \rho) \cap [\theta_1]_{\mathcal{M}_\varrho}$ **od;**
      $[\varphi']_{\mathcal{M}} = ws(\rho \cap [\![\varrho]\!])$ ;
**od;**
**return** $[\varphi']_{\mathcal{M}}$;
```

Fig. 2. BDI-ATLES symbolic model checking

4 BDI-ATLES Model Checking

Given a BDI-ATLES model $\mathcal{M}$ and a formula φ, the model checking algorithm for BDI-ATLES computes the set of states in $\mathcal{M}$ that satisfy φ. To that end, the algorithm has to take into account the rational choices of each BDI agent, that is, those choices that are the consequence of the agent's goals and capabilities specified by functions ϱ and ω in formulae of the form $\langle\langle A \rangle\rangle_{\omega,\varrho}\varphi$. Roughly speaking, the algorithm restricts, at each step, the options of BDI agents to their applicable plans. We start by extending the model $\mathcal{M}$ to embed the possible goals (based on the goal assignment) of BDI agents into each state, and then then discuss the model checking algorithm and its complexity.

So, given a BDI-ATLES model $\mathcal{M}=\langle \mathcal{A}, Q, \mathcal{P}, Act, d, \mathcal{V}, \sigma, \Theta \rangle$ and a goal assignment ϱ, the <u>*goal-extended model*</u> is a model $\mathcal{M}_\varrho=\langle \mathcal{A}, Q_\varrho, \mathcal{P}, Act, d_\varrho, \mathcal{V}_\varrho, \sigma_\varrho, \Theta \rangle$, where:

– $Q_\varrho \subseteq Q \times \prod_{agt \in \mathcal{A}_\varrho} 2^{\varrho[agt]}$ is the set of extended states, now accounting for the possible goals of BDI agents. When $q_\varrho = \langle q, g_1, \dots, g_{|\mathcal{A}_\varrho|} \rangle \in Q_\varrho$, where $q \in Q$ and $g_i \subseteq \varrho[agt_i]$, is an extended state, we use $ws(q_\varrho) = q$ and $gl(agt_i, q_\varrho) = g_i$ to project $\mathcal{M}$'s world state and agt_i's goals. To enforce belief-goal consistency we require no agent ever wants something already true: there are no $q_\varrho \in Q_\varrho$, $agt \in \mathcal{A}_\varrho$, and formula γ such that $\mathcal{V}(ws(q_\varrho)) \models \gamma$ and $\gamma \in gl(agt, q_\varrho)$.
– $\mathcal{V}_\varrho(q_\varrho) = \mathcal{V}(ws(q_\varrho))$, for all $q_\varrho \in Q_\varrho$, that is, state evaluation remains unchanged.
– $d_\varrho(agt, q_\varrho) = d(agt, ws(q_\varrho))$, that is, physical executability remains unchanged.
– $\sigma_\varrho(q_\varrho, a) = \langle q', g'_1, \dots, g'_{|\mathcal{A}_\varrho|} \rangle$, where $q' = \sigma(ws(q_\varrho), a)$ and $g'_i = gl(agt_i, q_\varrho) \setminus \{\gamma \mid \gamma \in gl(agt_i, q_\varrho), \mathcal{V}(q') \models \gamma\}$, is the transition function for the extended model.

Model $\mathcal{M}_\varrho$ is like $\mathcal{M}$ though suitably extended to account for agents' goals under the initial goal-assignment ϱ. Observe that the transition relation caters for persistence of goals as well as dropping of achieved goals. Indeed, the extended system will never evolve to an (extended) state in which some agent has some true fact as a goal. Hence, the transition relation is well-defined within Q_ϱ states. More interesting, the extended model keeps the original physical executability of actions and, as a result, it accommodates both rational and irrational paths. However, it is now possible to discriminate between them, as one can reason about applicable plans in each state. Finally, it is not difficult to see that the extended model is, in general, exponentially larger than the original one with respect to the number of goals $\max_{agt\in\mathcal{A}}(|\varrho[agt]|)$ and agents $|\mathcal{A}_\varrho|$.

As standard, we denote the states satisfying a formula φ by $[\varphi]$. When the model is not clear from the context, we use $[\varphi]_\mathcal{M}$ to denote the states in $\mathcal{M}$ that satisfy the formula φ. We extend $ws(\cdot)$ projection function to sets of extended states in the straightforward sense, that is, $ws(S) = \bigcup_{q\in S}\{ws(q)\}$. Thus, $ws([\varphi]_{\mathcal{M}_\varrho})$ denotes the set of all world states in $\mathcal{M}$ that are part of an extended state in $\mathcal{M}_\varrho$ satisfying the formula φ. Also, $[\![\varrho]\!]$ denotes the set of extended states where the agents' goals are as per goal assignment ϱ; formally, $[\![\varrho]\!] = \{q \mid q \in Q_\varrho, \forall agt \in \mathcal{A}_\varrho : gl(agt, q) = \varrho[agt]\}$.

Figure 2 shows the model checking algorithm for BDI-ATLES. It is based on the symbolic model checking algorithm for ATL [3] and ATLES [20]. The first three cases are handled in the same way as in ATL(ES). To check the BDI-ATLES coalition formulae $\langle\!\langle A\rangle\!\rangle_{\omega,\varrho}\varphi$, we extend the model as above (relative to the formula's goal assignment ϱ), and then check the plain ATL coalition formula $\langle\!\langle A\rangle\!\rangle\varphi$ in such extended model. Note that only the set of states having the goals as per the initial goal assignment are returned—all agents' initial goals are active in the first state of any rational trace.

Unlike standard ATL model checking, we restrict the agents' action choices as per their capabilities. This is achieved by modifying the usual pre-image function $Pre(\cdot)$ to only take into account actions resultant from agents' applicable plans. More concretely, $Pre(A,\omega,\Theta,\rho)$ is the set of (extended) states from where agents in coalition A can jointly force the next (extended) state to be in set ρ no matter how all other agents (i.e., agents in $\mathcal{A}\setminus A$) may act and provided all BDI-style agents (i.e., agents with capabilities defined under ω and Θ) behave as such. Formally:

$$\begin{aligned}Pre(A,\omega,\Theta,\rho) = \{q \mid\ & \forall i \in A, \exists a_i \in d^+_\varrho(i,q,\omega,\Theta),\\ & \forall j \in \mathcal{A}\setminus A, \forall a_j \in d^+_\varrho(j,q,\omega,\Theta) : \sigma_\varrho(q,\langle a_1,\ldots,a_{|\mathcal{A}|}\rangle)\in\rho\},\end{aligned}$$

where auxiliary function $d^+_\varrho(agt,q,\omega,\Theta)$ denotes the set of all actions that an agent agt may take in state q under capabilities as per defined in ω and Θ:

$$d^+_\varrho(agt,q,\omega,\Theta) = \begin{cases} d_\varrho(agt,q) & \text{if } agt \notin \mathcal{A}_\omega \\ d_\varrho(agt,q) \cap d^{\mathrm{BDI}}(agt,q,\bigcup\limits_{c\in\omega[agt]}\Theta(c)) & \text{if } agt \in \mathcal{A}_\omega \end{cases}$$

An action belongs to set $d^+_\varrho(agt,q,\omega,\Theta)$ if it is physically possible (i.e., it belongs to $d_\varrho(agt,q)$), and BDI-rational whenever the agent in question is a BDI agent. To capture the latter constraint, set $d^{\mathrm{BDI}}(agt,q,\Pi)$ is defined as the set of all rational actions for agent agt in (extended) state q when the agent is equipped with the set of plans Π:

$$d^{\mathrm{BDI}}(agt,q,\Pi) = \begin{cases} \{a \mid \phi[a]\psi \in \Delta(agt,q,\Pi)\} & \text{if } \Delta(agt,q,\Pi) \neq \emptyset \\ \{\textsc{NoOp}\} & \text{otherwise} \end{cases}$$

where $\Delta(agt,q,\Pi) = \{\phi[a]\psi \in \Pi \mid \mathcal{V}(q) \models \phi, \gamma \in gl(agt,q), \psi \models \gamma\}$ is the set of all applicable plans in Π at state q. So, summarising, function $Pre(\cdot,\cdot,\cdot,\cdot)$ is an extension of the standard ATL $Pre(\cdot)$ function in which the agents that have goals and capabilities defined—the BDI agents—do act according to those goals and capabilities.

It is clear that the modified version of $\mathit{Pre}(\cdot)$ function does not alter the complexity of the underlying ATL-based algorithm. In fact, the variation is similar to that used for model checking ATLES, except that the action filtering does not come from strategy terms, but from agent plans. This means that the algorithm runs in polynomial time w.r.t. the size of model $\mathcal{M}_\varrho$ (which is exponential w.r.t. the original model $\mathcal{M}$).

Theorem 1. *Model checking a BDI-ATLES formula $\langle\langle A\rangle\rangle_{\omega,\varrho}\varphi$ (against a model $\mathcal{M}$) can be done in exponential time on the number of agents $|\mathcal{A}|$ and goals $\max_{a\in\mathcal{A}}(|\varrho[a]|)$.*

Of course, should we have included agents' goals explicitly in models (rather than using a succinct representation), as done with intentions in ATL+intentions (ATLI) [15], the model checking problem would retain ATL's polynomial complexity. The same would apply if one just generalized ATLES to explicitly require all rational-strategies be part of the model. The fact is, however, that generating such rational strategies by hand (to include them in models) will be extremely involved, even for small problems. In addition, our approach decouples agent's mental attitudes from the physical ATL-like model, and enables reasoning at the level of formulae without changing the model.

We shall note that the exponentiality may not show up in certain applications. In many cases, for example, one is interested in just one BDI agent acting in an environment. In that case, only such agent will be ascribed goals and capabilities. Since it arises due to agents with goals, the exponential complexity would therefore only be on the number of goals for such agent. Similarly, in situations where all agents have a single goal to achieve (e.g., to pick gold), the model checking would then be exponential on the number of BDI agents only. In the next section we shall provide one interpretation of goals for which the model checking problem remains polynomial.

5 BDI-ATLES with Maintenance Goals

So far, we have worked on the assumption that agents have a set of "flat" *achievement* goals, goals that the agent needs to eventually bring about. One can however consider alternative views of goals that could suit different domains. In particular, we have considered achievement goals with *priorities* and repetitive/reactive *maintenance* goals. In the first case, the framework can be easily generalized to one in which goals can be prioritized without an increase in complexity [21].

A more promising case arises when goals are given a maintenance interpretation, that is, (safety) properties that ought to be preserved temporally. For example, a Mars robot has the goal to always maintain the fuel level above certain threshold. We focus our attention on the so-called *repetitive* or *reactive* maintenance goals [10, 12]: goals that ought to be restored whenever "violated." Should the fuel level drop below the threshold, the robot will act towards re-fueling. This type of goals contrast with *proactive* maintenance goals [12], under which the agent is expected to proactively avoid situations that will violate the goal. The fact is, however, that almost all BDI platform—like JACK, JASON, and JADEX—only deal with the reactive version, thus providing a middle ground between expressivity and tractability.

Technically, to accommodate maintenance goals within BDI-ATLES, one only needs to do a small adaptation of the semantics of the logic so that goals are not dropped forever once satisfied, but "re-appear" when violated. We refer to this alternate version of our logic as BDI-ATLES^M. Of course, the model checking algorithm discussed above also needs to be slightly adapted to deal with the new goal semantics. Interestingly, one only needs to adapt the definition of a goal-extended model M_ϱ by re-defining components Q_ϱ and $\sigma_\varrho(q_\varrho, a)$; see [21] for details.

Theorem 2. *Model checking in BDI-ATLES^M can be done in polynomial time (w.r.t. the model and the formula).*

Hence, for (reactive) maintenance goals, we retain ATL(ES) polynomial complexity.[3] Of course, this bound is tight, as BDI-ATLES^M subsumes ATL (just take $\omega = \varphi = \emptyset$ in every coalition formula) and model checking ATL is PTIME-complete [3].

6 Discussion

We have developed an ATL-like logic that relates closely to the BDI agent-oriented programming paradigm widely used to *implement* multi-agent systems. In the new logic, the user can express the capability of agents equipped with know-how knowledge in a natural way and can reason in the language about *what agents can achieve under such capabilities*. Besides the general framework with standard achievement goals, we argued that one could instead appeal to goals with priorities or a special type of maintenance goals. We provided algorithms for model checking in such a framework and proved its (upper-bound) complexity in the various cases. Overall, we believe that this work is a first principled step to bring together two different fields in the area of multi-agent systems, namely, verification of strategic behaviour and agent programming.

The framework presented here made a number of assumptions requiring further work. Due to valuation function $\mathcal{V}$ in a structure, all agents are assumed to have full shared observability of the environment. This is, of course, a strong assumption in many settings. We considered here basically reactive plans, akin to the language of GOAL [11], certain classes of 2APL/3APL [8, 13], reactive modules [4], and universal plans [19]. We would like to explore the impact of allowing plan bodies having sequences of actions, and more importantly, sub-goaling, as well as the possibility of agents imposing (new) goals to other agents, via so-called BDI *messages*. Also, in the context of complex plan bodies, one could then consider both a linear as well as interleaved execution styles of plans within each agent (for its various goals). Most of these issue appear to be orthogonal to each other, and hence can be investigated one by one. With the core framework laid down, our next efforts shal focus on the above issues, as well as proving whether the complexity result provided in Theorem 1 is tight.

We close by noting that, besides ATLES, our work has strong similarities and motivations to those on *plausibility* [14] and *intention* [15] reasoning in ATL. Like ATLES, however, those works are still not linked to any approach for the actual development of agents, which is the main motivation behind our work. Nonetheless, we would like to investigate how to integrate plausibility reasoning in our logic, as it seems orthogonal to that of rational BDI-style behavior. Indeed, the plausibility approach allows us to focus the reasoning to certain parts of an ATL structure using more declarative specifications.

[3] Note the complexity of model checking ATLES is known only for memoryless strategies [20].

References

[1] Alechina, N., Dastani, M., Logan, B., Meyer, J.-J.: A logic of agent programs. In: Proc. of the National Conference on Artificial Intelligence (AAAI), pp. 795–800 (2007)
[2] Alechina, N., Dastani, M., Logan, B., Meyer, J.-J.: Reasoning about agent deliberation. In: Proc. of Principles of Knowledge Representation and Reasoning (KR), pp. 16–26 (2008)
[3] Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM (49), 672–713 (2002)
[4] Baral, C., Son, T.C.: Relating theories of actions and reactive control. Electronic Transactions of AI (ETAI) 2(3-4), 211–271 (1998)
[5] Bordini, R.H., Braubach, L., Dastani, M., Fallah-Seghrouchni, A., Gómez Sanz, J.J., Leite, J., O'Hare, G., Pokahr, A., Ricci, A.: A survey of programming languages and platforms for multi-agent systems. Informatica (Slovenia) 30(1), 33–44 (2006)
[6] Bratman, M.E., Israel, D.J., Pollack, M.E.: Plans and resource-bounded practical reasoning. Computational Intelligence 4(3), 349–355 (1988)
[7] Busetta, P., Rönnquist, R., Hodgson, A., Lucas, A.: JACK intelligent agents: Components for intelligent agents in Java. AgentLink Newsletter 2, 2–5 (1999)
[8] Dastani, M.: 2APL: A practical agent programming language. Autonomous Agents and Multi-Agent Systems 16(3), 214–248 (2008)
[9] Dastani, M., Jamroga, W.: Reasoning about strategies of multi-agent programs. In: Proc. of Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 997–1004 (2010)
[10] Dastani, M., van Riemsdijk, B., Meyer, J.-J.: Goal types in agent programming. In: Proc. of Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 1285–1287 (2006)
[11] de Boer, F., Hindriks, K., van der Hoek, W., Meyer, J.: A verification framework for agent programming with declarative goals. Journal of Applied Logic 5(2), 277–302 (2007)
[12] Duff, S., Harland, J., Thangarajah, J.: On proactivity and maintenance goals. In: Proc. of Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 1033–1040 (2006)
[13] Hindriks, K.V., de Boer, F.S., van der Hoek, W., Meyer, J.-J.: Agent programming in 3APL. Autonomous Agents and Multi-Agent Systems 2(4), 357–401 (1999)
[14] Jamroga, W., Bulling, N.: A framework for reasoning about rational agents. In: Proc. of Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 1–3 (2007)
[15] Jamroga, W., van der Hoek, W., Wooldridge, M.J.: Intentions and Strategies in Game-Like Scenarios. In: Bento, C., Cardoso, A., Dias, G. (eds.) EPIA 2005. LNCS (LNAI), vol. 3808, pp. 512–523. Springer, Heidelberg (2005)
[16] Padgham, L., Lambrix, P.: Formalisations of capabilities for BDI-agents. Autonomous Agents and Multi-Agent Systems 10(3), 249–271 (2005)
[17] Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In: Proc. of Principles of Knowledge Representation and Reasoning (KR), pp. 473–484 (1991)
[18] Rao, A.S., Georgeff, M.P.: An abstract architecture for rational agents. In: Proc. of Principles of Knowledge Representation and Reasoning (KR), pp. 438–449 (1992)
[19] Schoppers, M.: Universal plans for reactive robots in unpredictable environments. In: Proc. of the Int. Joint Conference on Artificial Intelligence (IJCAI), pp. 1039–1046 (1987)
[20] Walther, D., van der Hoek, W., Wooldridge, M.: Alternating-time temporal logic with explicit strategies. In: Conference on Theoretical Aspects of Rationality and Knowledge, pp. 269–278. ACM Press (2007), doi:10.1145/1324249.1324285
[21] Yadav, N., Sardina, S.: Reasoning about agent programs using ATL-like logics (2012) Available from CoRR, `http://arxiv.org/abs/1207.3874`

Qualitative Approximate Behavior Composition

Nitin Yadav and Sebastian Sardina[⋆]

RMIT University, Melbourne, Australia

Abstract. The behavior composition problem involves automatically building a controller that is able to realize a desired, but unavailable, target system (e.g., a house surveillance) by suitably coordinating a set of available components (e.g., video cameras, blinds, lamps, a vacuum cleaner, phones, etc.) Previous work has almost exclusively aimed at bringing about the desired component in its totality, which is highly unsatisfactory for unsolvable problems. In this work, we develop an approach for *approximate* behavior composition without departing from the classical setting, thus making the problem applicable to a much wider range of cases. Based on the notion of simulation, we characterize what a maximal controller and the "closest" implementable target module (optimal approximation) are, and show how these can be computed using ATL model checking technology for a special case. We show the uniqueness of optimal approximations, and prove their soundness and completeness with respect to their imported controllers.

1 Introduction

The behavior composition problem (e.g., [2, 6, 12, 19]) involves the automatic synthesis of a controller that is able to "realize" (i.e., implement) a desired, though non-existent, complex target system by suitably coordinating a collection of partially controllable available behaviors. A behavior here refers to the abstract operational model of a device or program, generally represented as a non-deterministic transition system. Thus, in a smart building setting, one may look for a controller able to coordinate the execution of a set of devices installed in a house—music and movie players, game consoles, automatic blinds and lights, radios, etc.—such that it appears as if a complex entertainment system was actually being run. A solution to the problem is called a *composition*.

The composition problem is appealing to a wide range of audiences. Indeed, with computers now present in everyday devices like mobile phones, credit cards, or places like homes, offices and factories, the trend is to build embedded complex devices from a collection of simple components. In addition, the problem can be related to several subareas of AI and CS, including web-service composition [10], reactive synthesis [14], agent-oriented programming [18], robot ecologies [15], and automated planning [8].

While the behavior composition problem has been substantially studied in an AI context lately (e.g., [6, 17, 19]), previous work has exclusively aimed at the synthesis of *complete* realisations of the desired target component—compositions that implement the desired component in its totality. This poses a major limitation in problem instances with no (exact) compositions. For such cases, a merely "no solution" outcome is extremely unsatisfactory. The need to address this shortcoming has already been noted in

[⋆] We acknowledge the support of the Australian Research Council under grant DP120100332.

previous works [19, 20]. In this paper, we develop a qualitative account of *approximate behavior composition* that caters for instances admitting no exact solutions.

Intuitively, the overarching idea is to *look for those parts of the target module that can be realized with the available modules*, and provide this as an (approximate) solution. More precisely, given a target module, the task is to identify the *closest* alternative target module that can be fully realized with the behaviors at hand—the optimal approximate target. Of course, it is expected that such alternative target will generally provide less functionalities than the original one. Indeed, some execution paths may be impossible to generate with the new target (e.g., it may no more be feasible to play video games when listening to music). Moreover, the alternative target may accommodate less "freedom" of choices in executions (e.g., when requesting to watch a movie, one may now need to commit to whether one will be playing a video game or listening to radio afterwards). Nonetheless, the user can request actions as per the alternative (approximate) target and be guaranteed her requests will always be fulfilled.

Observe that in this paper we assume a setting of *strict* uncertainty, in that the space of possibilities (behaviors' evolutions and target requests) is known, but the probabilities of these potential alternatives cannot be quantified [7]. This contrasts with our previous approach [20], which assumes all such probabilities have been specified for the domain and then looks for the "best" controller possible from a decision-theoretic perspective. Consequently, our account here can be seen as the next natural extension of the "classical" composition framework found in the literature, in that no no additional domain information is assumed. We shall discuss and compare this further in Section 6.

The rest of the paper is organized as follows. In the next two sections, we introduce the composition framework as known in the literature. Besides providing the standard notion for exact compositions (complete solutions to the problem), we also introduce the notion of *maximal compositions*, as controllers that can do as well as any other controller. After that, we develop the main contribution of our work, namely, the notion of *optimal target approximations* as the best alternative target behaviors that can be fully realized in the system at hand. We demonstrate that "importing" controllers from optimal approximations amounts to using maximal controllers (for the original target), thus providing correctness for optimal approximations. In addition, we show that the imported controllers of an optimal approximation together realize the same set of traces as those realized by maximal controllers (together as well), thereby providing a completeness result. More importantly, we prove that optimal approximations are in fact unique (up to simulation equivalence), a very interesting and unexpected property. Finally, we describe how optimal approximate targets can be computed for the special case of deterministic systems (as, for example, in the context of service composition; e.g, [2, 3]) by reducing the problem to ATL model checking, opening the door for advanced model checking tools. We close the paper with a short discussion and conclusions. An extended version of the paper, including proofs, can be found in [21].

2 The Behavior Composition Framework

In a behavior composition setting, a set of *available behaviors* are meant to jointly bring about a *virtual target behavior* [6, 17, 19]. We follow the composition framework in [17] with two minor modifications. For simplicity, we do not deal with the so-called environment, the shared space where behaviors are meant to execute. Nonetheless, all results presented here can be easily generalized to account for an environment. Second, we shall generalize target behaviors to non-deterministic transition systems.

Behaviors. A behavior stands for the operational model of a program or device. In general, behaviors provide, step by step, the user a set of actions that it can perform (relative to its specification). At each step, the behavior can be instructed to execute one of the legal actions, causing the behavior to transition to a successor state, and thereby providing a new set of applicable actions.

Formally, a *behavior* is a tuple $\mathcal{B} = \langle B, \mathcal{A}, b_0, \varrho \rangle$, where:[1]

- B is the finite set of behavior's states;
- $\mathcal{A}$ is a set of actions;
- $b_0 \in B$ is the initial state;
- $\varrho \subseteq B \times \mathcal{A} \times B$ is the behavior's transition relation, where $\langle b, a, b' \rangle \in \varrho$, or $b \xrightarrow{a} b'$ in $\mathcal{B}$, denotes that action a executed in behavior state b may lead the behavior to successor state b'.

Note that we allow behaviors to be non-deterministic, that is, given a state and an action, the behavior may transition to more than one state. This implies that one cannot know beforehand what actions will be available to execute after an action is performed, as the next set of applicable actions would depend on the successor state in which the behavior happens to be in. Hence, we say that non-deterministic behaviors are only *partially controllable*. A *deterministic* behavior is one where there is no state $b \in B$ and action $a \in A$ for which there exist two transitions $b \xrightarrow{a} b'$ and $b \xrightarrow{a} b''$ in $\mathcal{B}$ with $b' \neq b''$. A deterministic behavior is *fully controllable*. For the sake of legibility and easier notation, we shall assume, wlog, that behaviors capture non-terminating processes and hence do not have any terminating state with no outgoing transition.[2]

System and Enacted System. A system is a collection of behaviors at disposal. Technically, an (available) *system* is a tuple $\mathcal{S} = \langle \mathcal{B}_1, \ldots, \mathcal{B}_n \rangle$, where $\mathcal{B}_i = \langle B_i, \mathcal{A}_i, b_{i0}, \varrho_i \rangle$, for $i \in \{1, \ldots, n\}$, is a behavior, called an *available behavior* in the system.

To refer to the behavior that emerges from the joint execution of behaviors in a system, we use the notion of enacted system behavior. The *enacted system behavior* of an available system $\mathcal{S}$ (as above) is a tuple $\mathcal{E}_{\mathcal{S}} = \langle S_{\mathcal{S}}, \mathcal{A}, \{1, \ldots, n\}, s_{\mathcal{S}0}, \delta_{\mathcal{S}} \rangle$, where:

- $S_{\mathcal{S}} = B_1 \times \cdots \times B_n$ is the finite set of $\mathcal{E}_{\mathcal{S}}$'s states; when $s_{\mathcal{S}} = \langle b_1, \ldots, b_n \rangle$, we denote b_i by $beh_i(s_{\mathcal{S}})$, for $i \in \{1, \ldots, n\}$;
- $\mathcal{A} = \bigcup_{i=1}^{n} \mathcal{A}_i$ is the set of actions of $\mathcal{E}_{\mathcal{S}}$;
- $s_{\mathcal{S}0} \in S_{\mathcal{S}}$ with $beh_i(s_{\mathcal{S}0}) = b_{i0}$, for $i \in \{1, \ldots, n\}$, is $\mathcal{E}_{\mathcal{S}}$'s initial state;
- $\delta_{\mathcal{S}} \subseteq S_{\mathcal{S}} \times \mathcal{A} \times \{1, \ldots, n\} \times S_{\mathcal{S}}$ is $\mathcal{E}_{\mathcal{S}}$'s transition relation, where $\langle s_{\mathcal{S}}, a, k, s'_{\mathcal{S}} \rangle \in \delta_{\mathcal{S}}$, or $s_{\mathcal{S}} \xrightarrow{a,k} s'_{\mathcal{S}}$ in $\mathcal{E}_{\mathcal{S}}$, iff:
  - $beh_k(s_{\mathcal{S}}) \xrightarrow{a} beh_k(s'_{\mathcal{S}})$ in $\mathcal{B}_k$; and
  - $beh_i(s_{\mathcal{S}}) = beh_i(s'_{\mathcal{S}})$, for $i \in \{1, \ldots, n\} \setminus \{k\}$.

[1] With no shared environment in this paper, behaviors are not equipped with guard conditions (as done in [6, 19]) and the set of actions $\mathcal{A}$ are included in their definitions.

[2] As customary, e.g., in LTL verification, this can be easily achieved by introducing "fake" loop transitions.

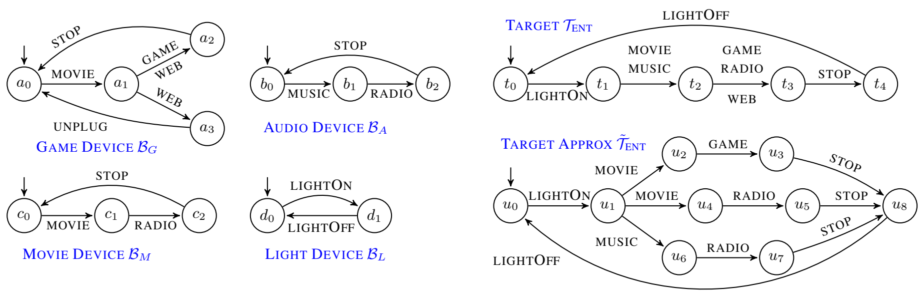

STOP
GAME
WEB
MOVIE
WEB
UNPLUG
GAME DEVICE $\mathcal{B}_G$
STOP
MUSIC
RADIO
AUDIO DEVICE $\mathcal{B}_A$
STOP
MOVIE
RADIO
MOVIE DEVICE $\mathcal{B}_M$
LIGHTON
LIGHTOFF
LIGHT DEVICE $\mathcal{B}_L$
TARGET $\mathcal{T}_{\text{ENT}}$
LIGHTOFF
LIGHTON
MOVIE
MUSIC
GAME
RADIO
WEB
STOP
TARGET APPROX $\tilde{\mathcal{T}}_{\text{ENT}}$
LIGHTON
MOVIE
GAME
STOP
MOVIE
RADIO
STOP
MUSIC
RADIO
STOP
LIGHTOFF

Fig. 1. A smart house scenario with four available behaviors. Target $\mathcal{T}_{\text{ENT}}$ cannot be fully realized in the system, but its optimal approximation $\tilde{\mathcal{T}}_{\text{ENT}}$ can.

The enacted system behavior $\mathcal{E}_{\mathcal{S}}$ is technically the asynchronous product of the available behaviors. The index k in transitions makes explicit which behavior is performing the action in the transition—all other behaviors remain still.

Target. A target behavior $\mathcal{T} = \langle T, \mathcal{A}_T, t_0, \varrho_T \rangle$ is a, possibly *non-deterministic*, behavior that represents the desired functionality to be obtained (through the available system). In contrast with all previous works, we allow for *non-deterministic* target specifications. Nonetheless, the objective is *not* to capture incomplete information, and hence partial controllability, of the target module, but to be able to accommodate action requests carrying more "information." This will come handy for our account of approximation. Thus, in order to preserve the full controllability of the target, we shall consider requests in terms of *target transition*, rather than just actions.

Informally, the behavior composition task is stated as follows: Given a system $\mathcal{S}$ and a target behavior $\mathcal{T}$, is it possible to (partially) control the available behaviors in $\mathcal{S}$ in a step-by-step manner—by instructing them on which action to execute next and observing, afterwards, the outcome in the behavior used—so as to "realize" the desired target behavior. In other words, by adequately controlling the system, it appears as if one was actually executing the target module. (See next section for more details.)

As noted by De Giacomo and Sardina [6], the behavior composition problem is related to planning (under incomplete information) [8], being both synthesis tasks, though here, we look for whom to delegate the next action at each step (whatever such action happens to be at runtime), rather than what those actions should be.

Figure 1 depicts a universal home entertainment system in a smart house scenario. Target $\mathcal{T}_{\text{ENT}}$ encapsulates the desired functionality, which involves first switching on the lights when entering the room, then providing various entertainment options (e.g., listening to music, watching movies, browsing the Web, etc.), and finally stopping active modules and switching off the lights. There are four available devices installed in the house that can be used to bring about such desired behavior, namely, a game device $\mathcal{B}_G$, an audio device $\mathcal{B}_A$, a movie device $\mathcal{B}_M$, and the lights controller $\mathcal{B}_L$. Note that action WEB in the device $\mathcal{B}_G$ is non-deterministic, as it may bring the module into states a_2 or a_3. If the device happens to evolve to state a_3, then, for some reason, it is not enough to stop the device to reset it: the device needs to be completely unplugged.

3 Controllers and Compositions

Next, we formally define what constitutes a solution for a behavior composition problem. In doing so, we shall not only look at the problem from a binary perspective — solvable vs unsolvable–but instead provide a *qualitative* account of "optimal" solutions. From now on, let $\mathcal{S} = \langle \mathcal{B}_1, \ldots, \mathcal{B}_n \rangle$ be an available system and $\mathcal{T} = \langle T, \mathcal{A}, t_0, \varrho_T \rangle$ be a target behavior to be realized on $\mathcal{S}$.

Controller. A controller is a component able to activate, stop, and resume any of the available behaviors, and to instruct them to execute an (allowed) action. The controller has *full observability* on the available behaviors; that is, it can keep track (at runtime) of their current states—if details have to be hidden, this can be done by means of non-determinism within the abstract behaviors exposed.

To formally define controllers and solutions, we rely on the notions of traces and histories. A trace for a given enacted system $\mathcal{E}_\mathcal{S} = \langle S_\mathcal{S}, \mathcal{A}, \{1, \ldots, n\}, s_{\mathcal{S}0}, \delta_\mathcal{S} \rangle$ is a, possibly infinite, sequence of the form $s^0 \xrightarrow{a^1, k^1} s^1 \xrightarrow{a^2, k^2} \cdots$ such that *(i)* $s^0 = s_{\mathcal{S}0}$; and *(ii)* $s^j \xrightarrow{a^{j+1}, k^{j+1}} s^{j+1}$ in $\mathcal{E}_\mathcal{S}$, for all $j > 0$. A history is just a finite prefix $h = s^0 \xrightarrow{a^1, k^1} \cdots \xrightarrow{a^\ell, k^\ell} s^\ell$ of a trace. We denote s^ℓ by *last*(h), ℓ by $|h|$ (i.e., the length of h), and sequence $a^1 \cdot \ldots \cdot a^\ell$ as $[h]$ (i.e., the projection on actions). Traces and histories can also be defined for a behavior $\mathcal{B}$ in a similar fashion: behavior traces have the form $s^0 \xrightarrow{a^1} s^1 \xrightarrow{a^2} \cdots$ such that *(i)* $s^0 = b_0$; and *(ii)* $s^j \xrightarrow{a^{j+1}} s^{j+1}$ in $\mathcal{B}$, for all $j > 0$. We use $\mathcal{H}_\mathcal{S}$ and $\mathcal{H}_\mathcal{B}$ to denote the set of system histories (i.e., histories of $\mathcal{E}_\mathcal{S}$) and histories of behavior $\mathcal{B}$, respectively.

A controller for target $\mathcal{T}$ on system $\mathcal{S}$ is a partial function $C : \mathcal{H}_\mathcal{S} \times (T \times \mathcal{A} \times T) \mapsto \{1, \ldots, n\}$, which, given a system history $h \in \mathcal{H}_\mathcal{S}$ and a requested target transition $\langle t, a, t' \rangle \in \varrho_T$, returns the index of an available behavior to which the action a is delegated for execution. For legibility, we shall write $C(h, t_1 \xrightarrow{a} t_2)$ to compactly denote $C(h, t_1, a, t_2)$. Note here the slight departure form previous notions of controllers (e.g., [6, 17, 19]), in that a controller now receives a complete target transition as the next request, not just an action. While this has no impact when dealing with deterministic targets, it guarantees full controllability for nondeterministic ones.

Intuitively, a controller (fully) realizes a target behavior if for every trace (i.e., run) of the target, at every step, the controller returns the index of an available behavior that can perform the requested action. Formally, one first defines when a controller C realizes a trace of the target $\mathcal{T}$. Though not required for this paper, the reader is referred to [6, 17] for details on how to formally characterize trace realization. We denote $\Delta^C_{\langle \mathcal{S}, \mathcal{T} \rangle}$ the set of traces of $\mathcal{T}$ that controller C is able to realize in system $\mathcal{S}$. Then, a controller C realizes the target behavior $\mathcal{T}$ *iff* it realizes all its traces. In that case, C is said to be an exact composition for target $\mathcal{T}$ on system $\mathcal{S}$.

Now, suppose we are given a target behavior $\mathcal{T}$ and an available system $\mathcal{S}$, and that, as expected in many domains, there is no exact composition for $\mathcal{T}$ on $\mathcal{S}$—the target cannot be *completely* realized in the system. This is indeed the case in our example, as there is no exact composition for $\mathcal{T}_{\text{ENT}}$ in the house system. Merely returning a negative "no solution" outcome is highly unsatisfactory. The question then is: what does it mean for a controller C_1 to achieve "a better realization" of $\mathcal{T}$ on $\mathcal{S}$ than controller C_2?

To answer such a question in a qualitative manner, we rely on the extent at which controllers are able to honour arbitrary long set of target requests. We say that controller C_1 *dominates* controller C_2, denoted $C_1 \geq C_2$, *iff* $\Delta^{C_2}_{\langle \mathcal{S},\mathcal{T}\rangle} \subseteq \Delta^{C_1}_{\langle \mathcal{S},\mathcal{T}\rangle}$—$C_1$ can honour all request sequences that C_2 can honour, and possibly more. As usual, $C_1 > C_2$ is equivalent to $C_1 \geq C_2$ but $C_2 \not\geq C_1$, that is, $\Delta^{C_2}_{\langle \mathcal{S},\mathcal{T}\rangle} \subset \Delta^{C_1}_{\langle \mathcal{S},\mathcal{T}\rangle}$. A controller C is said to be a *maximal composition* (for a target on a system) *iff* for every other controller C', if $C' \geq C$, then $C \geq C'$ (or equivalently $C' \not> C$). In other words, maximal compositions are those for which there is no other controller that can realize strictly more runs of the target behavior in the system. We use $\textsc{MaxComp}(\mathcal{S},\mathcal{T})$ to denote the set of all maximal compositions for target $\mathcal{T}$ on system $\mathcal{S}$.

Consider the following two controllers for our smart house. Whereas controller C_1 allocates all requests to the light device $\mathcal{B}_L$, controller C_2 delegates media and light requests to the audio $\mathcal{B}_A$ and light $\mathcal{B}_L$ devices, respectively. Then, C_1 realizes just one target trace, that is, $\Delta^{C_1}_{\langle \mathcal{S},\mathcal{T}\rangle} = \{t_0 \xrightarrow{\text{LIGHTON}} t_1\}$. On the other hand, C_2 realizes such a trace as well as trace $t_0 \xrightarrow{\text{LIGHTON}} t_1 \xrightarrow{\text{MOVIE}} t_2 \xrightarrow{\text{RADIO}} t_3 \xrightarrow{\text{STOP}} t_4$ (and all its prefixes). Therefore, $\Delta^{C_1}_{\langle \mathcal{S},\mathcal{T}\rangle} \subset \Delta^{C_2}_{\langle \mathcal{S},\mathcal{T}\rangle}$ and $C_2 > C_1$ holds. The reader may notice that even better controllers than C_2 exist when all four behaviors are used.

As expected, whenever a behavior composition problem admits an exact composition—the target is fully realizable—the set of exact compositions coincides with that of maximal compositions. When full realizations are impossible, though, maximal compositions capture the best controllers that one could hope for.

4 Target Approximation

Whereas maximal compositions, as defined above, provide a way of handling instances with no exact solution, they do not convey useful insights on how well such instances can be solved. Even if we are given the set of traces that a maximal composition realizes, it will be difficult to reconstruct what it means in terms of the problem specification. As a consequence, using a maximal non-exact composition may yield dead-end executions where no further actions can be honoured. What is more, while there are various techniques to construct exact compositions (e.g., [6, 16, 19]), it is far from clear how to build maximal composition controllers.

So, in this section, we will look at "approximation" from a different perspective that is arguably more intuitive and computationally more amenable than dealing with controller functions, namely, we are concerned with what parts of the target can in fact be brought about. More concretely, we are interested in the following task:

> *Given an available system $\mathcal{S}$ and a target behavior $\mathcal{T}$, find an (approximate) target behavior $\tilde{\mathcal{T}}$ that can be fully realized on $\mathcal{S}$ (by some controller $C_{\tilde{\mathcal{T}}}$) and such that $\tilde{\mathcal{T}}$ is "as close as possible" to the original target behavior $\mathcal{T}$.*

We call this the *approximate behavior composition problem*. Once an approximate target $\tilde{\mathcal{T}}$ is obtained, one may either use such new target directly or consider "importing" its exact compositions into the original target module $\mathcal{T}$. Hopefully, in the latter case, the imported controllers will turn out to be the best possible controllers for the original target. These are arguably the main ideas of our work and what we shall develop below. Before doing so, we should point out that defining approximate targets based merely on trace/language inclusion is not sufficient. While two targets may yield exactly the same sequences of requests, one may accept an exact composition while the other may not. In our smart house scenario, for instance, the two sequences LIGHTON · MOVIE · GAME · STOP and LIGHTON · MOVIE · RADIO · STOP may be realized by the same controller for the approximation $\tilde{\mathcal{T}}_{\mathrm{ENT}}$, but not for the original target $\mathcal{T}_{\mathrm{ENT}}$.

In order to capture approximate targets, we make use of the formal notion of *simulation* [13]. A simulation relation captures the similarity in the behavior of two transition systems. Intuitively, a (transition) system S_1 "simulates" another system S_2 if S_1 is able to *match* all of S_2's moves. We make this precise for our (target) behaviors as follows. Let $\mathcal{T}_i = \langle T_i, \mathcal{A}, t_{i0}, \varrho_i \rangle$, where $i \in \{1, 2\}$, be two target behaviors. A *simulation relation* of $\mathcal{T}_2$ by $\mathcal{T}_1$ is a relation $\mathit{Sim} \subseteq T_2 \times T_1$ such that $\langle t_2, t_1 \rangle \in \mathit{Sim}$ implies that for every transition $\langle t_2, a, t_2' \rangle \in \varrho_2$ in $\mathcal{T}_2$, there exists a transition $\langle t_1, a, t_1' \rangle \in \varrho_1$ in $\mathcal{T}_1$ such that $\langle t_2', t_1' \rangle \in \mathit{Sim}$. We say that a state $t_2 \in T_2$ is *simulated* by a state $t_1 \in T_1$ (or t_1 simulates t_2), denoted $t_2 \preceq t_1$, *iff* there exists a simulation relation Sim of T_2 by T_1 such that $\langle t_2, t_1 \rangle \in \mathit{Sim}$. Observe that relation $\preceq$ is itself a simulation relation (of $\mathcal{T}_2$ by $\mathcal{T}_1$), and in fact, it is the largest simulation relation, in that all simulation relations are contained in it. Informally, $t_2 \preceq t_1$ means that t_1 in $\mathcal{T}_1$ can "mimic" all moves of t_2 in $\mathcal{T}_2$, and that this property is propagated in their corresponding successor states. We say that a target behavior $\mathcal{T}_1$ *simulates* target behavior $\mathcal{T}_2$, denoted $\mathcal{T}_2 \preceq \mathcal{T}_1$, if it is the case that $t_{20} \preceq t_{10}$, that is, their initial states are in simulation and, as a result, $\mathcal{T}_1$ can always mimic $\mathcal{T}_2$ from the start. In our example, t_2 and t_1 in $\mathcal{T}_{\mathrm{ENT}}$ simulate states u_4 and u_1, respectively, in $\tilde{\mathcal{T}}_{\mathrm{ENT}}$ (i.e., $u_4 \preceq t_2$ and $u_1 \preceq t_1$), but not the other way around (i.e., $t_2 \not\preceq u_4$ and $t_1 \not\preceq u_1$). Two targets are said to be *simulation equivalent*, denoted $\mathcal{T}_1 \sim \mathcal{T}_2$, whenever they simulate each other.

We then argue that a qualitative comparison of target approximations can be achieved based on their simulation "hierarchy" (see that $\preceq$ is a pre-order). We say that a target behavior $\tilde{\mathcal{T}}$ *approximates* target $\mathcal{T}$ on system $\mathcal{S}$ (or $\tilde{\mathcal{T}}$ is an approximation of $\mathcal{T}$ on $\mathcal{S}$) *iff* $\tilde{\mathcal{T}} \preceq \mathcal{T}$ and there is an exact composition for $\tilde{\mathcal{T}}$ on $\mathcal{S}$ (i.e., $\tilde{\mathcal{T}}$ is simulated by $\mathcal{T}$ and it can be fully realized on available system $\mathcal{S}$).

Despite being fully solvable, an approximation will generally provide "*less*" than the original target. First, an approximation may be missing certain executions altogether. In the smart house scenario, approximation $\tilde{\mathcal{T}}_{\mathrm{ENT}}$ does not account for the action sequence LIGHTON · MUSIC · GAME · STOP · LIGHTOFF. Second, an approximation may require the user to commit earlier to future possible request choices. In that sense, a user of target $\tilde{\mathcal{T}}_{\mathrm{ENT}}$ needs to decide when requesting MOVIE in state u_1 if she will later play a GAME or listen to RADIO. Notice such extra "temporal" information is not required at state t_1 in original target $\mathcal{T}_{\mathrm{ENT}}$. It is exactly to accommodate this feature that we have departed from the standard view of deterministic targets.

Of course, between full realization and the trivial empty approximation, there lies a whole spectrum of approximating targets. Among these, we are interested in those that are "closest" to the original target, in that the minimum possible is given up. We say that a target behavior $\tilde{\mathcal{T}}$ is an *optimal approximate* of target $\mathcal{T}$ on system $\mathcal{S}$ *iff*:

1. $\tilde{\mathcal{T}}$ is an approximation of $\mathcal{T}$ on $\mathcal{S}$; and
2. there is no target behavior $\tilde{\mathcal{T}}'$ that approximates $\mathcal{T}$ on $\mathcal{S}$ such that $\tilde{\mathcal{T}} \prec \tilde{\mathcal{T}}'$, that is, $\mathcal{T}$ cannot be approximated by a strictly more general target module.

Intuitively, an optimal target approximation is a maximal representation of those aspects of the original target that can be completely implemented. When the target behavior does admit a full realization in the system, the optimal approximation is then expected to represent the target module in all its extent.

Theorem 1. *Suppose there is an exact composition for target $\mathcal{T}$ on system $\mathcal{S}$. Then, $\tilde{\mathcal{T}}$ is an optimal approximation of $\mathcal{T}$ on $\mathcal{S}$ iff $\tilde{\mathcal{T}} \sim \mathcal{T}$.*

Importantly, there can only be one way of optimally approximating a given target.

Theorem 2. *An optimal approximation $\tilde{\mathcal{T}}$ of a target $\mathcal{T}$ on a system $\mathcal{S}$ is unique upto simulation equivalence.*

We observe that, for non-deterministic transition systems, simulation is a stronger measure of equivalence than language inclusion [9]. Therefore, if a target $\tilde{\mathcal{T}}$ approximates another target $\mathcal{T}$, then the action request sequences resulting from the traces of $\tilde{\mathcal{T}}$ will be a subset of those produced by $\mathcal{T}$. It follows then that if $C_{\tilde{\mathcal{T}}}$ is an exact composition for $\tilde{\mathcal{T}}$, then $C_{\tilde{\mathcal{T}}}$ ought to be able to handle a subset of $\mathcal{T}$'s request sequences.

4.1 Imported Controllers

In contrast with maximal controllers, optimal approximations are specified in the *same* language as the original problem. The user can thus decide to request actions as per the new (approximate) target with guaranteed full realizability. Nonetheless, one may still ask in which sense these solutions are "correct." To answer that, we show that using an exact composition for an optimal approximation amounts to using a maximal composition for the original target. To that end, we define what it means to "import" a controller $C_{\mathcal{T}'}$ designed for one target module $\mathcal{T}'$ into another target module $\mathcal{T}$.

We start by defining the family of functions that are meant to explain sequences of action requests in a target. Informally, the function $\textsc{Expl}_{\mathcal{T}}(\sigma)$ outputs a history of the target $\mathcal{T}$ compatible with the given sequence of actions σ. Formally, a function $\textsc{Expl}_{\mathcal{T}} : \mathcal{A}^* \mapsto \mathcal{H}_{\mathcal{T}}$ is a target explanatory function for a target $\mathcal{T}$ if for any action sequence $\sigma = a^1 \cdot \ldots \cdot a^\ell \in \mathcal{A}^*$, with $\ell \geq 0$, it is the case that $\textsc{Expl}_{\mathcal{T}}(\sigma) = t^0 \xrightarrow{a^1} \cdots \xrightarrow{a^\ell} t^\ell \in \mathcal{H}_{\mathcal{T}}$. In general, there will be many of such functions, since the same sequence of action requests can arise from different runs of a non-deterministic target. For instance, sequence $\textsc{lightOn} \cdot \textsc{movie}$ can be explained in two ways on target $\tilde{\mathcal{T}}_{\textsc{ENT}}$, namely, via histories $u_0 \xrightarrow{\textsc{lightOn}} u_1 \xrightarrow{\textsc{movie}} u_2$ and $u_0 \xrightarrow{\textsc{lightOn}} u_1 \xrightarrow{\textsc{movie}} u_4$.

Using target explanatory functions, we next characterize the set of so-called *induced* controllers. Suppose we have a controller $C_{\mathcal{T}'}$ for a target $\mathcal{T}'$ (on a system $\mathcal{S}$). An induced controller (from controller $C_{\mathcal{T}'}$) for a target behavior $\mathcal{T}$ is one that handles requests from $\mathcal{T}$ as if they were requests issued as per module $\mathcal{T}'$. Recall that a controller for a system $\mathcal{S}$ outputs the behavior index to which a given transition-action request is delegated to at a certain system history. Formally, then, we say that $C_{\mathcal{T}}^{\mathcal{T}'}$ is an <u>*induced controller*</u> (from controller $C_{\mathcal{T}'}$ on target $\mathcal{T}'$) for target $\mathcal{T}$ over system $\mathcal{S}$ if there exists a target explanatory function $\textsc{Expl}_{\mathcal{T}'}(\cdot)$ for $\mathcal{T}'$ such that for every system history $h \in \mathcal{H}_{\mathcal{S}}$ and transition $t_1 \xrightarrow{a} t_2$ in $\mathcal{T}$, the following holds (recall that $[h]$ denotes the sequence of actions in history h):

$$C_{\mathcal{T}}^{\mathcal{T}'}(h, t_1 \xrightarrow{a} t_2) = \begin{cases} C_{\mathcal{T}'}(h, t_1' \xrightarrow{a} t_2') & \textsc{Expl}_{\mathcal{T}'}([h] \cdot a) = t^0 \xrightarrow{a^1} \cdots \xrightarrow{a^{|h|}} t_1' \xrightarrow{a} t_2' \\ \text{undefined} & \textsc{Expl}_{\mathcal{T}'}([h] \cdot a) \text{ is undefined} \end{cases}$$

That is, $\mathcal{T}$'s request $t_1 \xrightarrow{a} t_2$ is delegated at history h as controller $C_{\mathcal{T}'}$ would delegate request $t_1' \xrightarrow{a} t_2'$ from target $\mathcal{T}'$ if h's requests leave target $\mathcal{T}'$ in state t_1' and the current requested action a is indeed explained by transition request $t_1' \xrightarrow{a} t_2'$ in $\mathcal{T}'$. When there is no explanation in the $\mathcal{T}'$—$\textsc{Expl}(\cdot)$ is undefined—the induced controller is left undefined. Note that different ways of explaining original target's sequences of requests (i.e., different explanatory functions) yield different induced controllers.

Finally, an *imported* controller is a maximal (i.e., non-strictly dominated) controller within the family of induced controllers—the "best" induced controllers. Technically, the set of <u>*imported controllers*</u> from C on $\mathcal{T}$ into target $\mathcal{T}'$, denoted $\Omega_{\langle C,\mathcal{T}\rangle}^{\mathcal{T}'}$ is the set of all controllers $\hat{C}$ for $\mathcal{T}'$ such that *(i)* $\hat{C}$ is an induced controller from C on target $\mathcal{T}$ for $\mathcal{T}'$; and *(ii)* there is no other induced controller C' such that $C' > \hat{C}$.

First, we show that better target approximations amount to better, or more precisely "never worse," imported controllers.

Theorem 3. *Let $\tilde{\mathcal{T}}_1$ and $\tilde{\mathcal{T}}_2$ be two target approximations of target $\mathcal{T}$ on system $\mathcal{S}$, and let $\tilde{C}_1$ and $\tilde{C}_2$ be exact compositions of $\tilde{\mathcal{T}}_1$ and $\tilde{\mathcal{T}}_2$, resp. Suppose also that $\tilde{\mathcal{T}}_2 \preceq \tilde{\mathcal{T}}_1$ (i.e, $\tilde{\mathcal{T}}_1$ simulates $\tilde{\mathcal{T}}_2$). Then, for every controller $C_1 \in \Omega_{\langle \tilde{C}_1,\tilde{\mathcal{T}}_1\rangle}^{\mathcal{T}}$, there is no controller $C_2 \in \Omega_{\langle \tilde{C}_2,\tilde{\mathcal{T}}_2\rangle}^{\mathcal{T}}$ such that $C_2 > C_1$ holds.*

In other words, if $\tilde{\mathcal{T}}_1$ is as good an approximation as $\tilde{\mathcal{T}}_2$, then $\tilde{\mathcal{T}}_1$'s imported controllers will not be worse than those imported from $\tilde{\mathcal{T}}_2$. More importantly, the next result demonstrates that importing controllers from an *optimal* approximation yields maximal compositions (for the original target being approximated), and that, together, they account for every trace of the original target that could ever be realized. In other words, $\Omega_{\langle \tilde{C},\tilde{\mathcal{T}}\rangle}^{\mathcal{T}}$ is sound and "complete."

Theorem 4. *Let $\tilde{\mathcal{T}}$ be an optimal approximation of target $\mathcal{T}$ on system $\mathcal{S}$, and $\tilde{C}$ be an exact composition for $\tilde{\mathcal{T}}$. Then,*

- *For all $C \in \Omega_{\langle \tilde{C},\tilde{\mathcal{T}}\rangle}^{\mathcal{T}}$, it holds that $C \in \textsc{MaxComp}(\mathcal{S}, \mathcal{T})$; and*
- *$\bigcup_{C \in \Omega_{\langle \tilde{C},\tilde{\mathcal{T}}\rangle}^{\mathcal{T}}} \Delta_{\langle \mathcal{S},\mathcal{T}\rangle}^{C} = \bigcup_{C \in \textsc{MaxComp}(\mathcal{S},\mathcal{T})} \Delta_{\langle \mathcal{S},\mathcal{T}\rangle}^{C}$, that is, all imported controllers account together for all realizable target traces.*

These two results are important in that they establish the relationship between approximating the target and optimizing its controller: optimizing targets implies optimizing controllers. A direct and expected consequence of Theorems 1 and 4 is that if the optimal approximation is simulation equivalent to the target, then every imported controller from such approximation is in fact an exact composition.

5 Computing Optimal Approximations for Deterministic Systems

Various techniques have been used to actually solve classical behavior composition problems, including PDL satisfiability [6], direct search-based approaches [19], LTL/ATL synthesis [5, 16], and computation of special kind of simulation relations [3, 17]. Unfortunately, all those techniques synthesize *exact* composition controllers. In the context of our work, we are interested in *computing optimal target approximations* instead. We show how this can be effectively done for the special case of *deterministic* available behaviors, as in the case of service composition [2, 3].

De Giacomo and Felli [5] has shown that the controller generator (i.e., a structure representing all exact compositions) can be synthesised by resorting to Alternating-time Temporal Logic (ATL) model checking. ATL [1] is a logic for reasoning about the ability of group of agents (i.e., coalitions) in multi-agent game structures. The advantages of reducing the composition problem to that of ATL reasoning is that it provides access to some of the most advanced model checking techniques and tools, such as MCMAS [11], that are in active development within the agent community.

ATL formulae are built by combining propositional formulas, the usual temporal operators—namely, $\bigcirc$ ("in the next state"), $\Box$ ("always"), $\Diamond$ ("eventually"), and $\mathcal{U}$ ("strict until")—and a *coalition path quantifier* $\langle\langle A\rangle\rangle$ taking a set of agents A as parameter. Intuitively, an ATL formula $\langle\langle A\rangle\rangle\phi$, where A is a set of agents, holds in an ATL structure if by suitably choosing their moves, the agents in A *can force* ϕ *true*, no matter how other agents happen to move. The semantics of ATL is defined in so-called *concurrent game structures* where, at each point, all agents simultaneously choose their moves from a finite set, and the next state deterministically depends on such choices.

In order to reduce a behavior composition problem to an ATL model checking problem, De Giacomo and Felli [5] basically define an ATL structure $\mathcal{M}_{\mathcal{S},\mathcal{T}}$ with one agent per available and target behavior, and one distinguished agent contr representing the controller. A state $\langle b_1, \ldots, b_n, t_s, a, t_d, k\rangle$ in such a model encodes the current state b_i of each available behavior, the current state t_s of the target, the current action a being requested by the target, the next target state t_d given the request, and the index of the available behavior to which the last action was delegated to. The initial states of $\mathcal{M}_{\mathcal{S},\mathcal{T}}$ encode all possible initial configurations of the composition framework—initial states for all behaviors and a legal initial request. Also, the structure is made to encode all legal evolutions of the composition instance. The task then involves model checking the special formula $\varphi = \langle\langle \mathsf{contr}\rangle\rangle\Box(\bigwedge_{i=1,\ldots,n} \mathit{state}_i \neq \mathit{error}_i)$ (against structure $\mathcal{M}_{\mathcal{S},\mathcal{T}}$),[3] which states that the controller agent has a strategy so that none of the n available behaviors end up in an error state. A behavior arrives to a distinguished "error"state if it is ever delegated an action that it cannot perform. As a result, the controller agent ought to make sure it always delegates actions in the right way so as to satisfy every potential request, that is, it has to solve the composition problem. Finally, De Giacomo and Felli [5, Definition 2 & Theorems 3 and 4] show how to extract a correct controller generator—a structure representing all exact compositions—from the set of *winning states* $[\varphi]_{\mathcal{M}_{\mathcal{S},\mathcal{T}}}$, namely, all those states q in $\mathcal{M}_{\mathcal{S},\mathcal{T}}$ such that $q \models \varphi$. Intuitively, a winning state for

[3] We note that [5] deals with final states where the composition execution may stop. For simplicity, we have not dealt with final configurations here, but one can easily accommodate them.

them is one in which the current request is legally honored to some available behavior and *all* corresponding successor states are winning.

Surprisingly, it turns out that one can readily adapt De Giacomo and Felli's reduction to actually synthesize an optimal approximation for a, possibly non-solvable, *deterministic* composition problem (and to extract the corresponding controller generator). Though it looks counter-intuitive, the key for this is to *include the target behavior in the coalition* so that the joint-strategy also includes selecting which transition from the actual target may be requested. In other words, we are instead to model check the following formula against structure $\mathcal{M}_{\mathcal{S},\mathcal{T}}$:

$$\tilde{\varphi} = \langle\langle \textsf{contr}, \textsf{tgt} \rangle\rangle \Box (\bigwedge_{i=1,\ldots,n} \mathit{state}_i \neq \mathit{error}_i).$$

In this case, a winning state in $[\tilde{\varphi}]_{\mathcal{M}_{\mathcal{S},\mathcal{T}}}$ is one in which the target requests actions such that the controller can (always) legally honor them to an available behavior, and has *some* corresponding successor winning state. Observe here the implicit existential quantification on the requests, as compared with the universal quantification implied in De Giacomo and Felli [5]'s encoding for exact composition synthesis.

Intuitively, the idea behind formula $\tilde{\varphi}$, as opposed to formula φ, is that the coalition is now in control of what can be requested (and what should not be). This suggests that the coalition has the ability to select which parts of the target can be executed without driving the available system into an "error" state (due to an impossible fulfilment of a request). It follows then that one can extract an *optimal* approximation from the *maximal* winning set $[\tilde{\varphi}]_{\mathcal{M}_{\mathcal{S},\mathcal{T}}}$, as the following result demonstrates.

Theorem 5. *Let $\mathcal{S} = \langle \mathcal{B}_1, \ldots, \mathcal{B}_n \rangle$ be a system and $\mathcal{T} = \langle T, \mathcal{A}, t_0, \varrho_T \rangle$ a target module. Then, behavior $\hat{\mathcal{T}} = \langle \hat{T}, \mathcal{A}, \hat{t_0}, \hat{\varrho} \rangle$ is an optimal approximation for $\mathcal{T}$ on $\mathcal{S}$, where:*

- $\hat{T} = \{\langle b_1, \ldots, b_n, t_s \rangle \mid \langle b_1, \ldots, b_n, t_s, a, t_d, k \rangle \in [\tilde{\varphi}]_{\mathcal{M}_{\mathcal{S},\mathcal{T}}}\} \cup \{\hat{t_0}\}$;
- $\hat{t_0} = \langle b_{10}, \ldots, b_{n0}, t_0 \rangle$ *is the initial state of* $\hat{\mathcal{T}}$;
- $\hat{\varrho}(\langle b_1, \ldots, b_n, t_s \rangle, a, \langle b'_1, \ldots, b'_n, t_d \rangle)$ iff *for some action $a' \in \mathcal{A}$, and indexes $k, k' \in \{1, \ldots, n\}$, it is the case that:*
  - $\langle b_1, \ldots, b_n, t_s, a, t_d, k \rangle, \langle b'_1, \ldots, b'_n, t'_s, a', t'_d, k' \rangle \in [\tilde{\varphi}]_{\mathcal{M}_{\mathcal{S},\mathcal{T}}}$*; and*
  - $\langle b_1, \ldots, b_n, t_s, a, t_d, k \rangle$ *may transition to* $\langle b'_1, \ldots, b'_n, t'_s, a', t'_d, k' \rangle$ *in* $\mathcal{M}_{\mathcal{S},\mathcal{T}}$.

It is not hard to see that the controller generator [17] for $\hat{\mathcal{T}}$ can be extracted by keeping those behavior delegations that transition a winning game state into another winning state in $\mathcal{M}_{\mathcal{S},\mathcal{T}}$. In terms of computational complexity, the model checking task on ATL can be done in polynomial time wrt to the size of the game structure [1]. Since the size of such space is exponential on the number of available behaviors, computing the optimal approximation can be done in exponential time (for deterministic systems). Observe that, in the worst case, the approximation problem itself is (at least) exponential, as it subsumes the classical behavior composition problem (which is known to be EXPTIME-complete even under deterministic behaviors). Indeed, in order to check if a problem has an exact composition one can compute its optimal approximation and test (in polynomial time) if it is simulation equivalent with the original target.

The full details of the ATL encoding, together with an implementation in MCMAS of our running example, can be found in [21].

6 Discussion

We have proposed a qualitative framework for approximate behavior composition in which the task is to find the *closest* possible target module that can be implemented with the available modules. To that end, we relied on the formal notion of simulation and that of imported controllers for the specification of the problem, and on ATL model checking for actual computation of solutions for the special case of deterministic systems. To our knowledge, this is the first account that is able to accommodate behavior composition instances with no complete solutions—arguably the most common ones—while still remaining within the original problem formulation.

Initially, the work of Girard and Pappas [9] appeared to be extremely related to our objectives, as it proposes a notion of transition system approximation based on the notion of simulation. However, their work differs in *what* is being approximated. In the most general notion of simulation, only some aspects of states are observable and two states in simulation are meant to coincide on their observable aspects. In Girard and Pappas's account, an approximate transition system is allowed to differ on such observables up to some extent: s simulates s' implies s can (always) replicate all moves of s' and s's observation is "similar" to that of s'. It follows then that the approximating transition system *must* still be able to mimic *all* actions of the approximated system. In our framework, there is no notion of state observations (every state has the same observations) and hence we only focus on the similarities of states in terms of the potential behavior they can generate. We believe though that one can use their account of approximation when performing composition *within a shared environment* (as in [6, 19]), so as to allow the environment to evolve "close enough" to what is necessary.

Confronted with a behavior composition problem instance admitting no complete solution (i.e., no exact composition) one can, of course, think of other approaches orthogonal to the one developed here. For example, one may look for additional available behavior modules or enhancement of existing ones with new capabilities that will recover exactness. In some cases, simply adding extra "copies" of existing modules could be enough. Thus, installing an extra video camera in the house may turn the problem solvable. One could also consider a framework where essential and optional functionalities can be specified, and look for controllers that fully realize the former ones while optimizing the latter ones. We shall focus on these ideas on future work, as well as on generalizing the actual synthesis techniques from Section 5 to nondeterministic systems, possibly relying on more expressive games using GR(1) formulas [4].

The only approach, as far as we know, to deal with unsolvable composition instances is the one we pursued previously in [20] within a decision-theoretic framework. There, the idea is to look for a controller that maximizes the *"expected realizability"* of the target behavior. There are however two major differences with our current proposal. First, their controller may in some runs yield dead-end situations, that is, states from where no further target request can be fulfilled. Under our framework, the user (of the target) can never arrive to those "error" situations, as the optimal approximation is always fully implementable. Second, in our work we kept the strict uncertainty setting from the composition problem found in the literature—no extra knowledge of the domain is assumed to be available. We note that it is well known that *strict* uncertainty cannot always be reduced to a setting where the uncertainty can be measured [7]. Nonetheless, it would be interesting to be able to accommodate extra domain knowledge *when available*.

References

[1] Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM (49), 672–713 (2002)

[2] Balbiani, P., Cheikh, F., Feuillade, G.: Composition of interactive web services based on controller synthesis. In: Proc. of SERVICES, pp. 521–528 (2008)

[3] Berardi, D., Cheikh, F., De Giacomo, G., Patrizi, F.: Automatic service composition via simulation. International Journal of Foundations of Computer Science 19(2), 429–452 (2008)

[4] Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Saar, Y.: Synthesis of reactive(1) designs. Journal of Computer and System Sciences, 1–28 (2011)

[5] De Giacomo, G., Felli, P.: Agent composition synthesis based on ATL. In: Proc. of AAMAS, pp. 499–506 (2010)

[6] De Giacomo, G., Sardina, S.: Automatic synthesis of new behaviors from a library of available behaviors. In: Proc. of IJCAI, pp. 1866–1871 (2007)

[7] French, S.: Decision Theory: An Introduction to the Mathematics of Rationality. Ellis Horwood (1986)

[8] Ghallab, M., Nau, D.S., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann Publishers Inc. (2004)

[9] Girard, A., Pappas, G.: Approximation metrics for discrete and continuous systems. IEEE Transactions on Automatic Control 52(5), 782–798 (2007)

[10] Hull, R.: Web services composition: A story of models, automata, and logics. In: Proc. of SCC, pp. 18–19 (2005)

[11] Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 682–688. Springer, Heidelberg (2009)

[12] Lustig, Y., Vardi, M.Y.: Synthesis from Component Libraries. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 395–409. Springer, Heidelberg (2009)

[13] Milner, R.: An algebraic definition of simulation between programs. In: Proc. of IJCAI, pp. 481–489 (1971)

[14] Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Proc. of POPL, pp. 179–190 (1989)

[15] Saffiotti, A., Broxvall, M.: PEIS ecologies: Ambient intelligence meets autonomous robotics. In: Proc. of the International Conference on Smart Objects and Ambient Intelligence, pp. 275–280 (2005)

[16] Sardina, S., De Giacomo, G.: Realizing multiple autonomous agents through scheduling of shared devices. In: Proc. of ICAPS, pp. 304–312 (2008)

[17] Sardina, S., Patrizi, F., De Giacomo, G.: Behavior composition in the presence of failure. In: Proc. of KR, pp. 640–650 (2008)

[18] Shoham, Y.: Agent-oriented programming. Artificial Intelligence Journal 60, 51–92 (1993)

[19] Stroeder, T., Pagnucco, M.: Realising deterministic behaviour from multiple non-deterministic behaviours. In: Proc. of IJCAI, pp. 936–941 (2009)

[20] Yadav, N., Sardina, S.: Decision theoretic behavior composition. In: Proc. of AAMAS, pp. 575–582 (2011)

[21] Yadav, N., Sardina, S.: Qualitative approximate behavior composition (2012) Available from CoRR, `http://arxiv.org/abs/1207.3863`

A Preferential Framework for Trivialization-Resistant Reasoning with Inconsistent Information

Anna Zamansky[*]

Vienna University of Technology

Abstract. Paraconsistent entailments based on more than two truth-values are useful formalisms for handling inconsistent information in large knowledge bases. However, such entailments suffer from two major drawbacks: they are often too cautious to allow intuitive classical inference, and are trivialization-prone. Two preferential mechanisms have been proposed to deal with these two problems, but they are formulated in different terms, and are hard to combine. This paper is a step towards a systematization and generalization of these approaches. We define an abstract framework, which allows for incorporating various preferential criteria into paraconsistent entailments in a modular way. We show that many natural cases of previously studied entailments can be simulated within this framework. Its usefulness is also demonstrated using a concrete domain related to ancient geography.

1 Introduction

Handling inconsistent information in large knowledge bases is an important practical problem, that has been recently drawing a lot of attention. The main drawback of classical logic (CL) in this context is that it fails to accommodate the fact that knowledge bases containing contradictory data may still produce useful answers to queries. This is because in CL a single inconsistency leads to trivialization of the whole knowledge base. The traditional approach to handling inconsistency in knowledge bases has been that of *revision*: in case of an inconsistency, some pieces of information must be abandoned in order to maintain consistency. In many natural cases, however, inconsistency is an inherent, and even a very important part of information systems, and should not be treated as an undesirable phenomenon (see [12] for further discussion).

As a running example in this paper, we use a domain, in which inconsistent information plays an important role, and as such should often be preserved, rather than attempted to be discarded. The domain is taken from a concrete practical problem, pointed out to the author in a personal communication[1] by a researcher

[*] The author is supported by funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 252314.

[1] The author would like to thank Ekaterina Ilyuschetchkina for her valuable contribution to this paper.

of descriptive ancient geography. Due to a vast amount of information extracted from various ancient texts, there is a need for automated methods for keeping track and reasoning about geographic information. More specifically, texts of ancient authors (such as Eratosthenes, Strabo, Ptolemy, Pausanias, etc.) are used in the study of ancient geography to extract information about the ancient perception of the world. Such texts contain a lot of conflicting and contradictory information. The explanations for the contradictions vary: the primary objective of such texts is not scientific and thus they are often imprecise, the authors rely on information from different sources (the identity of which is usually not known), etc. One example of a text containing several controversial contradictions is a description of the habitable world written in a terse and elegant style by Dionysios Periegetes[2] ([15]). The text begins with describing a "bird's-eye" view of the world. In it, Periegetes describes schematically the continents of Africa, Europe and Asia as triangles (of different forms), and notes that the eastern border of Africa is defined by the Nile river. When later describing the "regional picture" of the continent of Africa, however, he states that on the east Africa borders with the Arabian Gulf[3] (the modern Red Sea). When describing in details the Nile river, he repeats, however, that Nile is the eastern border of Africa. Another example of (perhaps a more subtle) contradiction is that in his "bird's-eye" description of the world, Periegetes describes the continent of Africa as having the form of a triangle. However, when describing the regional picture, he states that is has the form of a trapezoid. But taking into account the mathematical theories developed far before his time, it is commonly believed that Periegetes makes a clear distinction between a triangle and a trapezoid.

Attempting to collect the pieces of information described in the above examples in a traditional knowledge base obviously results in its trivialization, i.e., everything (classically) follows from it. However, discarding any of the facts from the knowledge base would lead to loss of important information. The ideal solution is finding a way to keep all pieces of information in the knowledge base, while preventing its trivialization. This naturally leads to the need for *paraconsistent entailment relations*, which would allow to make useful inferences from inconsistent theories.

One of the most common ways of defining useful paraconsistent entailments is using three-valued (or in general many-valued) matrices. The idea is to add a new truth value $\top$ (or more), with the intuitive meaning of being "inconsistent", or "both true and false". The entailments induced by such matrices are "well-behaved" consequence relations (crs), which enjoy a simple and intuitive semantics and have a well-developed proof theory. However, such crs suffer from two major drawbacks. The first is that in many cases they are too cautious to

[2] The lifedates of Dionysios Periegetes (also known as Dionysios of Alexandria) are estimated around the 2nd century A.D. His poem enjoyed great popularity in ancient times. Although he is a poet and not a scientist, his texts today are a useful source for extracting information on geographic knowledge in the ancient world.

[3] More precisely, he speaks of the isthmus between the Arabian Gulf and the Mediterranean Sea.

allow natural (and seemingly harmless) classical inferences. Interestingly, such problem obtains even for those paraconsistent logics which are maximal (that is, any extension leads to a logic that is no longer paraconsistent), which means that extending the logic cannot be a satisfactory solution. One solution, proposed in [17] in the spirit of Shoham's preferential semantics ([21]), is to focus on preferential refinements of the basic inference relation, applying the principle of inconsistency minimization: the interpretations which are as close as possible to classical interpretations are preferred, resulting in a better approximation of classical reasoning.

The second drawback of paraconsistent many-valued logics (induced by some matrix $\mathcal{M}$), which becomes crucial in the context of practical applications, is that despite the fact that they tolerate classical inconsistency, they may still be trivialized in case of inconsistency in $\mathcal{M}$. A database repair method was proposed in [9,20] for two particular paraconsistent logics. However, in our context repair implies loss of information, so instead of refining the knowledge base, we would prefer to refine our entailment relation. A distance-based approach for defining trivialization-tolerant variants for logics induced by denotational semantics was proposed in [2,4] (see also [16]). The idea is to apply distance-based minimization of non-satisfiability for "approximation" of $\mathcal{M}$-models of an $\mathcal{M}$-inconsistent knowledge base. The resulting entailment coincides with the original logic in case that the knowledge base is $\mathcal{M}$-consistent, but does not trivialize otherwise. This method, however, does not apply to the above mentioned preferential refinements, which are, as observed in [17], trivialization-prone as well.

In this paper we propose a *systematic* approach to incorporate preferential criteria into paraconsistent entailments based on many-valued matrices. The approach is based on the observation, that despite the fact that the methods of [17] for handling cautiousness and the methods of [2,4] for handling trivialization are formulated in different terms, they are in fact different facets of minimization according to "epistemic" criteria. We therefore define an abstract framework for specifying such criteria in a *uniform* way, which also allows for a modular combination of these criteria. This allows for defining trivialization-resistant variants of various preferential paraconsistent entailments based on a matrix $\mathcal{M}$, which coincide with the original entailment whenever the premises are $\mathcal{M}$-consistent, but do not trivialize otherwise. This is done via *lexicographic aggregation of preference*, which is a common technique in choice theory [1]. The framework also captures many previously studied paraconsistent entailments (including the preferential relations on [17], summation-based entailments of [2,4] and maximal-consistency based cautious semantics of [13]). We demonstrate the usefulness of the framework using the above mentioned domain of descriptive ancient geography.

2 Preliminaries

In what follows, $\mathcal{L}$ denotes a propositional language with a countable set $\mathsf{Atoms} = \{p, q, r \ldots\}$ of atomic formulas and a (countable) set $\mathcal{W}_{\mathcal{L}} = \{\psi, \phi, \sigma, \ldots\}$ of well-formed formulas. A theory Γ is a finite set of formulas. The set of all theories of $\mathcal{L}$ is denoted by $\mathcal{T}_{\mathcal{L}}$.

Definition 1. – An *entailment relation* $\vdash$ for $\mathcal{L}$ is a binary relation between theories from $\mathcal{T}_{\mathcal{L}}$ and formulas in $\mathcal{W}_{\mathcal{L}}$.
- An entailment relation $\vdash$ for $\mathcal{L}$ is a *(Tarskian) consequence relation (tcr)* if it has the following properties: (i) *Reflexivity*: if $\psi \in \Gamma$ then $\Gamma \vdash \psi$, (ii) *Monotonicity*: if $\Gamma \vdash \psi$ and $\Gamma \subseteq \Gamma'$, then $\Gamma' \vdash \psi$, and (iii) *Transitivity*: if $\Gamma \vdash \psi$ and $\Gamma', \psi \vdash \varphi$ then $\Gamma, \Gamma' \vdash \varphi$.
- A *propositional logic* is a pair $\langle \mathcal{L}, \vdash \rangle$, where $\vdash$ is a structural[4] consequence relation for $\mathcal{L}$.

The most standard semantic way of defining logics is by many-valued matrices:

Definition 2. A (many-valued) *matrix* for a language $\mathcal{L}$ is a triple $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$, where $\mathcal{V}$ is a non-empty set of truth values, $\mathcal{D}$ is a non-empty proper subset of $\mathcal{V}$, containing the *designated* elements of $\mathcal{V}$, and $\mathcal{O}$ includes an n-ary function $\tilde{\diamond}_{\mathcal{M}} : \mathcal{V}^n \to \mathcal{V}$ for each n-ary connective $\diamond$ of $\mathcal{L}$.

Example 1. The simplest example of a many-valued matrix is the (standard) classical two-valued matrix for $\mathcal{L}_{cl}$, which we shall denote by $\mathcal{M}_{cl}$.

The three-valued matrix $\mathbf{M_{B_0}} = \{\{t, \top, f\}, \{t, \top\}, \mathcal{O}\}$ for $\mathcal{L}_{cl} = \{\neg, \vee, \wedge, \supset\}$ is used in [17], and is defined as follows:

$\tilde{\wedge}$	t	f	$\top$
t	t	f	$\top$
f	f	f	f
$\top$	$\top$	f	$\top$

$\tilde{\vee}$	t	f	$\top$
t	t	t	t
f	t	f	$\top$
$\top$	t	$\top$	$\top$

$\tilde{\to}$	t	f	$\top$
t	t	f	$\top$
f	t	t	t
$\top$	t	f	$\top$

	$\tilde{\neg}$
t	f
f	t
$\top$	$\top$

Definition 3. Let $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$ be a matrix for $\mathcal{L}$.

- An $\mathcal{M}$*-valuation* for $\mathcal{L}$ is a function $\nu : \mathcal{W}_{\mathcal{L}} \to \mathcal{V}$ such that for every n-ary connective $\diamond$ of $\mathcal{L}$ and every $\psi_1, \ldots, \psi_n \in \mathcal{W}_{\mathcal{L}}$, $\nu(\diamond(\psi_1, \ldots, \psi_n)) = \tilde{\diamond}_{\mathcal{M}}(\nu(\psi_1), \ldots, \nu(\psi_n))$. We denote the set of all the $\mathcal{M}$-valuations by $\Lambda_{\mathcal{M}}$.
- A valuation $\nu \in \Lambda_{\mathcal{M}}$ is an $\mathcal{M}$*-model* of a formula ψ, if it belongs to the set $mod_{\mathcal{M}}(\psi) = \{\nu \in \Lambda_{\mathcal{M}} \mid \nu(\psi) \in \mathcal{D}\}$. The $\mathcal{M}$-models of a theory Γ are the elements of the set $mod_{\mathcal{M}}(\Gamma) = \cap_{\psi \in \Gamma}\, mod_{\mathcal{M}}(\psi)$.
- A formula ψ (a theory Γ) is $\mathcal{M}$*-consistent* if $mod_{\mathcal{M}}(\psi) \neq \emptyset$ ($mod_{\mathcal{M}}(\Gamma) \neq \emptyset$).
- The entailment $\vdash_{\mathcal{M}}$ *induced by* $\mathcal{M}$, is defined by $\Gamma \vdash_{\mathcal{M}} \psi$ if $mod_{\mathcal{M}}(\Gamma) \subseteq mod_{\mathcal{M}}(\psi)$.

It is easy to see that for every matrix $\mathcal{M}$ for $\mathcal{L}$, $\langle \mathcal{L}, \vdash_{\mathcal{M}} \rangle$ is a propositional logic. This notion, however, is in many cases too restrictive. In the context of non-monotonic reasoning it is usual to consider the following weaker notion, that still guarantees in many cases that the induced entailment is "well-behaved" (see, e.g., [3,18,19]):

[4] An entailment $\vdash$ is structural if $\Gamma \vdash \varphi$ implies $\sigma(T) \vdash \sigma(\varphi)$ for every substitution σ.

Definition 4. An entailment $\vdash$ is a *cautious consequence relation* (with respect to $\mathcal{M}$) if it has the following properties: (i) *Cautious Reflexivity (with respect to $\mathcal{M}$):* if Γ is $\mathcal{M}$-consistent and $\psi \in \Gamma$ then $\Gamma \vdash \psi$, (ii) *Cautious Monotonicity* [11]: if $\Gamma \vdash \psi$ and $\Gamma \vdash \phi$ then $\Gamma, \psi \vdash \phi$, and (iii) *Cautious Transitivity* [18]: if $\Gamma \vdash \psi$ and $\Gamma, \psi \vdash \phi$ then $\Gamma \vdash \phi$.

3 Paraconsistent Entailments

The standard notion of paraconsistency with respect to $\neg$ (defined in [8,5] for propositional logics), can be extended to the context of (less restrictive) entailments as follows:

Definition 5. Let $\mathcal{L}$ be a language which includes a unary connective $\neg$. An entailment $\vdash$ for $\mathcal{L}$ is *paraconsistent* (with respect to $\neg$) if for every theory Γ and every $\mathcal{L}$-formula ψ, such that $\psi, \neg\psi \in \Gamma$, there is some $\mathcal{L}$-formula ϕ, such that $\Gamma \nvdash \phi$.

There is a vast amount of different paraconsistent entailments that have been suggested and investigated over the years. A natural question arises: what are the properties that a "well-behaved" entailment should satisfy? This question was considered in [5] for the case of propositional logics. The intuitive answer is that an ideal paraconsistent logic should follow the original intention of one of the founders of paraconsistent logics, Newton da Costa ([10]), by "retaining as much of classical logic as possible", while still allowing non-trivial inconsistent theories. This rather vague notion of "ideal logics" was defined in [5] in precise terms, and it includes the following basic requirements[5]: (i) containment of classical logic, (ii) absolute maximal paraconsistency (in the sense that any extension of the logic results in a non-paraconsistent logic), (iii) maximal paraconsistency with respect to classical logic (in the sense that extending the set of theorems of the logic results in classical logic), and (iv) reasonable language (that is, having a natural implication and conjunction). It was also shown that practically all reasonable three-valued paraconsistent logics are ideal in the defined sense.

In addition to having a natural implication and conjunction, a useful feature of the language of a paraconsistent logic is an internalization of the notion of consistency. A well-known example of logics which have this feature is the family of paraconsistent logics motivated by da Costa's approach, known as *Logics of Formal (In)consistency* (LFIs, [8]). In these logics we are able to make a distinction in the language between consistent and inconsistent propositions (in other words, "normal" ones – which cannot be both true and false, and "abnormal" ones for which such case is possible), and use this distinction to restrict classical rules of inference only to the "normal" ones. This distinction is often done using a primitive (or defined) unary connective $\circ$, where the meaning of $\circ\varphi$ is "φ is consistent".

[5] We refer the reader to [5] for the formal definitions and further details.

Example 2. Consider the matrix $\mathcal{M}_{\mathbf{B}}$ for $\mathcal{L} = \{\wedge, \vee, \supset, \neg, \circ\}$ obtained by extending $\mathcal{M}_{\mathbf{B_0}}$ from Example 1 by the following interpretation of $\circ$: $\tilde{\circ}t = \tilde{\circ}f = t$, $\tilde{\circ}\top = f$. The logic $\mathbf{B} = \langle \mathcal{L}, \vdash_{\mathcal{M}_{\mathbf{B}}} \rangle$ was introduced in [9,20] for repairing evolutionary databases (it was called **LFI1** there), and shown in [5] to be ideal in the sense explained above. Note that the explosion principle of classical logic does not hold in $\mathbf{B}$: $\psi, \neg\psi \nvdash_{\mathbf{B}} \varphi$, and so $\mathbf{B}$ is paraconsistent. However, a weakened version of the explosion principle does hold: $\circ\psi, \psi, \neg\psi \nvdash_{\mathbf{B}} \varphi$. Intuitively, this means that a consistent proposition cannot be both true and false.

Example 3. Let us utilize the logic $\mathbf{B}$ in the context of the ancient geography domain. The text of Periegetes starts with a "bird's-eye" description of the world, in which (among others) the following facts are mentioned: (i) The earth consists of three continents: Africa, Europe and Asia, (ii) Africa and Europe have the form of a triangle, (iii) The eastern border of Africa is the Nile river. In the regional description of Africa, a new fact appears: (iv) The eastern border of Africa is the Arabian Gulf. We can capture this information in the knowledge base EK (of explicit knowledge of Periegetes):

Continent(africa)
Continent(europe)
Continent(asia)
EastBorder (africa, nile)
Triangle(europe)
Triangle(africa)
EastBorder (africa, arabgulf)

We will also maintain an additional set of formulas which reflects implicit assumptions related to the geographic concepts (for instance, that a continent has only one geographical object on its eastern border), to the additional background knowledge Periegetes is assumed to have (for instance, that triangle is a different shape from a trapezoid), etc. Suppose we impose the following set of assumptions[6]: $\mathcal{IA} = \{\forall x \forall y \forall z$ EastBorder$(x,y) \wedge$ EastBorder$(x,z) \rightarrow y = z$, $\mathsf{nile} \neq \mathsf{arabgulf}$ $\}$. Clearly, $KB = EK \cup \mathcal{IA}$ is classically inconsistent, and is trivialized in classical logic. However, this is not the case if a three-valued paraconsistent logic is used instead. Using $\mathbf{B}$, however, trivialization is avoided: (i) $KB \nvdash_{\mathbf{B}}$ $\mathsf{EastBorder(europe, arabgulf)}$, (ii) $KB \nvdash_{\mathbf{B}}$$\mathsf{Triangle(asia)}$.

4 The Problems of Cautiousness and Trivialization

Despite the usefulness of three-valued paraconsistent logics for dealing with inconsistency, such entailments suffer from two major drawbacks. We discuss them in further details below.

[6] To simplify the presentation in the sequel, we would like to restrict our discussion to the propositional level. Thus we interpret here the atomic formulas as propositional atoms, and the universally quantified formulas as the set of their substitution instances.

The first drawback, pointed out in [17], is that three-valued paraconsistent logics are often too cautious to make intuitive classical entailments. Intuitively, e.g., when a knowledge base contains evidence that ψ is true and does not contain any evidence that $\neg\psi$ is true, one cannot infer that $\neg\psi$ is false, although it seems plausible.

Example 4. Let $\mathbf{B}_\bot$ be the logic obtained from $\mathbf{B}$ by adding the bottom element $\bot$ (which is assigned f by all valuations). Now note that in KB from Example 3, we have evidence that $\mathsf{Continent(europe)}$ is true, but *no* evidence that its negation is true. Despite this, we cannot infer that its negation is false: $KB \nvdash_{\mathbf{B}_\bot} \neg\mathsf{Continent(europe)} \to \bot$.

One of the solutions to the problem of cautious inference, proposed in [17], is to focus on preferential refinements of the basic inference relation, applying the principle of inconsistency minimization. The definitions in [17] apply only to the three-valued matrix $\mathcal{M}_{\mathbf{B_0}}$ from Example 1, but can be adapted as follows to the context of arbitrary finite-valued matrices:

Definition 6. *For a set of elements U and a well-founded (partial) order $\leq$ on U, we define: $min(\leq, U) = \{u \in U \mid \neg\exists u' \in U.u' \leq u \text{ and } u \not\leq u'\}$.*

Definition 7. *Let $\mathcal{M}$ be a matrix and $\leq$ a well-founded (partial) order on $\Lambda_{\mathcal{M}}$. Then $\Gamma \vdash^{\leq}_{\mathcal{M}} \psi$ if $min(\leq, mod_{\mathcal{M}}(\Gamma)) \subseteq mod_{\mathcal{M}}(\psi)$.*

Example 5. The following are two examples of order relations on $\Lambda_{\mathcal{M}_{\mathbf{B_0}}}$ proposed in [17,7], where the set of atoms in $\mathcal{L}$ is assumed to be finite (this restriction will be eliminated in our framework, see the discussion in Remark 1 below). For a valuation μ and a theory Γ, denote by $\mu!$ the set of atoms in $\mathcal{L}$, which are assigned $\top$ by μ. Let $\mu \leq_P \nu$ iff $\mu! \subseteq \nu!$, and $\mu \leq_{CP} \nu$ iff $|\nu!| \leq |\mu!|$.

Example 6. Let us return to Example 4 and see how using, e.g., $\vdash^{\leq_P}_{\mathbf{M}_{\mathbf{B}_\bot}}$ instead of $\vdash_{\mathbf{M}_{\mathbf{B}_\bot}}$ solves the problem discussed there. It is easy to see that for every $\nu \in min(mod_{\mathcal{M}_\mathbf{B}}(\Gamma), \leq_P)$, $\nu(\mathsf{Continent(europe)}) = t$. It now follows that $KB \vdash^{\leq_P}_{\mathbf{B}_\bot} \neg\mathsf{Continent(europe)} \to \bot$.

The second major drawback of paraconsistent entailments is that they are prone to trivialization. Of course, such entailments are no longer trivialized in the face of a classical inconsistency. However, the danger of trivialization remains, as it may happen that a knowledge base is no longer $\mathcal{M}$-consistent, as demonstrated by the following example.

Example 7. Consider the following toy knowledge base $EB_0 = \{\mathsf{EastBorder(africa, nile)}, \mathsf{Triangle(europe)}, \mathsf{Triangle(africa)}, \neg\mathsf{Triangle(africa)}\}$.
Suppose also that the researchers come to believe that all geographical facts specified in the "bird's-eye" of the world should be taken as consistent, in the sense that they do not have reason to expect to find any information contradicting this part of the text. This includes the facts that Europe and Africa have the form of a triangle. This can be captured by the following set

of implicit assumptions (using the consistency operator available in the logic **B**): $\mathcal{IA}_0 = \{\circ\mathsf{Triangle(africa)}, \circ\mathsf{Triangle(europe)}\}$. However, the knowledge base $KB_0 = EB_0 \cup \mathcal{IA}_0$ is $\mathcal{M}_{\mathbf{B}}$-inconsistent, resulting in its trivialization.

A method for avoiding trivialization for logics induced by denotational semantics was proposed in [2,4]. The idea is to apply distance-based minimization of non-satisfiability for "approximation" of $\mathcal{M}$-models of an $\mathcal{M}$-inconsistent knowledge base. The resulting entailment coincides with the original logic in case that the knowledge base is $\mathcal{M}$-consistent, and does not trivialize otherwise. The problem of trivialization, however, obtains also for the preferential entailments from Definition 7, to which the above method is not applicable. Indeed, preference is made there over models of the knowledge base, and as soon as the set of models is empty, we are again facing trivialization. In the next section we combine distance-based and preferentical approaches in a unified framework, allowing for trivialization-resistant preferential entailments.

5 Trivialization-Resistant Preferential Framework

In what follows we define a framework for incorporating preference criteria into paraconsistent entailments based on *arbitrary* many-valued matrices. The framework is based on the key notion of *order generators.* The intuitive idea is that the preference criteria are knowledge base dependent, that is each theory induces its own order relation on the space of valuations. This dependency is encapsulated in an order generator, which generates a different order for each theory[7]:

Definition 8. *Let $\mathcal{M}$ be a matrix.*

- *An order generator **O** for $\mathcal{M}$ is a function which for every theory Γ returns a well-founded partial order $\leq_\Gamma$ on $\Lambda_\mathcal{M}$.*
- *For some order generator **O** for $\mathcal{M}$, we write $\Gamma \vdash^{\mathbf{O}}_{\mathcal{M}} \psi$ if $min(\mathbf{O}(\Gamma), \Lambda_\mathcal{M}) \subseteq mod_\mathcal{M}(\psi)$.*

The key property of entailments of the form defined above is that for any matrix satisfying the following natural normality condition from [4], they are completely trivialization-resistant:

Definition 9. *A matrix $\mathcal{M}$ for $\mathcal{L}$ is* normal *if for every $\nu \in \Lambda_\mathcal{M}$, there is a $\mathcal{L}$-formula ψ, such that $\nu \not\models_\mathcal{M} \psi$.*

It is easy to see, e.g., that any matrix $\mathcal{M}$ in which a bottom element is definable, is normal. Moreover, so is any three-valued paraconsistent matrix for a language with $\neg$ and $\circ$, where these unary connectives are interpreted like in $\mathcal{M}_\mathbf{B}$ from Example 2.

Proposition 1 (non-trivialization). *Let $\mathcal{M}$ be a normal matrix for $\mathcal{L}$ and **O** an order generator. Then for every theory Γ, there is some ψ, such that $\Gamma \not\vdash^{\mathbf{O}}_{\mathcal{M}} \psi$.*

[7] See also the discussion in Remark 1 on order generators.

Proof: Let Γ be a theory. Since $\mathbf{O}(\Gamma)$ is a well-founded relation, $min(\mathbf{O}(\Gamma), \Lambda_{\mathcal{M}})$ is non-empty. Let $\mu \in min(\mathbf{O}(\Gamma), \Lambda_{\mathcal{M}})$. By the normality of $\mathcal{M}$, there is some ψ, such that $\mu \notin mod_{\mathcal{M}}(\psi)$. Hence $\Gamma \not\vdash^{\mathbf{O}}_{\mathcal{M}} \psi$.

Order generators will be constructed using (any number of) *profile settings*, consisting of two ingredients. The first is a *profile*, assigned to each valuation μ, which contains information to judge how "well-behaved" this valuation is with respect to a given knowledge base KB. Making this idea more concrete, a possible profile is a set of formulas, on which μ is "well-behaved" (the latter can mean different things: satisfaction of a formula, assigning a particular truth-value to a formula, etc.). To keep this set finite, we only use formulas which are "relevant" (this notion can also vary, and is defined below in precise terms) for KB. The second ingredient of a profile setting is some ordering on profiles (or finite sets of formulas). Since a profile is theory-dependent, we encapsulate this dependency (like in order generators) by defining *profile matching*:

Definition 10. Let $\mathcal{M}$ be a matrix for $\mathcal{L}$.

- A *profile matching* for $\mathcal{M}$ is a function $\mathcal{J}$, which given an $\mathcal{M}$-valuation μ and a theory Γ, returns a finite set of formulas, and which satisfies the boundedness condition: for every theory Γ, there is some number n_Γ, such that for all $\mu \in \Lambda_{\mathcal{M}}$, $|\mathcal{J}(\mu, \Gamma)| \leq n_\Gamma$.
- A *profile setting* for $\mathcal{M}$ is a pair $\mathcal{S} = \langle \mathcal{J}, \leq \rangle$, where $\mathcal{J}$ is a profile matching for $\mathcal{M}$ and $\leq$ a partial well-founded order on $\mathcal{T}_{\mathcal{L}}$.

A profile setting can then be used to define natural orderings among valuations. Moreover, we can define more fine-grained orderings by applying a common technique in choice theory based on lexicographic aggregation of orderings ([1]):

Definition 11. *Let $\mathcal{S}_1 = \langle \mathcal{J}_1, \leq_1 \rangle, \ldots, \mathcal{S}_n = \langle \mathcal{J}_n, \leq_n \rangle$ be profile settings for $\mathcal{M}$. $\mathbf{O}_{\mathcal{S}_1,\ldots,\mathcal{S}_n}$ is defined inductively as follows:*

- $\mathbf{O}_{\mathcal{S}_1}(\Gamma)(\mu, \nu)$ *iff* $\mathcal{J}_1(\mu, \Gamma) \leq_1 \mathcal{J}_1(\nu, \Gamma)$.
- *For* $n > 1$, $\mathbf{O}_{\mathcal{S}_1,\ldots,\mathcal{S}_n}(\Gamma)(\mu, \nu)$ *iff* $\mathbf{O}_{\mathcal{S}_1,\ldots,\mathcal{S}_{n-1}}(\Gamma)(\mu, \nu)$ *and if also* $\mathbf{O}_{\mathcal{S}_1,\ldots,\mathcal{S}_{n-1}}(\Gamma)(\nu, \mu)$*, then* $\mathcal{J}_n(\mu, \Gamma) \leq_n \mathcal{J}_n(\nu, \Gamma)$.

The intuitive idea behind lexicographic aggregation is that the preference expressed by $\mathcal{S}_{i-1}$ is more important than the one expressed by $\mathcal{S}_i$. Below we shall use this idea to capture the fact that approximating the behaviour of a model of the knowledge base is more important than approximating a classical behaviour. Note that the fact that profile settings are based on well-founded orders ensures that $\mathbf{O}_{\mathcal{S}_1,\ldots,\mathcal{S}_n}$ is indeed an order generator.

Now we provide some concrete examples of the two ingredients of a profile setting. Some natural examples of orders on $\mathcal{T}_{\mathcal{L}}$ are (i) $\Gamma_1 \leq_c \Gamma_2$ if $|\Gamma_1| \leq |\Gamma_2|$, and (ii) $\Gamma_1 \leq_i \Gamma_2$ if $\Gamma_1 \subseteq \Gamma_2$. To define profile matchings, we will need the notion of contexts from [4], capturing the intuitive idea of "relevance" for a given knowledge base.

Definition 12. A *context* is a finite set of formulas (i.e., an element of $\mathcal{T}_{\mathcal{L}}$). A *context generator* (for $\mathcal{L}$) is a function $\mathcal{G} : \mathcal{T}_{\mathcal{L}} \to \mathcal{T}_{\mathcal{L}}$, producing a context for every theory.

Example 8. Common examples for context generators are, e.g., the following functions defined for every theory Γ by $\mathcal{G}^{\mathsf{At}}(\Gamma) = \mathsf{Atoms}(\Gamma)$ (denoting the atoms of Γ), $\mathcal{G}^{\mathsf{ID}}(\Gamma) = \Gamma$, and $\mathcal{G}^{\mathsf{SF}}(\Gamma) = \mathsf{SF}(\Gamma)$ (denoting the subformulas of Γ).

Now we are ready to define the profile matching $\mathcal{J}_{\mathcal{Y}}^{\mathcal{G}}$ induced by a context generator $\mathcal{G}$ and a subset $\mathcal{Y}$ of truth-values of $\mathcal{M}$. The intuition behind the captured preference is that we want to minimize in some sense (according to the chosen order relation) the formulas "relevant" to our knowledge base (according to the generator $\mathcal{G}$) which are assigned values from $\mathcal{Y}$. For instance, by choosing $\mathcal{Y}$ to be the non-designated truth-values of $\mathcal{M}$, we "maximize the satisfaction" of the knowledge base, preferring valuations which approximate models of it. By choosing $\mathcal{Y}$ to contain the inconsistent truth-value $\top$, we "maximize classical reasoning", preferring valuations which best approximate classical valuations. The most natural approach seems to us having "model approximation" as a primary objective, which can then be refined by other criteria.

Definition 13. *Let $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$ be a matrix, $\mathcal{Y} \subseteq \mathcal{V}$, and $\mathcal{G}$ - a context generator. The profile maching $\mathcal{J}_{\mathcal{Y}}^{\mathcal{G}}$ is defined as follows: $\mathcal{J}_{\mathcal{Y}}^{\mathcal{G}}(\mu, \Gamma) = \{\psi \in \mathcal{G}(\Gamma) \mid \mu(\psi) \in \mathcal{Y}\}$.*

Example 9. Let $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$ be a matrix. Denote $\overline{\mathcal{D}} = \mathcal{V} \setminus \mathcal{D}$. Then $\mathcal{J}_{mod} = \mathcal{J}_{\overline{\mathcal{D}}}^{\mathcal{G}^{\mathsf{ID}}}$ is a profile matching for $\mathcal{M}$. Moreover, in the case that $\mathcal{V} = \{t, f, \top\}$, so are $\mathcal{J}_{con} = \mathcal{J}_{\{\top\}}^{\mathcal{G}^{\mathsf{Atoms}}}$ and $\mathcal{J}_{true} = \mathcal{J}_{\{f,\top\}}^{\mathcal{G}^{\mathsf{ID}}(\Gamma)}$.

Example 10. Let $\Gamma = \{p, \neg p, \circ p, q, \circ q, \neg(p \wedge q)\}$. Let $\mathcal{M}$ be the matrix $\mathcal{M}_{\mathbf{B}}$ from Example 2. To demonstrate the difference between various profile settings, consider the cases $\mathcal{S}_c^{con} = \langle \mathcal{J}_{con}, \leq_c \rangle$, $\mathcal{S}_i^{mod} = \langle \mathcal{J}_{mod}, \leq_i \rangle$, and $\mathcal{S}_c^{mod} = \langle \mathcal{J}_{mod}, \leq_c \rangle$. Note that the "relevant" formulas for $\mathcal{J}_{mod}$ are all the formulas from Γ, while the "relevant" formulas for $\mathcal{J}_{con}$ are only the atoms occurring in Γ. Computing the profiles of the $\mathcal{M}$-valuations, we obtain:

ν_i	p	q	$\neg(p \wedge q)$	$\neg p$	$\circ p$	$\circ q$	$\mathcal{J}_{mod}(\nu_i, \Gamma)$	$\mathcal{J}_{con}(\nu_i, \Gamma)$
ν_1	t	t	f	f	t	t	$\{\neg(p \wedge q), \neg p\}$	$\emptyset$
ν_2	t	$\top$	$\top$	f	t	f	$\{\neg p, \circ q\}$	$\{q\}$
ν_3	t	f	t	f	t	t	$\{q, \neg p\}$	$\emptyset$
ν_4	$\top$	t	$\top$	$\top$	f	t	$\{\circ p\}$	$\{p\}$
ν_5	$\top$	$\top$	$\top$	$\top$	f	f	$\{\circ p, \circ q\}$	$\{p, q\}$
ν_6	$\top$	f	t	$\top$	f	t	$\{q, \circ p\}$	$\{p\}$
ν_7	f	t	t	t	t	t	$\{p\}$	$\emptyset$
ν_8	f	$\top$	t	t	t	f	$\{p, \circ q\}$	$\{q\}$
ν_9	f	f	t	t	t	t	$\{p, q\}$	$\emptyset$

So we have $min(\mathbf{O}_{\mathcal{S}_i^{mod}}, \Lambda_{\mathcal{M}}) = \{\nu_1, \nu_2, \nu_3, \nu_4, \nu_7\}$, while $min(\mathbf{O}_{\mathcal{S}_c^{mod}}, \Lambda_{\mathcal{M}}) = \{\nu_4\}$. Hence, e.g., $\Gamma \not\vdash^{\mathbf{O}_{\mathcal{S}_i^{mod}}} \circ q$, while $\Gamma \vdash^{\mathbf{O}_{\mathcal{S}_c^{mod}}} \circ q$. The cautiousness of

$\vdash^{\mathbf{O}_{\mathcal{S}_i^{mod}}}$ can be recovered by using an aggregation-based order. For instance, $min(\mathbf{O}_{\mathcal{S}_i^{mod},\mathcal{S}_c^{con}}, \Lambda_{\mathcal{M}}) = \{\nu_1, \nu_3, \nu_7\}$, and so $\Gamma \vdash^{\mathbf{O}_{\mathcal{S}_i^{mod},\mathcal{S}_c^{con}}} \circ q$.

Below we show some basic properties of the defined entailments. First of all, for $\mathcal{M}$-consistent knowledge bases, the two basic entailments coincide with the logic $\vdash_{\mathcal{M}}$ (while for inconsistent ones they behave differently, as we have just seen in Example 10 above).

Proposition 2. *Let $\mathcal{M}$ be a matrix and Γ - an $\mathcal{M}$-consistent theory. Let $\mathcal{S} = \langle \mathcal{J}_{mod}, \leq \rangle$, where $\leq$ is either $\leq_i$ or $\leq_c$. Then $\Gamma \vdash_{\mathcal{M}} \psi$ iff $\Gamma \vdash_{\mathcal{M}}^{\mathbf{O}_{\mathcal{S}}} \psi$*

The basic entailment induced by $\mathcal{J}_{mod}$ is particularly well-behaved:

Proposition 3. *Let $\mathcal{S}_1 = \langle \mathcal{J}_1, \leq_1 \rangle$, where $\mathcal{J}_1 = \mathcal{J}_{mod}$ and $\leq_1$ is either $\leq_i$ or $\leq_c$. Then $\vdash_{\mathcal{M}}^{\mathbf{O}_{\mathcal{S}}}$ is a cautious consequence relation (with respect to $\mathcal{M}$).*

Proposition 4 (decidability). *We say that a profile setting $\mathcal{S} = \langle \mathcal{J}, \leq \rangle$ for a finite matrix $\mathcal{M}$ is* simple *if $\mathcal{J}$ has the form from Definition 13, and $\leq$ is either $\leq_c$ or $\leq_i$. For simple profile settings $\mathcal{S}_1 = \langle \mathcal{J}_1, \leq_1 \rangle, \ldots, \mathcal{S}_n = \langle \mathcal{J}_n, \leq_n \rangle$ for a finite matrix $\mathcal{M}$, the question whether $\Gamma \vdash_{\mathcal{M}}^{\mathbf{O}_{\mathcal{S}_1,\ldots,\mathcal{S}_n}} \psi$ is decidable.*

Remark 1. In addition to defining new paraconsistent entailments, many natural cases of previously studied paraconsistent formalisms can be simulated within our framework, which is perhaps an indication for the naturality of the above definitions. Below are some examples:

1. For $\mathcal{S}_c = \langle \mathcal{J}_{mod}, \leq_c \rangle$ and any matrix $\mathcal{M}$, $\vdash_{\mathcal{M}}^{\mathbf{O}_{\mathcal{S}}}$ coincides with the distance-based paraconsistent entailment $\mid\!\sim_{\langle \mathsf{Atoms},\mathsf{d},\Sigma \rangle}$ from [4], where d is the standard distance on $\{t, \top, f\}$ (where, e.g., $\mathsf{d}(t, f) = 1$ and $\mathsf{d}(t, \top) = \mathsf{d}(f, \top) = 0.5$).
2. For $\mathcal{S}_i = \langle \mathcal{J}_{mod}, \leq_i \rangle$ and the classical matrix $\mathcal{M}_{cl}$, $\vdash_{\mathcal{M}}^{\mathbf{O}_{\mathcal{S}}}$ coincides with the (propositional fragment) of the cautious entailment based on maximally consistent subsets of [13,14].
3. Let $\mathcal{M}$ be any paraconsistent three-valued matrix. Let $\mathcal{S}_0$ be either $\mathcal{S}_c$ or $\mathcal{S}_i$. Then the following holds for any $\mathcal{M}$-consistent theory Γ and any ψ: (i) For $\mathcal{S}_1 = \langle \mathcal{J}_{\top}^{\mathsf{At}}, \leq_i \rangle$, $\Gamma \vdash_{\mathcal{M}}^{\mathbf{O}_{\mathcal{S}_0,\mathcal{S}_1}} \psi$ iff $\Gamma \models_P \psi$, (ii) For $\mathcal{S}_1 = \langle \mathcal{J}_{\top}^{\mathsf{At}}, \leq_c \rangle$, $\Gamma \vdash_{\mathcal{M}}^{\mathbf{O}_{\mathcal{S}_0,\mathcal{S}_1}} \psi$ iff $\Gamma \models_{CP} \psi$, (iii) For $\mathcal{S}_1 = \langle \mathcal{J}_{\{\top,f\}}^{\mathsf{ID}}, \leq_i \rangle$, $\Gamma \vdash_{\mathcal{M}}^{\mathbf{O}_{\mathcal{S}_0,\mathcal{S}_1}} \psi$ iff $\Gamma \models_{BS}^{\Gamma} \psi$, (iv) For $\mathcal{S}_1 = \langle \mathcal{J}_{\{\top,f\}}^{\mathsf{ID}}, \leq_c \rangle$, $\Gamma \vdash_{\mathcal{M}}^{\mathbf{O}_{\mathcal{S}_0,\mathcal{S}_1}} \psi$ iff $\Gamma \models_{CBS}^{\Gamma} \psi$, where $\models_P, \models_{CP}, \models_{BS}^{\Gamma}, \models_{CBS}^{\Gamma}$ are the entailments defined in [17,7] (see also Example 5), and we assume that the set of atoms of $\mathcal{L}$ is finite and is equal to $\mathsf{Atoms}(\Gamma)$.
   It should be noted that there are a number of aspects, in which the above entailments differ from their corresponding counterparts $\models_P, \models_{CP}, \models_{BS}^{\Gamma}$ and $\models_{CBS}^{\Gamma}$. First of all, while the former coincide with the latter for $\mathcal{M}$-consistent premises, they do not trivialize in the presence of $\mathcal{M}$-inconsistency. Secondly, due to the incorporation of context generators, the former do not depend on the (rather restrictive) assumption of finiteness of the underlying language. Finally, the entailments $\models_{BS}^{\Gamma}$ and $\models_{CBS}^{\Gamma}$ are called in [17] non-standard relations, as their definition depends on the belief base Γ under consideration.

The incorporation of the notion of order generators, however, solves this problem, as the dependency of the belief base is encapsulated in it, so the corresponding relations can be thought of as standard in the sense of [17].

Example 11. Revisiting the knowledge base KB_0 from Example 7, let us now apply, e.g., the entailment $\vdash^{\mathbf{O}_{\mathcal{S}_0,\mathcal{S}_1}}_{\mathcal{M}_{\mathbf{B}_\perp}}$ for $\mathcal{S}_0 = \langle \mathcal{J}_{mod}, \leq_c \rangle$ and $\mathcal{S}_1 = \langle \mathcal{J}_{con}, \leq_i \rangle$ (where $\mathcal{M}_{\mathbf{B}_\perp}$ is the matrix from Example 4). The entailment is trivialization-resistant, since despite the fact that KB_0 is not $\mathcal{M}_{\mathbf{B}_\perp}$-consistent, we cannot infer from it $\mathsf{Triangle(africa)}$ (while we can still infer $\mathsf{Triangle(europe)}$). Moreover, the entailment is less cautious than $\vdash_{\mathcal{M}_{\mathbf{B}_\perp}}$ or $\vdash^{\mathbf{O}_{\mathcal{S}_0}}_{\mathcal{M}_{\mathbf{B}_\perp}}$, since, as opposed to the latter, we can infer in it, e.g., $\neg\mathsf{EastBorder(africa, nile)} \rightarrow \perp$ from KB_0 (i.e, $\mathsf{EastBorder(africa, nile)}$ is consistently true).

6 Summary and Further Research

In this paper we have proposed an abstract framework for incorporating various "epistemic" preference criteria into paraconsistent entailments, which are useful for overcoming problems such as cautiousness and trivialization. The framework has a number of attractive properties. Most importantly, it allows for defining *trivialization-resistant* variants of paraconsistent entailments based on a many-valued matrix, including the case of preferential ones, which to the best of our knowledge has not been treated before in this context. Moreover, it is *general*, as nothing is assumed about the underlying matrix, and the preference criteria definable in it also have a general form, which can also be further generalized. The framework is also *modular*, as it has a built-in mechanism for easily combining any finite number of preference criteria. We believe that this framework can be a useful tool for the design of paraconsistent knowledge bases. The reason is that for a problem coming from a new, not yet studied domain (like the domain of descriptive ancient geography), it is often hard to choose the right paraconsistent approach (e.g., fixing the matrix, choosing the most natural preference criteria, etc.). In many cases the right choice becomes evident only after some experimental data is available. Our framework, which combines many paraconsistent approaches and allows for a smooth transition between them, can be used for their comparative study.

The most immediate research directions include investigating the logical properties of the entailments defined in our framework, as well as of their computational complexity (from the results of [17] for some special cases, it is clear that intractability is expected for the general case). It should be noted, however, that the current framework aims at generality rather than efficiency, while for improving the latter, entailment-specific optimizations can be employed. For practical applications, the results should also be extended to the first-order case. Concerning the considered domain of ancient geography, it seems promising to combine our approach for handling inconsistent knowledge bases with methods for handling vague geographic concepts along the lines of [6].

References

1. Andréka, H., Ryan, M., Schobbens, P.-Y.: Operators and laws for combining preference relations. Journal of Logic and Computation 12(1), 13–53 (2002)
2. Arieli, O.: Distance-based paraconsistent logics. International Journal of Approximate Reasoning 48(3), 766–783 (2008)
3. Arieli, O., Avron, A.: General patterns for nonmonotonic reasoning: from basic entailments to plausible relations. Logic Journal of the IGPL 8(2), 119–148 (2000)
4. Arieli, O., Zamansky, A.: Inconsistency-Tolerance in Knowledge-Based Systems by Dissimilarities. In: Lukasiewicz, T., Sali, A. (eds.) FoIKS 2012. LNCS, vol. 7153, pp. 34–50. Springer, Heidelberg (2012)
5. Avron, A., Arieli, O., Zamansky, A.: Ideal paraconsistent logics. Studia Logica 99(1-3), 31–60 (2011)
6. Bennett, B., Mallenby, D., Third, A.: An ontology for grounding vague geographic terms. In: Eschenbach, C., Gruninger, M. (eds.) Proceedings of the 5th International Conference on Formal Ontology in Information Systems (FOIS 2008). IOS Press (2008)
7. Besnard, P., Schaub, T.: Circumscribing inconsistency. In: Proc. of IJCAI 1997, pp. 150–155 (1997)
8. Carnielli, W.A., Coniglio, M.E., Marcos, J.: Logics of formal inconsistency. In: Gabbay, D.M., Guenthner, F. (eds.) Handbook of Philosophical Logic, 2nd edn., vol. 14, pp. 15–107. Springer (2007)
9. Carnielli, W.A., Marcos, J., de Amo, S.: Formal inconsistency and evolutionary databases. Logic and Logical Philosophy 8, 115–152 (2000)
10. da Costa, N.C.A.: On the theory of inconsistent formal systems. Notre Dame Journal of Formal Logic 15, 497–510 (1974)
11. Gabbay, D.: Theoretical foundation for non-monotonic reasoning, Part II: Structured non-monotonic theories. In: Proc. SCAI 1991. IOS Press (1991)
12. Gabbay, D., Hunter, A.: Making inconsistency respectable: A logical framework for inconsistency in reasoning. In: Fundamentals of Artificial Intelligence. Springer (1992)
13. Grant, J., Subrahmanian, V.S.: Reasoning about inconsistent knowledge bases. In: IEEE TKDE, vol. 7(1), pp. 177–189 (1995)
14. Grant, J., Subrahmanian, V.S.: Applications of paraconsistency for data and knowledge bases. Synthese 125(1/2), 121–132 (2000)
15. Ilyuschetchkina, E.: Das Weltbild des Dionysios Periegetes. In: Die Vermessung der Oikumene / Mapping the Oikumene. De Gruyter, Berlin (forthcoming, 2012)
16. Konieczny, S., Lang, J., Marquis, P.: DA2 merging operators. Artificial Intelligence 157(1-2), 49–79 (2004)
17. Konieczny, S., Marquis, P.: Three-Valued Logics for Inconsistency Handling. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 332–344. Springer, Heidelberg (2002)
18. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
19. Makinson, D.: General patterns in nonmonotonic reasoning. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3, pp. 35–110 (1994)
20. de Amo, S., Carnielli, W.A., Marcos, J.: A Logical Framework for Integrating Inconsistent Information in Multiple Databases. In: Eiter, T., Schewe, K.-D. (eds.) FoIKS 2002. LNCS, vol. 2284, pp. 67–84. Springer, Heidelberg (2002)
21. Shoham, Y.: Reasoning about Change. MIT Press (1988)

DebateWEL: An Interface for Debating with Enthymemes and Logical Formulas*

Julien Balax, Florence Dupin de Saint-Cyr, and David Villard

IRIT, Université Paul Sabatier, 31062 Toulouse Cedex 9, France
julienbalax@gmail.com, florence.dupin@irit.fr, david0271@ymail.com

Abstract. The DebateWEL G.U.I. allows two players to exchange logical formulas and enthymemes in a persuasion debate game. DebateWEL protocol, described in [1], ensures consistency of each player with respect to himself and common-knowledge thanks to a SAT solver [2] which is called directly from the SAToulouse interface [3]. The game ends either because the players have found an agreement or because the limit of time has been reached (failure of the debate).

Being persuasive is very useful in many situations. Indeed good orators are considered to be very clever, hence an interesting challenge for AI is to be able to design artificial persuasive orators. Before proposing such artificial agents, it is necessary to define the framework in which they are going to play.

Persuasion dialogs are particular dialogs in which one agent aims at convincing others that a first assertion (called the subject of the dialog) holds. In order to design a framework in which a persuasion dialog can take place, it is necessary to define the moves that each agent is allowed to make. This definition is called a *protocol*. In persuasion dialogs, the possible types of moves are mainly assertions and challenges while in other kinds of dialogs (*e.g.* negotiations dialogs) moves like requirements or proposals are allowed. Apart from its type, a move is also characterized by its content. In a persuasion dialog this content is an assertion or an argument (*i.e.*, a rational utterance explaining the reasons for a given claim). Here are the specificities of DebateWEL protocol:

– There is no information about the state of mind of the agents, only what is publicly said is considered.
– The contents of the moves are based on classical propositional logic.
– Arguments are not abstract entities but explicit pairs (S, φ) where the support S is a set of formulas and the claim φ is a formula. Moreover, pairs in which S is not a proof of φ are allowed. If something is missing to prove φ from S or if φ is too vague then (S, φ) is called *enthymeme for* a given perfect argument that should have a more complete support or a more precise claim. As explained in [1,4,5]), enthymemes are very common in human dialogs.
– At the beginning of the dialog the agents agree on *a common knowledge* on which they can base their future statements, it is expressed under the form of formulas and enthymemes (they can agree on a set of approximate arguments

* This work was funded by the ANR project LELIE on risk analysis and prevention (http://www.irit.fr/recherches/ILPL/lelie/accueil.html).

that they consider as sufficient "proofs"). This common knowledge may only increase during the dialog.

– The protocol is *flexible*: any agent can take is turn as soon as the previous move is finished (even by himself), and can utter *any* move provided that it does not introduce any *self-contradiction* or *repetition*, and that, before the end, *all the commitments* induced by the moves of his adversary are fulfilled.
– The dialog ends either with the victory of one agent when his adversary has agreed with him, or with a failure when they did not succeed to fulfill their commitment in time (either expressed in seconds or in terms of number of symbols used). DebateWEL is designed for inciting agents rather to agree than to reach a failure of the dialog: the score is high for a persuasive player, middle for a persuaded player and bad if the dialog fails.

Let us recall definitions from [1]: $\mathscr{L}$ being a propositional language, an *approximate argument* is a pair (S, φ) where $S \subseteq \mathscr{L}$ and $\varphi \in \mathscr{L}$. A *logical argument* (S, φ) is such that: (1) $S \subset \mathscr{L}$, (2) $S \nvdash \bot$, (3) $S \vdash \varphi$, and (4) $\nexists S' \subset S$ s.t. $S' \vdash \varphi$. (S, φ) and (S', φ') being approximate arguments, (S', φ') *completes* (S, φ) iff (1) $S \subset S'$ and $\varphi = \varphi'$ or (2) $S \subseteq S'$ and $\{\varphi'\} \cup S' \vdash \varphi$ and $\varphi \neq \varphi'$. a is an *enthymeme for* a' iff a' is a logical argument and a' completes a (a is said *incomplete*).

We consider a set of eleven speech acts, the six usual speech acts used in persuasion dialog: Accept for accepting a formula, Argue for uttering an argument (which maybe incomplete), Assert for uttering a formula, Challenge for asking for an argument explaining a formula, Close to end the dialog and Retract for removing an asserted formula. Two speech acts that are specific for enthymemes are added to this set, namely Quiz and Agree (proposed by [6]) for asking for a completion of an argument and for agreeing with the fact that a given pair (S, φ) is sufficient to convince that φ holds. We also add three more moves: Quizlink enabling to ask for a completed argument that relates its claim to the subject of the dialog, Replace allowing to complete an argument and Dismantle for retracting an argument. In [1], it is assumed that speech acts commit either the utterer (to be consistent and to be able to explain himself when challenged) and the hearer (to acknowledge that he agrees with the utterances of his adversary (unless they have been retracted)).

DebateWEL interface handles these commitments with a commitment store [7] divided into three parts (see Figure 1). On the center, the common knowledge part is divided into formulas and arguments. On each side of the screen, the commitment stores of each agent are divided into three parts: propositional formulas and approximate arguments asserted by the agent and commitments towards the other agent, *i.e.*, the requests to which he should answer. The preconditions of each move are translated into consistency checks (done by SAT4j) within the commitment store while the effects of the moves are addition to or removal from the commitment store (following the precise definitions of each move given in [1]).

In the screen shot of Figure 1, the first player is named Schopenhauer, he has asserted $drama \rightarrow supreme$ meaning "In drama, English are supreme", the common knowledge is $\{opera \rightarrow music, opera\}$ meaning that the two players agree on the fact that "opera is music" and that "opera exists". After this first

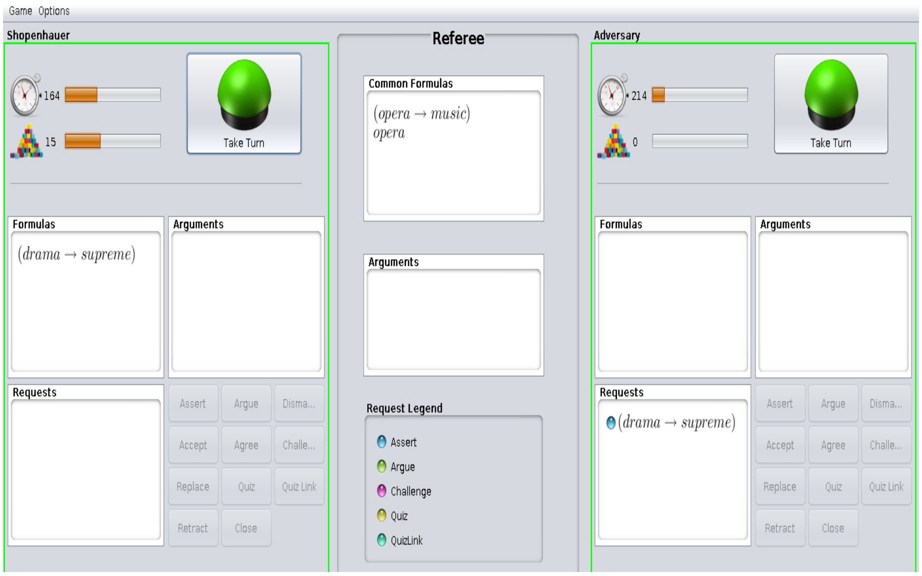


Fig. 1 Screen shot of DebateWEL main screen

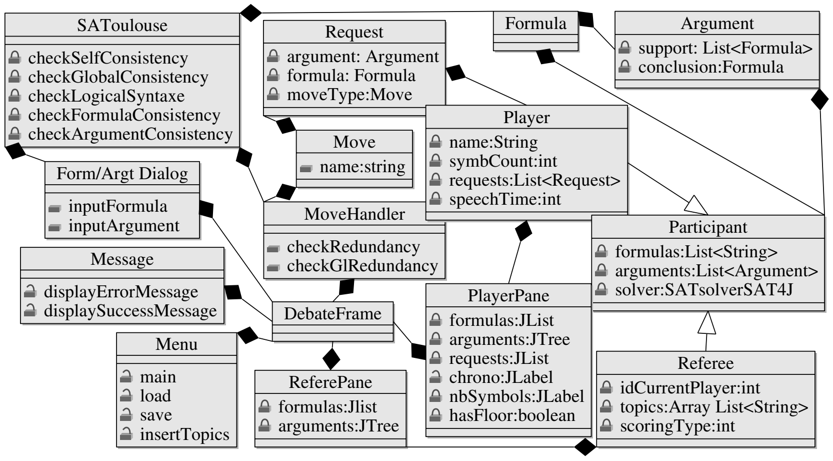


Fig. 2 Simplified Architecture of DebateWEL

move, the second player named "Adversary" is commited to Accept the formula *drama* → *supreme* (as long as Schopenhauer has not retracted this assertion). Schopenhauer has been permitted to do it because it is consistent with what he had already said (nothing) and with common knowledge. Checking consistency is done by using SAT4j. The interface for entering formulas is based on SAToulouse as shown on Figure 2 which represents the architecture of DebateWEL.

DebateWEL has several software features:

– The hand is materialized by a buzzer, when it is green the agent can buzz. After buzzing, the buzzer becomes orange and the current agent can play its

move while his adversary must wait (the moves that are not possible are grayed button, if the formula or argument entered is not consistent or redundant then an error message is delivered). In Figure 1, the two agents can buzz, the quicker to buzz will be allowed to play.
– In order to help the players, they can choose to load one or several files containing formulas concerning a given topic, it is then possible to select directly these formulas instead of entering them, moreover it is also possible to select among formulas that have already been asserted in the dialog.
– A concise view of arguments is proposed: by default, only the claim is shown (the support can be fold and unfold simply by clicking on it).
– The type of the move of a request is symbolized by a colored bullet and the meaning of the colors is displayed in the center of the screen (see Figure 1).
– The score at the end of the dialog is based on the number of the player's formulas present in the common knowledge at the end of the dialog, a big penalty is associated to a player who has not fulfilled his commitments.
– There are two options for the limitation of the dialog: either the time taken by the agent or the number of symbols used in his utterances.
– At the end of the game the common-knowledge is stored in a file.
– DebateWEL enables the user to load an old game, thus it is possible to stop the dialog and save it (in order to continue later).

In the near future, we plan to develop an artificial player on the basis of an A* algorithm, and we also want to make this application available on the web. Indeed if this application is used, then it can help to collect new arguments (hence enrich a set of argument benchmarks, something that is really needed in the field of Argumentation theory).

References

1. Dupin de Saint-Cyr, F.: Handling Enthymemes in Time-Limited Persuasion Dialogs. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS (LNAI), vol. 6929, pp. 149–162. Springer, Heidelberg (2011)
2. Le Berre, D., Parrain, A.: The Sat4j library, release 2.2. Journal on Satisfiability, Boolean Modeling and Computation, 59–64 (2010)
3. Gasquet, O., Schwarzentruber, F., Strecker, M.: SATOULOUSE: The Computational Power of Propositional Logic Shown to Beginners. In: Blackburn, P., van Ditmarsch, H., Manzano, M., Soler-Toscano, F. (eds.) TICTTL 2011. LNCS, vol. 6680, pp. 77–84. Springer, Heidelberg (2011)
4. Besnard, P., Hunter, A.: A logic-based theory of deductive arguments. Artificial Intelligence 128(1-2), 203–235 (2001)
5. Hunter, A.: Real arguments are approximate arguments. In: Proceedings of the 22nd National Conference on Artificial Intelligence, pp. 66–71. AAAI Press (2007)
6. Black, E., Hunter, A.: Using enthymemes in an inquiry dialogue system. In: Padgham, L., Parkes, D. (eds.) Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - AAMAS, International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, pp. 437–444 (2008)
7. Hamblin, C.: Fallacies. Methuen, London (1970)

OMiGA: An Open Minded Grounding On-The-Fly Answer Set Solver

Minh Dao-Tran, Thomas Eiter, Michael Fink, Gerald Weidinger, and Antonius Weinzierl[⋆]

Institute of Information Systems, Vienna University of Technology
Favoritenstraße 9-11, A-1040 Vienna, Austria
{dao,eiter,fink,weidinger,weinzierl}@kr.tuwien.ac.at

Abstract. We present a new solver for Answer-Set Programs whose main features include grounding on-the-fly and readiness for use in solving distributed answer-set programs. The solver is implemented in Java and uses an underlying Rete network for propagation. Initial experimental results show the benefit of using Rete for this purpose, but also exhibit the need for learning in the presence of grounding on-the-fly.

1 Motivation

Answer-set programming (ASP), based on [2], is a paradigm where a problem is encoded as a program that is a set P of nonmonotonic rules, facts, and constraints. The program P is then given to a solver that searches for specific interpretations called answer sets of P such that all rules and constraints are satisfied.

Example 1. Consider a variant of cut-set: given a graph, remove one edge and compute the reachability of the modified graph. An ASP encoding is $P =$

$$\left\{\begin{array}{ll} r_1\colon \mathit{del}(X,Y) \leftarrow e(X,Y), \text{not } k(X,Y). & r_4\colon \mathit{reach}(X,Y) \leftarrow k(X,Y). \\ r_2\colon \quad k(X,Y) \leftarrow e(X,Y), \mathit{del}(X_1,Y_1), X_1 \neq X. & r_5\colon \mathit{reach}(X,Z) \leftarrow \mathit{reach}(X,Y), \\ r_3\colon \quad k(X,Y) \leftarrow e(X,Y), \mathit{del}(X_1,Y_1), Y_1 \neq Y. & \qquad\qquad \mathit{reach}(Y,Z). \end{array}\right\}.$$

Rule r_1 uses negation to create a choice point for each edge; r_2 and r_3 ensure that only one edge is removed. Finally, r_4 and r_5 compute reachability among the edges kept. Observe that only r_1 requires a guess and if for one ground instance the rule is guessed applicable, an answer set can be found using only propagation.

For humans, abstract rules as above are easy to understand and process, while traditional answer-set solvers can not handle them. They apply (pre-)grounding, i.e., substituting all variables with the actual values they might take. So for each combination of edges, a new rule is generated where the variables are eliminated.

As the ground program might be very large compared to the non-ground program, techniques like intelligent grounding have been developed in order to restrict the rules which must be grounded. But for distributed systems, these techniques conflict with

[⋆] This work has been funded by the Vienna Science and Technology Fund (WWTF) project ICT 08-020 and by the Austrian Science Fund (FWF) project P20841.

information hiding and the fact that not all information is known when pre-grounding must take place. As open domains become more important, the same problem arises even in non-distributed settings.

To avoid pre-grounding the answer set solvers GASP [5] and ASPeRiX [3,4] have been developed, both based on the same evaluation technique which has similarities to the SModels [6] approach. We reconsider this technique since it can be improved with well-suited data structures like Rete [1], and we develop a solver, called OMiGA, with an eye on distributed systems.

2 Methods and Techniques

Intuitively, OMiGA keeps a partial interpretation $I = (I^+, I^-)$ containing all ground atoms considered true (resp., false) in I^+ (resp., I^-). If there is a non-ground rule r with a valuation V of variables such that all positive atoms of the grounded rule $r[V]$ are true, i.e., $B^+(r[V]) \subseteq I^+$, and all negative atoms of $r[V]$ are false, i.e., $B^-(r[V]) \subseteq I^-$, then the head of r is grounded and derived, i.e., I^+ is extended by $H(r[V])$. This *propagation* is repeated until a fixpoint is reached.

If propagation is finished and a valuation V of a rule r exists such that $B^+(r[V]) \subseteq I^+$ and $B^-(r[V]) \cap I^+ = \emptyset$, then $r[V]$ possibly is applicable in some answer set; we have to *guess* whether it is the case or not. If yes, I^- is extended by $B^-(r[V])$ and I^+ by $H(r[V])$, making $r[V]$ applicable. If not, conceptually a constraint is added ensuring that $r[V]$ does not become applicable.

At any point, if a constraint is violated or $I^+ \cap I^- \neq \emptyset$, the current guessing branch is inconsistent, backtracking is issued, and subsequently a guess is inverted.

To minimize time needed to find a valuation V of r for propagation or guessing, we employ a Rete-like network, where nodes represent parts of the input program P. Each node has an associated working memory (WM) in which all valuations that are true under I are stored. There are *basic nodes*, *join nodes*, and *head nodes*, representing facts, subsets of rule body atoms, and rule heads, respectively.

These nodes are connected according to P, e.g., for rule r_5 in Example 1 there exists one join-node connected twice to the basic node of *reach*, and it is connected to the head node for r_5. If a new fact is added to its corresponding basic node, it is propagated through the network and thus only processed where it might lead to new results. To represent I^+ and I^- there are two basic nodes for each predicate p. For guessing, the Rete network is built such that each rule has a specific join-node whose WM contains valuations V for which $B(r[V]) \subseteq I^+$, i.e., finding a rule whose applicability can be guessed amounts to selecting from that node.

3 Solver Architecture

The basic architecture of OMiGA is shown in Figure 1. Its input is an answer-set program that is parsed and rewritten to optimize guessing (in a standard way, introducing new atoms representing satisfaction of rule bodies). From these rules, the Rete Builder then creates the Rete network that is later used in the central component of our solver. Here, a Manager controls the depth-first search for answer sets as follows: Rete is repeatedly

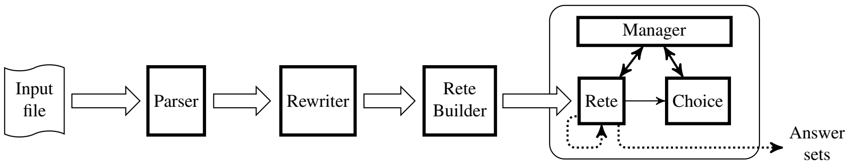
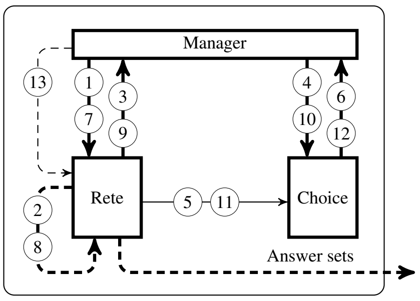


Fig. 1. System architecture



$(1, 7)$	M (Manager) triggers R (Rete) to propagate
(2)	R propagates until a fix point and informs M (3) $I^+ = \{e(1,2), e(2,3), e(3,4), e(2,4)\}$
$(4, 10)$	M triggers C (Choice) to make a choice
(5)	C looks up Rete network, finds a choice $r_1[X/2, Y/3]$
(6)	C makes a choice: r_1 is applicable $I^+ = I^+ \cup \{del(2,3)\}, I^- = \{k(2,3)\}$
(8)	R propagates until a fix point and informs M (9) $I^+ = I^+ \cup \left\{ \begin{array}{l} k(1,2), k(3,4), k(2,4), reach(1,4), \\ reach(1,2), reach(3,4), reach(2,4) \end{array} \right\}$
(11)	C looks up Rete network, finds no more choice
(12)	C informs M of no more choice
(13)	M requests R to print the answer

Fig. 2. Illustration of the propagation-and-guess evaluation

triggered to evaluate all rules and derive new rule heads. If no new atoms can be derived, i.e., the propagation phase is finished, then the Choice component selects a rule whose positive body is fulfilled, a new decision level is entered and the rule is guessed to be applicable. This process of propagation and guess is repeated until either no more choices can be made, i.e., an answer set is found and printed, or an inconsistent state is reached. In the latter case, backtracking is done and the last guess is inverted. Figure 2 details a run for Example 1, on a graph with edges $E = \{e(1,2), e(2,3), e(3,4), e(2,4)\}$.

4 Evaluation

We compare `OMiGA` to clingo[1] 3.0.4, DLV[2] 2011-12-21, and ASPeRiX 0.2.4; we omit GASP, as it is known to be slower than ASPeRiX [4] and time measurement due to entanglement with Prolog is ambiguous. The instances are: (i) reachability, a positive program for graph reachability (from the 2009 ASP Contest), with close to $24K$ and $700K$ edges, (ii) 3-colorability on sparse graphs of size 10 and 20, (iii) locstrat, a benchmark program from [4] with 200 and 400 nodes, (iv) cutedge, our running example on a random graph with 100 nodes and close to $2.8K$ resp. $4.9K$ edges.

As Table 1 shows, the use of Rete pays off as it stores partial joins and hence reduces time for propagation. `OMiGA` is consistently faster than ASPeRiX, except for locstrat where ASPeRiX uses must-be-true propagation while `OMiGA` only propagates rule heads. Also note that due to the use of Java and the building of the Rete network, our solver has an increased startup-time. On NP-hard problems like 3-colorability, the issue

[1] `http://potassco.sourceforge.net/`
[2] `http://www.dlvsystem.com`

Table 1. Comparative systems evaluation (**c**: clingo, **d**: DLV, **a**: ASPeRiX, **o**: OMiGA)

	reachability		3-colorability		locstrat		cutedge	
	24K	**700K**	**10**	**20**	**200**	**400**	**2.8K**	**4.9K**
c	0.33	5.00	0.00 0.00	0.00 0.00	0.46 0.46	2.06 2.05	25.85 27.34	75.06 79.26
d	0.44	4.56	0.00 0.00	0.00 0.00	5.88 5.67	46.93 47.78	107.07 214.67	301.54 600.08
a	2.84	—	0.01 1.06	— —	0.01 0.08	0.07 0.33	1.70 16.70	4.62 46.02
o	1.20	15.53	0.16 0.35	1.97 5.37	0.38 0.65	0.61 1.32	0.77 3.05	0.85 3.53

Running time in seconds, left: first answer, right: first 10 answers (if applicable).

of missing learning from conflicts is visible as clingo (using nogood learning) and DLV (using backjumping and look-back heuristics) perform extremely well. For non-ground ASP solving, learning is an open issue.

Comparing the cutedge benchmark with reachability, the effect of intelligent pre-grounding is evident since for the positive reachability instances, the pre-grounder can efficiently evaluate the programs. If those pre-grounding strategies are not possible due to only one guessing rule like in cutedge, the propagation becomes ineffective, very much in contrast to a Rete-based propagation.

5 Ongoing Work and Conclusion

We have presented OMiGA,[3] a new grounding on-the-fly solver for answer set programs, which also can be employed in a distributed setting where nodes contain programs that can access atoms at other nodes, by exchanging only the Manager component. As our experimental results show, Rete pays off and makes OMiGA outperform clingo and DLV for propagation-intense instances. Future work on backtracking, conflict-driven learning, and extended propagation is planned.

References

1. Forgy, C.L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence 19(1), 17–37 (1982)
2. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. 9(3/4), 365–386 (1991)
3. Lefèvre, C., Nicolas, P.: A First Order Forward Chaining Approach for Answer Set Computing. In: Erdem, E., Lin, F., Schaub, T. (eds.) LPNMR 2009. LNCS, vol. 5753, pp. 196–208. Springer, Heidelberg (2009)
4. Lefèvre, C., Nicolas, P.: The First Version of a New ASP Solver: ASPeRiX. In: Erdem, E., Lin, F., Schaub, T. (eds.) LPNMR 2009. LNCS, vol. 5753, pp. 522–527. Springer, Heidelberg (2009)
5. Palù, A.D., Dovier, A., Pontelli, E., Rossi, G.: Gasp: Answer set programming with lazy grounding. Fundam. Inform. 96(3), 297–322 (2009)
6. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artif. Intell. 138(1-2), 181–234 (2002)

[3] http://www.kr.tuwien.ac.at/research/systems/omiga

The Multi-Engine ASP Solver ME-ASP

Marco Maratea[1], Luca Pulina[2], and Francesco Ricca[3]

[1] DIBRIS, Univ. degli Studi di Genova, Viale F.Causa 15, 16145 Genova, Italy
[2] POLCOMING, Univ. degli Studi di Sassari, Viale Mancini 5, 07100 Sassari, Italy
[3] Dipartimento di Matematica, Univ. della Calabria, Via P. Bucci, 87030 Rende, Italy
marco@dist.unige.it, lpulina@uniss.it, ricca@mat.unical.it

Abstract. In this paper we describe the new system ME-ASP, which applies machine learning techniques for inductively choosing, among a set of available ones, the "best" ASP solver on a per-instance basis. Moreover, we report the results of some experiments, carried out on benchmarks from the "System Track" of the 3rd ASP Competition, showing the state-of-the-art performance of our solver.

1 Introduction

Answer Set Programming [7] (ASP) is a truly-declarative programming paradigm proposed in the area of non-monotonic reasoning and logic programming. The idea of ASP is to represent a given computational problem by a logic program whose answer sets correspond to solutions, and then use a solver to find such solutions [12]. The language of ASP is very expressive, indeed all problems in the second level of the polynomial hierarchy are expressible in ASP [4]. Moreover, the applications of ASP nowadays belong to several fields from Artificial Intelligence to Knowledge Management [2]. The development of efficient and fast ASP systems is, thus, a crucial task made even more challenging by existing and new-coming applications.

As witnessed by the ASP competition series (see [3] for the most recent), several efficient ASP solvers have been proposed up to now, which are based on different solving techniques ranging from ASP-specific approaches to translation to SAT/Difference Logic. Inspired by the recent research results on the neighbor fields of SAT and QBF, where inductive techniques for algorithm selection were applied with success [18,16], we have developed ME-ASP, a *multi-engine* solver for propositional ASP programs.

In this paper we describe this new system. In order to obtain a robust ASP solver, i.e., a system able to perform well across a wide set of problem domains, we leverage a number of efficient ASP systems (e.g., [6,14,10,11,9,17]), and we apply machine learning techniques for inductively choosing, among the available ones, the "best" solver to be run on the basis of the characteristics, also called features, of the input program at hand.

We also report the results of some experiments carried out on the grounded version of all benchmarks employed in the "System Track" of the 3rd ASP Competition [3] falling in the "*NP*" and "*Beyond NP*" categories of the competition, that show the state-of-the-art performance of our multi-engine solver; indeed, ME-ASP is able to solve substantially more instances than the winner of the "System Track" of the 3rd ASP Competition.

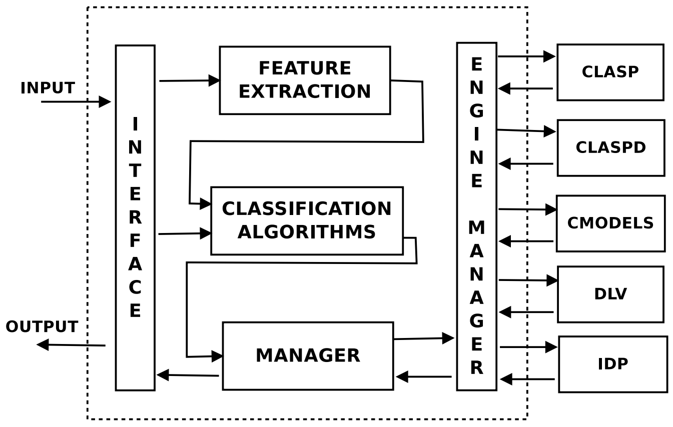


Fig. 1. The architecture of ME-ASP. The dotted box denotes the whole system and, inside it, each solid box represents its modules. Arrows denote functional connections between modules.

It is worth mentioning that, machine learning techniques have been already applied to ASP solving, i.e. CLASPFOLIO and DORS [5,1]. In particular, the CLASPFOLIO system was conceived and implemented for selecting the "best" heuristic configuration of the CLASP solver. An important difference with ME-ASP is that the application of algorithm selection strategies is limited in CLASPFOLIO (which is, actually, a unique binary including CLASP) to the variants of a single engine; moreover, CLASPFOLIO is not able to deal with ASP programs with syntactically-unrestricted disjunction.

2 The structure of ME-ASP

Figure 1 presents the architecture of ME-ASP[1]. Looking at the figure, we can see that ME-ASP is composed of the five modules described in the following.

INTERFACE manages both the input received by the user and the output of the whole system. It also dispatches the input data to the remaining modules, as denoted by the outgoing arrows. In particular, INTERFACE collects (i) the ground ASP program in ASP-Core format [3], and (ii) the classifier type and its inductive model.

FEATURE EXTRACTION extracts the syntactic features of the input ground program, as detailed in [13]. The CPU time spent for the extraction is negligible.

CLASSIFICATION ALGORITHMS is devoted to the prediction of the engine to run. It implements five different inductive models, namely Aggregation Pheromone density based pattern Classification, Decision Rules, Decision Trees, Nearest-neighbor, and Support Vector Machine. We implemented the first one following the methodology described in [8], while the remaining ones are built on top of the RAPIDMINER library [15]. This module receives as input both the classifier type and its inductive model (from INTERFACE) and a vector of features (from FEATURE EXTRACTION). It returns to MANAGER the name of the predicted engine.

[1] ME-ASP is available for download at `http://www.mat.unical.it/ricca/me-asp`

Table 1. Results on the 10 grounded instances for each domain evaluated at the 3rd ASP competition. The instances of the DisjunctiveScheduling, PackingProblem and WeightAssignmentTree are not solved by any solver. The table is organized as follows. In the first column we report the benchmark, followed by three groups of columns, each one related to an evaluated solver. Each group is composed of two columns, namely "#Solved" (i.e., the total amount of solved instances within the time limit) and "Time" (i.e., the total CPU time spent on the solved instances).

Problem	ME-ASP		CLASPD		SOTA	
	#Solved	Time	#Solved	Time	#Solved	Time
GraphColouring	4	527.67	3	302.09	4	523.38
HanoiTower	9	1107.67	2	416.94	9	1041.76
KnightTour	8	755.67	8	544.21	8	728.12
Labyrinth	5	415.43	3	275.12	5	344.95
MazeGeneration	10	52.15	10	32.63	10	31.37
MinimalDiagnosis	10	1889.46	10	1859.86	10	69.01
MultiContextSystemQuerying	10	687.93	10	1177.08	10	87.45
Numberlink	8	254.01	7	47.32	8	226.06
SokobanDecision	9	1312.74	7	487.50	9	1182.24
Solitaire	5	767.98	2	57.98	8	1238.21
StrategicCompanies	5	1290.27	3	484.14	5	1152.00
TOTAL	83	9060.98	65	5684.87	86	6624.55

ENGINE MANAGER manages the interaction with the engines. It receives from MANAGER information about the engine to fire. At the end of the engine computation, ENGINE MANAGER returns to MANAGER the result. Finally, MANAGER works as a coordinator of ME-ASP modules, and it also provides the final result to INTERFACE.

The engines of ME-ASP, as depicted in Figure 1 (the rightmost boxes) are five state-of-the-art ASP solvers, namely CLASP [6] and its disjunctive version CLASPD, CMODELS [11], DLV [10], and IDP [14]; nonetheless, the architecture of ME-ASP is modular and allows one to easily update the engines set with additional solvers. Finally note that engines are used as "black-boxes", i.e., ME-ASP interacts with them via system calls.

3 Performance at a Glance

The experiments were carried out on CyberSAR, a cluster comprised of 50 Intel Xeon E5420 blades equipped with 64 bit Gnu Scientific Linux 5.5. The resources granted to the solvers are 600s of CPU time and 2GB of memory. Time measurements were carried out using the `time` command shipped with Gnu Scientific Linux 5.5.[2] In Table 1 we report the results of the ME-ASP version using Decision Trees as classifier in comparison with CLASPD – the winner of the "System Track" of the 3rd ASP Competition – and the state of the art (SOTA) solver, i.e., considering a problem instance, the oracle that always fares the best among available solvers.

[2] We remind that these are different hardware setting w.r.t. the 3rd ASP competition in both computer architecture and memory limits; importantly, the inputs were pre-grounded and saved in ASP-Core format.

Looking at Table 1, we can see that ME-ASP solves 18 instances more than CLASPD. More, here it is very interesting to note that its performance is very close to the SOTA solver which, we remind, has the ideal performance that we could expect on these instances with these engines. More details and additional experimental data concerning ME-ASP settings (i.e., solver selection, program features, solver training, and classification algorithms) can be found in [13].

References

1. Balduccini, M.: Learning and using domain-specific heuristics in ASP solvers. AICOM 24, 147–164 (2011)
2. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press, Tempe (2003)
3. Calimeri, F., Ianni, G., Ricca, F., Alviano, M., Bria, A., Catalano, G., Cozza, S., Faber, W., Febbraro, O., Leone, N., Manna, M., Martello, A., Panetta, C., Perri, S., Reale, K., Santoro, M.C., Sirianni, M., Terracina, G., Veltri, P.: The Third Answer Set Programming Competition: Preliminary Report of the System Competition Track. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 388–403. Springer, Heidelberg (2011)
4. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive Datalog. ACM Transactions on Database Systems 22(3), 364–418 (1997)
5. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., Schneider, M.T., Ziller, S.: A Portfolio Solver for Answer Set Programming: Preliminary Report. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 352–357. Springer, Heidelberg (2011)
6. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-driven answer set solving. In: Proc. of IJCAI 2007, pp. 386–392. Morgan Kaufmann Publishers (2007)
7. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)
8. Halder, A., Ghosh, A., Ghosh, S.: Aggregation pheromone density based pattern classification. Fundamenta Informaticae 92(4), 345–362 (2009)
9. Janhunen, T., Niemelä, I., Sevalnev, M.: Computing Stable Models via Reductions to Difference Logic. In: Erdem, E., Lin, F., Schaub, T. (eds.) LPNMR 2009. LNCS, vol. 5753, pp. 142–154. Springer, Heidelberg (2009)
10. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV System for Knowledge Representation and Reasoning. ACM TOCL 7(3), 499–562 (2006)
11. Lierler, Y.: Disjunctive Answer Set Programming via Satisfiability. In: Proc. of LPNMR 2005. LNCS, vol. 3662, pp. 447–451. Springer, Heidelberg (2005)
12. Lifschitz, V.: Answer Set Planning. In: Proc. of ICLP 1999, Las Cruces, New Mexico, USA, pp. 23–37. The MIT Press (November 1999)
13. Maratea, M., Pulina, L., Ricca, F.: Applying machine learning techniques to ASP solving. Number CVL 2012/003, p. 21. University of Sassari Tech. Rep. (March 2012)
14. Mariën, M., Wittocx, J., Denecker, M., Bruynooghe, M.: SAT(ID): Satisfiability of Propositional Logic Extended with Inductive Definitions. In: Kleine Büning, H., Zhao, X. (eds.) SAT 2008. LNCS, vol. 4996, pp. 211–224. Springer, Heidelberg (2008)
15. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In: Proc. of KDD 2006, pp. 935–940. ACM (2006)
16. Pulina, L., Tacchella, A.: A self-adaptive multi-engine solver for quantified boolean formulas. Constraints 14(1), 80–116 (2009)
17. Simons, P., Niemelä, I., Soininen, T.: Extending and Implementing the Stable Model Semantics. Artificial Intelligence 138, 181–234 (2002)
18. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Satzilla: Portfolio-based algorithm selection for SAT. Journal of Artificial Intelligence Research 32, 565–606 (2008)

A System for the Use of Answer Set Programming in Reinforcement Learning

Matthias Nickles

Department of Computer Science, Technical University of Munich
Boltzmannstr.3, D-85748 Garching, Germany
nickles@cs.tum.edu

Abstract. We present the software system *QASP* which integrates Reinforcement Learning (RL) with Answer Set Programming (ASP). Our framework allows for the ASP-based representation, computation and constraining of states and actions (and other events), and for the use of AnsProlog for the specification of action- and event-calculi and background knowledge for RL.

Keywords: Answer Set Programming, Relational Reinforcement Learning, Logical Planning, Satisfiability Checking, Statistical Relational Learning.

1 Introduction

Relational Reinforcement Learning (RRL) [1,4,5,7,6] enhances traditional Reinforcement Learning (RL) with expressive relational representation formats for actions, states and prior knowledge. This paper presents the software *QASP* ("Q-learning with Answer Set Programming") which implements an approach to RRL based on *Answer Set Programming* (ASP), making the strengths of ASP (such as a high degree of declarativity and an efficient approach to combinatorial search problems and automated planning) in practice available to RL. To our best knowledge, QASP has been the first software system which combined Reinforcement Learning with ASP.

With QASP, ASP can be used for the computation or constraining of state transitions, rewards and possible actions (using answer set programs), for background knowledge, for planning, and for the representation of states (in form of answer sets) and actions (as atoms). Various third-party ASP-grounders and solvers are supported, and as RL-algorithms (Relational) *Q-learning* and *SARSA-learning*, and *Relational Instance-Based Learning* (based on distance-weighted *k-Nearest Neighbor Learning*) (a relational generalization technique) [4,7] can be specified. Furthermore, the system allows for the induction of probabilistic first-order logical decision trees [5]. Our software also includes a GUI for the graphical presentation of results, policies and state/action/value spaces, and for the debugging of learning experiments.

2 Framework

In the following, an outline of the overall system architecture is presented. For its theoretical background, algorithms and preliminary experimental results, please see [3].

In our framework, a single agent learns an optimal (or near-optimal) policy for acting in a deterministic or non-deterministic environment. As learning approach either off-policy (Q-learning) or on-policy (SARSA) temporal-difference learning can be used. States and agent actions in the Markov Decision Process (MDP) are represented in a relational format (as (sets of) atoms), and rules governing state updates, rewards, possible actions etc. are specified in form of some user-defined answer set program. At this, ASP in QASP is not restricted to a certain event of action language - basically any answer set program can be used, provided there is some notion of fluents and event-triggered state updates. E.g., we have successfully used the *Circumscriptive Event Calculus*, which can for finite domains be translated into AnsProlog [8].

In RL, each learning episode consists of several learning steps which correspond to discrete time steps. By default, the ASP-solver is called at each learning step. Each solver call results in a set of stable models from which the following information is extracted (cf. Fig. 1): 1) the next state (or a set of possible new states), given the previous state, the agent's most recent action, and any background rules. 2) the set of actions which are logically possible at this point and from which the learning agent chooses its next action, 3) which rewards were achieved (numerical rewards are ordinary fluents in our framework, which allows for Markovian as well as non-Markovian rewards). After the learning agent has chosen and performed the next action and has observed the new state, its knowledge base in form of a logic program is extended appropriately (depending on the solver type (conventional or incremental), either the entire updated knowledge base or just the new piece of information acquired by the agent is submitted to the solver).

Both deterministic domains and non-deterministic / stochastic domains are supported. Non-determinism (and to a rather limited degree even probability distributions) of states and rewards, but also choice among actions, can be modelled by various means, e.g., using disjunctive logic programs (if the ASP solver supports these) or *choice constructs*.

By putting constraints on possible states and actions, the agent designer can formally enable and confine what the learning agent is able to (or must) conceive and do at each point in time. In particular, this allows for the "injection" of planning goals or sub-goals into the MDP, with the plans being computed by the ASP-solver, which effectively blends ASP-based planning with reinforcement learning (cf. [3] for details). Using an appropriate axiomatic basis such as the Event Calculus, ASP can also be used to specify action sequences, exploration policies and sub-policies, and for the simulation or processing of external events. As a simple example for how QASP operates, consider the following snippet of an AnsProlog program which models the well-known Blocks World domain:

```
time(minstep..maxstep).
#domain time(TIME).
holdsAt(on(X,Y),TIME) :- happens(stack(X,Y),TIME-1).
:- happens(stack(X,Y),TIME), holdsAt(on(Z,X),TIME).
holdsAt(reward(1),TIME) :- holdsAt(on(blueblock,table),TIME).
[...]
```

This program has a number of stable models with each stable model consisting of a number of atoms, e.g., `holdsAt(on(redblock,table),3)` and `happens(stack (greenblock,redblock),3)`, meaning that (according to the respective stable model) the red block is on the table at time 3 and, also at time 3, the agent attempts to

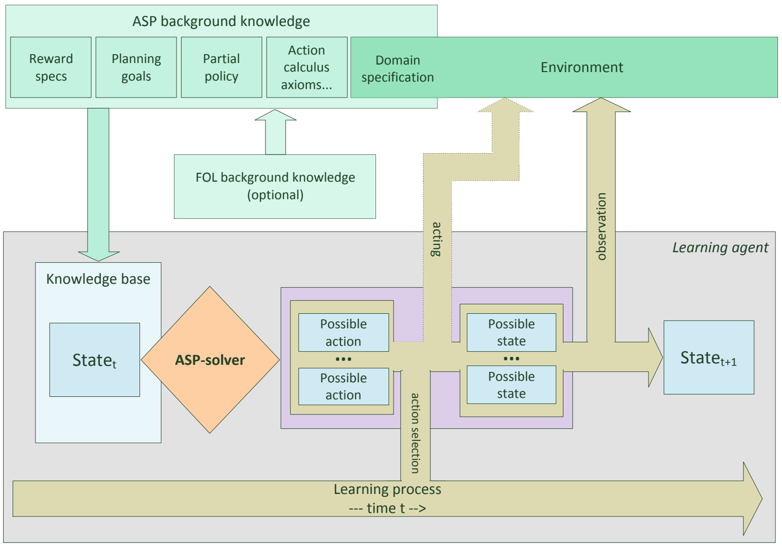


Fig. 1. Schematic architecture of the QASP learning core

stack the green block on top of the red block. Terms like `on(redblock, table)` represent fluents. A (Markov) state is represented as a set of fluents which *hold at* some time step, according to a certain stable model. A certain state can correspond to multiple time steps, i.e., a state can of course occur repeatedly during a learning episode. Rules like the one in line 3 determine which effects actions have on the truth of fluents, i.e., they determine the next state (or the set of potential next states in a non-deterministic environment). Stable models and their underlying rules also tell QASP which actions are logically possible at a certain time step. E.g., the action preconditioning rule in line 4 specifies that instances (groundings) of `stack(X,Y)` are *executable* actions provided there is currently no block on top of block `X`. Rule 5 specifies that there is a reward of 1 given in a state where the blue block is on the table.

3 Software Specifics

QASP is mostly programmed in Scala (a smaller, GUI-related part is written in Java). The current prototype (for Linux, Windows and Mac OS X) and its documentation can be found at `http://www.model.in.tum.de/~nickles/QASP/index.html` To run QASP, a Java JRE (version 6 or higher) and an external ASP-grounder and ASP-solver are required (grounders and solvers are not included in QASP and need to be installed separately). QASP supports a wide range of ASP-solvers and -grounders. Basically, any solver which understands the output of the grounder *lparse* [9] and any grounder which produces output in this format can be used, in particular *lparse* itself and the solver *smodels* [9]. In our experiments, we achieved very good results using *clingo* (a combined ASP-grounder + ASP-solver), *gringo* (a grounder) and *clasp* (a solver) from the Potsdam ASP collection "Potassco"[2][1]. QASP also supports

[1] `http://potassco.sourceforge.net/`

incremental ASP-solving, namely using *Cmodels* [10] (an incremental solver based on SAT solvers) and *oclingo* (an incremental reactive solver) [2]. Background knowledge in first-order logic syntax is supported by means of an optional third-party conversion tool (*F2LP* [2]). Support of further solvers (e.g., *DLV*) is planned for the near future.

QASP includes a rich set of tools for visualization, control and debugging of learning experiments. Experiments can be processed in single step mode in order to control state (stable model) transitions triggered by action events. Further features are model learning (in form of probabilistic first-order decision tree induction), shortest plan estimation for stochastic domains, graphical policy visualization in form of automata, and 3D graphical visualization of the state/action/value space or a sample thereof.

4 Conclusion

We have presented a software system for the integration of ASP and Reinforcement Learning. The system already supports a wide range of ASP tools and implements various approaches to on-policy and off-policy RL, however, there is still much room for improvement and extensions. For the near future it is planned to support further ASP solvers, to integrate QASP with ILP systems for improved model induction, and to enable learning using large sets of examples from online resources.

Acknowledgement. This work was supported by Deutsche Forschungsgemeinschaft (DFG).

References

1. Dzeroski, S., De Raedt, L., Blockeel, H.: Relational Reinforcement Learning. In: Procs. ICML 1998. Morgan Kaufmann (1998)
2. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam Answer Set Solving Collection. AI Communications 24(2) (2011)
3. Nickles, M.: Integrating Relational Reinforcement Learning with Reasoning about Actions and Change. In: Muggleton, S.H., Tamaddoni-Nezhad, A., Lisi, F.A. (eds.) ILP 2011. LNCS (LNAI), vol. 7207, pp. 255–269. Springer, Heidelberg (2012)
4. Driessens, K., Ramon, J.: Relational Instance Based Regression for Relational Reinforcement Learning. In: Procs. ICML (2003)
5. Croonenborghs, T., Ramon, J., Bruynooghe, M.: Towards Informed Reinforcement Learning. Procs. In: Workshop on Relational Reinforcement Learning at ICML 2004 (2004)
6. Van Otterlo, M.: The Logic of Adaptive Behavior. IOS Press, Amsterdam (2009)
7. Rodrigues, C., Gerard, P., Rouveirol, C.: Relational TD Reinforcement Learning. In: Procs. EWRL 2008 (2008)
8. Kim, T.-W., Lee, J., Palla, R.: Circumscriptive Event Calculus as Answer Set Programming. In: Procs. IJCAI 2009 (2009)
9. Niemelä, I., Simons, P.: Smodels - An Implementation of the Stable Model and Well-Founded Semantics for Normal Logic Programs. In: Fuhrbach, U., Dix, J., Nerode, A. (eds.) LPNMR 1997. LNCS, vol. 1265, pp. 420–429. Springer, Heidelberg (1997)
10. Giunchiglia, E., Lierler, Y., Maratea, M.: Answer Set Programming based on Propositional Satisfiability. Journal of Automated Reasoning (2006)

[2] `http://reasoning.eas.asu.edu/f2lp/`

The Tableau Prover Generator MetTeL2

Dmitry Tishkovsky, Renate A. Schmidt, and Mohammad Khodadadi[⋆]

School of Computer Science, The University of Manchester, UK

Abstract. This paper introduces METTEL², a tableau prover generator producing JAVA code from the specification of a tableau calculus for a logical language. METTEL² is intended to provide an easy to use system for non-technical users and allow technical users to extend the generated implementations.

1 Introduction

METTEL² is a tool for generating tableau provers from specifications of a syntax and a tableau calculus for a logical theory. The syntax and tableau rule specification languages of METTEL² are designed to be as simple as possible for the user and to be as close as possible to the traditional notation used in logic and automated reasoning textbooks. At the moment the syntax specification language is limited to (possibly multi-sorted) propositional languages with finitary connectives and uses a simplified BNF. The tableau calculus specification language covers different types of tableau calculi that fit the traditional representation of tableau rules of the form $X_0/X_1 \mid \cdots \mid X_m$ creating a branching point with m successors in tableau derivations. The X_i denote finite sets of expressions of the given logical theory. X_0 is the set of premises and $X_1, \ldots, X_m$ are the sets of conclusions of the rule. Many labelled semantic tableau calculi for modal, description, hybrid and superintuitionistic logics belong to this paradigm.

METTEL² is complementary to the tableau synthesis framework introduced in [4]. The framework effectively describes a class of logics for which tableau calculi can be automatically generated. This class includes many modal, description, intuitionistic and hybrid logics. The framework provides a theoretical foundation for sound, complete and terminating implementations of tableau procedures for logics from the mentioned class and, in particular, for many logics which can be specified in METTEL². The scope of METTEL² extends however that of tableau calculi derived in the framework and is not limited to semantic or labelled tableau calculi.

The tableau reasoning core of METTEL² is considerably based on the generic prover METTEL [6], but has been completely reimplemented and several new features have been added. Notable new features are dynamic backtracking and conflict-directed backjumping, as well as ordered forward and backward rewriting, which can be used to perform equality reasoning. There is support for different search strategies. The tableau rule specification language in METTEL² now

[⋆] This research is supported by UK EPSRC research grant EP/H043748/1.

allows the specification of rule application priorities thus providing a flexible and simple tool for defining rule selection strategies. To our knowledge, MetTeL^2 is the first system with full support of these techniques for *arbitrary* logical syntax.

The aim of the current implementation is to provide an easy to use prover generator with basic specification languages without sophisticated meta-programming features that might overwhelm non-technical users. For technical users, the generated code consists of a thoroughly designed hierarchy of public Java classes and interfaces that can be extended and integrated with other systems.

2 Application Areas and Experiences So Far

Software to generate code for provers is useful anywhere where automated reasoning is needed. The provers generated by MetTeL^2 output models for satisfiable problems on termination, so can be used for model generation purposes.

With MetTeL^2 a quick implementation of a tableau prover can be obtained and changes can be made without programming a single line of code. Prover generation is useful for obtaining provers for newly defined logics or new combinations of logics. This is particularly pertinent to an area such as multi-agent systems where the models are staggeringly complex. In ongoing work we are using MetTeL^2 in combination with the tableau synthesis framework to develop provers for multi-agent interrogative-epistemic logics [3]. For these logics and related dynamic epistemic logics there are almost no implemented reasoning tools. Therefore being able to generate tableau provers is very useful especially to researchers without the resources or expertise to implement automated reasoning tools themselves.

We have found MetTeL^2 useful for analysing tableau calculi under development whose properties are not known yet. For example, in research conducted for [2] we used MetTeL^2 to determine the refinability or unrefinability of tableau rules for a modal logic with global counting operators. MetTeL^2 can also be used to compare the effectiveness of different sets of tableau rules for the same logic. For example, with minimal effort it is possible to compare the effectiveness of standard tableau calculi with calculi following the KE approach where disjunction is handled by an analytic cut rule and a unit propagation rule (e.g., in terms of proof length, size of produced model, or other derivation statistics which are not tied to particular implementation details).

Concrete case studies we have undertaken with MetTeL^2 include implementing unlabelled tableau calculi for Boolean logic and three-valued Łukasiewicz logic, labelled tableau calculi for standard modal logics and description logics, and internalised tableau calculi for hybrid and description logics. We used MetTeL^2 to implement the first tableau decision procedure for $\mathcal{ALBO}^{\mathsf{id}}$, a description logic with the same expressive power as the two-variable fragment of first-order logic. Some of the specifications are available from the MetTeL^2 website `http://www.mettel-prover.org`.

3 The Implemented System

The parser for the specification of the user-defined logical language is implemented using the ANTLR parser generator. The specification is parsed and internally represented as an abstract syntax tree (AST). The internal ANTLR format for the AST is avoided for performance purposes. The created AST is passed to the generator class which processes the AST and produces the following files: (i) a hierarchy of JAVA classes representing the user-defined logical language, (ii) an object factory class managing the creation of the language classes, (iii) classes representing substitution, replacement, and rewrite orderings, (iv) an ANTLR grammar file for generating a parser of the user-specified language and the tableau language, (v) a main class for the prover parsing command line options and initiating the tableau derivation process, and (vi) JUNIT test classes for testing the parsers and testing the correctness of tableau derivations. The generated JAVA classes for syntax representation and algorithms for rule application follow the same paradigm as the generic prover METTEL [6].

METTEL^2 implements two general techniques for reducing the search space in tableau derivations: dynamic backtracking and conflict directed backjumping. Dynamic backtracking avoids repeating the same rule applications in parallel branches by keeping track of rule applications common to the branches. Conflict-directed backjumping derives conflict sets of expressions from a derivation. This causes branches with the same conflict sets to be discarded. Since METTEL^2 is a prover generator, dynamic backtracking and backjumping needed to be represented and implemented in a generic way, completely independent of any specific logical language and tableau rules.

The provers generated by METTEL^2 come with support for ordered backward and forward rewriting with respect to equality expressions appearing in the current branch. In the language specification, equality expressions can be identified with built-in keywords. Each JAVA class representing a tableau node keeps a rewrite relation completed with respect to all equality expressions appearing in a branch. Once an equality expression is added within a tableau node, backward rewriting is applied. This means the rewrite relation is rebuilt with respect to the newly added equality, and all expressions of the node are rewritten with respect to the rewrite relation. Forward rewriting (with respect to the current rewrite relation) is applied to all new expressions added to the branch during the derivation.

The core tableau engine of METTEL^2 provides various ways for controlling derivations. The default search strategy is depth-first left-to-right search, which is implemented as a MettelSimpleLIFOBranchSelectionStrategy request to the MettelSimpleTableauManager. The MettelSimpleTableauManager object manages tableau branches at the very top level: it stores branches for expansion and selects them in accordance with the specified branch selection strategy. Breadth-first search is implemented as a MettelSimpleFIFOBranchSelectionStrategy request and can be used after a small modification in the generated JAVA code. In future this will be configurable at the generation stage. A user can also implement their own search strategy and pass it to the MettelSimpleTableauManager.

The rule selection strategy can be controlled by specifying priority values for the rules in the tableau calculus specification. The rule selection algorithm checks the applicability of rules and returns a rule that can be applied to some expressions on the current branch according to rule priority values. If several rules are applicable preference is given to those which have smaller priority values. Rules within each priority group are checked for applicability sequentially. To ensure fairness for rules within the same priority group all rules in the group are checked for applicability an equal number of times. Again the user could implement their own rule selection strategy and modify the generated code.

To achieve termination for semantic tableau approaches some form of blocking is usually necessary. To generate a prover with blocking the user can specify a blocking rule similar to the unrestricted blocking rule from [5] as one of the rules of the tableau calculus. If the definition of the rule involves equality operators then rewriting is triggered (see above). Based on the results in [4,5], the blocking rule can be used to achieve termination for logics with the finite model property. The first of the two termination conditions in [4,5] is automatically true because the generated provers are equipped with ordered rewriting. The second termination condition can be satisfied by using appropriate priority values for tableau rules of the tableau calculus. By varying the specification of the blocking rule it is possible to perform blocking more selectively [1].

4 Conclusion

MetTeL2 can be downloaded from `http://www.mettel-prover.org`. A web-interface is provided, where a user can input their specifications in syntax aware textareas and generate provers. The user can either download the generated prover or directly run it in the web-interface.

References

1. Khodadadi, M., Schmidt, R.A., Tishkovsky, D.: An abstract tableau calculus for the description logic $\mathcal{SHOI}$ using unrestricted blocking and rewriting. In: Proc. DL 2012. CEUR Workshop Proceedings, vol. 846, pp. 224–234 (2012)
2. Khodadadi, M., Schmidt, R.A., Tishkovsky, D., Zawidzki, M.: Terminating tableau calculi for modal logic K with global counting operators (manuscript, 2012), `http://www.mettel-prover.org/papers/KEn12.pdf`
3. Minica, S., Khodadadi, M., Schmidt, R.A., Tishkovsky, D.: Synthesising and implementing tableau calculi for interrogative epistemic logics. In: Proc. PAAR 2012, pp. 109–123 (2012)
4. Schmidt, R.A., Tishkovsky, D.: Automated synthesis of tableau calculi. Log. Methods Comput. Sci. 7(2:6), 1–32 (2011)
5. Schmidt, R.A., Tishkovsky, D.: Using tableau to decide description logics with full role negation and identity (2011) (manuscript), `http://www.mettel-prover.org/papers/ALBOid.pdf`
6. Tishkovsky, D., Schmidt, R.A., Khodadadi, M.: MetTeL: A Tableau Prover with Logic-Independent Inference Engine. In: Brünnler, K., Metcalfe, G. (eds.) TABLEAUX 2011. LNCS, vol. 6793, pp. 242–247. Springer, Heidelberg (2011)

Author Index